

# Ontwerpdocument



***Digital Twin 3.0***

***Handpicked Agencies***

***Breda***

Door: Koen Pijnenburg

## **Introductie**

Ongeveer 90% van de data science projecten haalt het niet tot productie (Odegua, 2020). Dit komt doordat het proces wat gebruikt wordt om machine learning modellen te ontwikkelen niet perse goed aansluit bij de software engineering & DevOps processen. Om dit te voorkomen bij het Digital Twin 3.0 project zal hier rekening mee gehouden worden tijdens het ontwerp.

In dit document zal dit ontwerp opgesteld worden via de C4 methode. Dit is een ontwerp framework wat op vier abstractieniveaus de applicatie toelicht.

1. Context; Op hoog niveau wordt weergegeven wie er op welke manier interacteert met de systemen.
2. Container; Globaal overzicht van de software architectuur. Focus op technologie keuze en manier van communicatie.
3. Component; Overzicht van welke onderdelen samen de container vormen.
4. Code; Laag niveau overzicht van de code door, bijvoorbeeld, klasse diagrammen.

# Inhoudsopgave

<b>Vooronderzoek</b>	<b>3</b>
Opslag en ophalen	3
Frameworks en tooling	4
Feedback en iteratief	6
<b>C1: Context</b>	<b>7</b>
<b>C2: Containers</b>	<b>8</b>
<b>C3: Components</b>	<b>9</b>
<b>C4: Code</b>	<b>10</b>
Routes	10
Database diagram	11

# Vooronderzoek

Voor begonnen kan worden met het ontwerpen van de applicatie zullen eerste de volgende onderwerpen onderzocht moeten worden:

1. Het opslaan en ophalen van de data en voorspellingen.
2. Frameworks en tooling die hierbij passen.
3. Hoe feedback verzameld wordt en iteratief verwerkt kan worden.

Per onderwerp zullen meerdere vragen gesteld en beantwoord worden. Deze methode is gebaseerd op een artikel van Odegua (2020).

## Opslag en ophalen

De applicatie zal data ophalen vanuit een database om modellen mee te trainen. In dit hoofdstuk zal verder onderzocht worden hoe dit momenteel gebeurt binnen de Twindle applicatie en of dit past bij de geplande uitbreidingen.

### **Hoe wordt de trainings data opgeslagen?**

Op meerdere locaties zijn sensoren geïnstalleerd die periodiek gegevens meten. Deze gegevens worden via een message queue systeem opgeslagen. Er wordt gebruik gemaakt van een MongoDB database. Dit betekent dat combinaties van metingen als een zogenaamd 'document' worden opgeslagen.

### **Hoe groot is deze data?**

Er wordt data opgeslagen sinds januari 2021. Iedere twee minuten worden er verschillende metingen verricht en de resultaten verstuurd naar de database. Iedere dag gaat dit, per locatie waar een Digital Twin actief is, om ongeveer ~5 MB aan data.

### **Hoe zal de data worden opgehaald voor het trainen?**

Het huidige systeem leest de complete database uit en verwerkt deze tot een Pandas DataFrame object. Momenteel is dit nog geen probleem maar wanneer er grotere hoeveelheden data verwerkt moeten worden zal dit erg langzaam worden.

### **Hoe zal de data worden opgehaald voor voorspellen?**

Tijdens de experimenteer fase zijn ARIMA modellen ontwikkeld die toegepast zullen worden in de applicatie. Deze modellen zullen periodiek worden geüpdatet met de meest recente data. Op dat moment worden nieuwe forecasts gemaakt en opgeslagen. Deze kunnen via een API opgehaald worden door het front-end van de applicatie.

## Frameworks en tooling

Tijdens de ontwikkeling van de applicatie zal gebruik gemaakt worden van frameworks en tools voor APIs en databases.

### API

Er zijn verschillende soorten API frameworks en libraries beschikbaar voor Python. Voor deze applicatie is een framework nodig wat goed kan dienen als back-end voor een webapplicatie. In de onderstaande tabel zijn de drie meest geschikte frameworks te zien samen met hun efficiëntie en populariteit (Purkayastha, 2021).

Framework	Efficiëntie	Populariteit	Open Source
Flask	Erg snel en hoogwaardig	Meest populaire Python REST API framework.	Ja
Tornado	Gemiddeld	Gemiddeld	Ja
FastAPI	Snel	Gemiddeld. Is groeiende.	Ja

Uit dit overzicht kan de conclusie gemaakt worden dat Flask de beste keuze is voor het API framework.

### Databases





De applicatie zal meerdere soorten data moeten opslaan. Dit zijn standaard tabel data, afbeeldingen en pickle files. Daarnaast zijn er optimalisatiemogelijkheden voor de database van de bestaande Digital Twin applicatie.

Ten eerste zal er een afweging gemaakt worden tussen relationele- en NoSQL databases. Onderstaand valt een overzicht te zien van deze afwegingen en hoe deze toepasbaar zijn op de situatie (robvet, 2021).

Afweging	Relational	NoSQL	Situatie
Volume	Consistent, gemiddeld to grote schaal.	Erg grote schaal.	Waarschijnlijk zal de applicatie, nog, niet op erg grote schaal worden uitgezet. Dit betekent dat relational databases hier beter bij passen.
ACID	Ondersteund	Niet ondersteund	Het is belangrijk dat de data consistent is. Hiervoor zijn ACID transacties nodig die relational databases eisen.

Dynamisch	Data moet vooraf gestructureerd worden door middel van schemas.	Zonder schema kan er erg dynamisch gewerkt worden.	Aangezien de applicatie nog in ontwikkeling is zou het handig zijn dat er geen schema gedefinieerd hoeft te worden. NoSQL voorziet dit.
Schrijf veiligheid	Vereist.	Snelheid is belangrijker.	Er zal niet veel data opgeslagen worden en het is van belang dat dit goed gebeurt. Hiervoor is relational beter geschikt.
Ophalen	Complexe queries mogelijk	Vaak wat simpeler.	Er zijn geen complexe relaties tussen de data. Hierdoor zou simpele ophaling van data ideaal zijn. NoSQL past hierbij.
Distributie	Centraal	Verspreid	De database zal zich niet buiten nederland bevinden. relationeel past hierbij.

Aangezien vier van de zes afwegingen beter passen bij een relationele database zal deze toegepast worden tijdens het project. Hiervoor zal de MySQL database en database connector toegepast worden. Zoals in het onderstaande overzicht staat past dit het best bij het Flask framework.

web framework	None	Flask	Flask	Django
ORM	SQLAlchemy	SQLAlchemy	SQLAlchemy	Django ORM
database connector	(built into Python stdlib)	MySQL-python	psycopg	psycopg
relational database	 SQLite	 MySQL	 PostgreSQL	 PostgreSQL

Afbeelding 1: Database & ORM overzicht

## Feedback en iteratief

Wanneer de applicatie live staat en de modellen periodiek worden geüpdatet met nieuwe data is het cruciaal om bij te houden of deze aan de gestelde eisen voldoen. Hiervoor zal regelmatig feedback verzameld en geanalyseerd moeten worden.

### **Hoe wordt feedback verzameld?**

Wanneer een model geupdate wordt zal deze ook gevalideerd worden. Tijdens de experimenteerfase is besloten dat hier voor de  $R^2$  en Root Mean Squared Error gebruikt zullen worden. Deze scores kunnen in een database opgeslagen worden zodat ze later geanalyseerd kunnen worden.

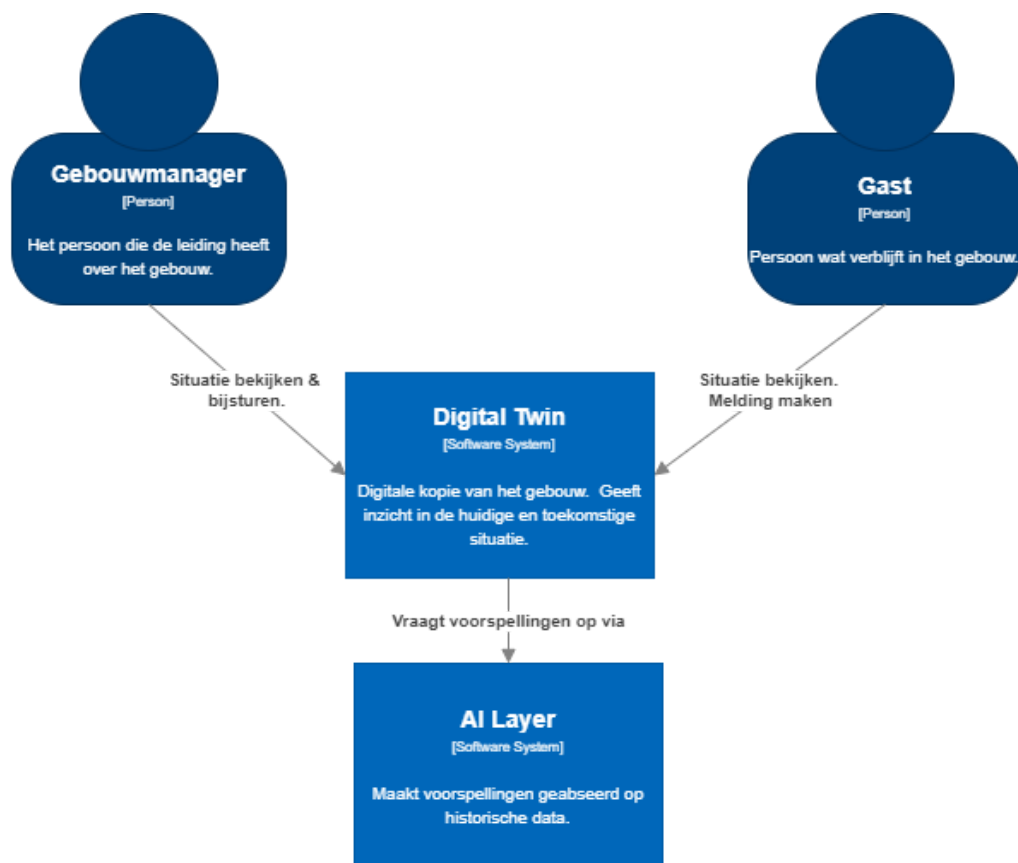
Daarnaast kunnen er grafieken gegenereerd worden waarin meer informatie over het presteren van de modellen valt terug te vinden. Deze kunnen ook opgeslagen worden zodat ze later bekeken kunnen worden. Gebaseerd op deze informatie kunnen de modellen worden bijgestuurd wanneer nodig.

### **Hoe is/wordt de CI/CD straat opgezet?**

BitBucket pipelines?

# C1: Context

Om context te geven aan het ontwerp zal er in dit hoofdstuk een diagram, op hoog niveau van abstractie, gemaakt worden van het complete systeem. Het doel van dit diagram is om weer te geven welke systemen er zijn en hoe zij samenwerken. Daarnaast wordt er in dit diagram een overzicht gegeven van de verschillende gebruikers en wat zij doen met de applicatie.



Stelsel Context Diagram voor Digital Twin 3.0

Zoals te zien valt in het diagram bestaat de applicatie uit twee systemen en twee verschillende gebruikersgroepen. Deze groepen zullen kort worden toegelicht.

## Systemen

De twee systemen die samen de Digital Twin 3.0 vormen zijn de huidige digital twin applicatie en, de nog te ontwikkelen, AI layer. De rest van het ontwerp zal zich vooral richten op de AI layer.

## Gebruikersgroepen

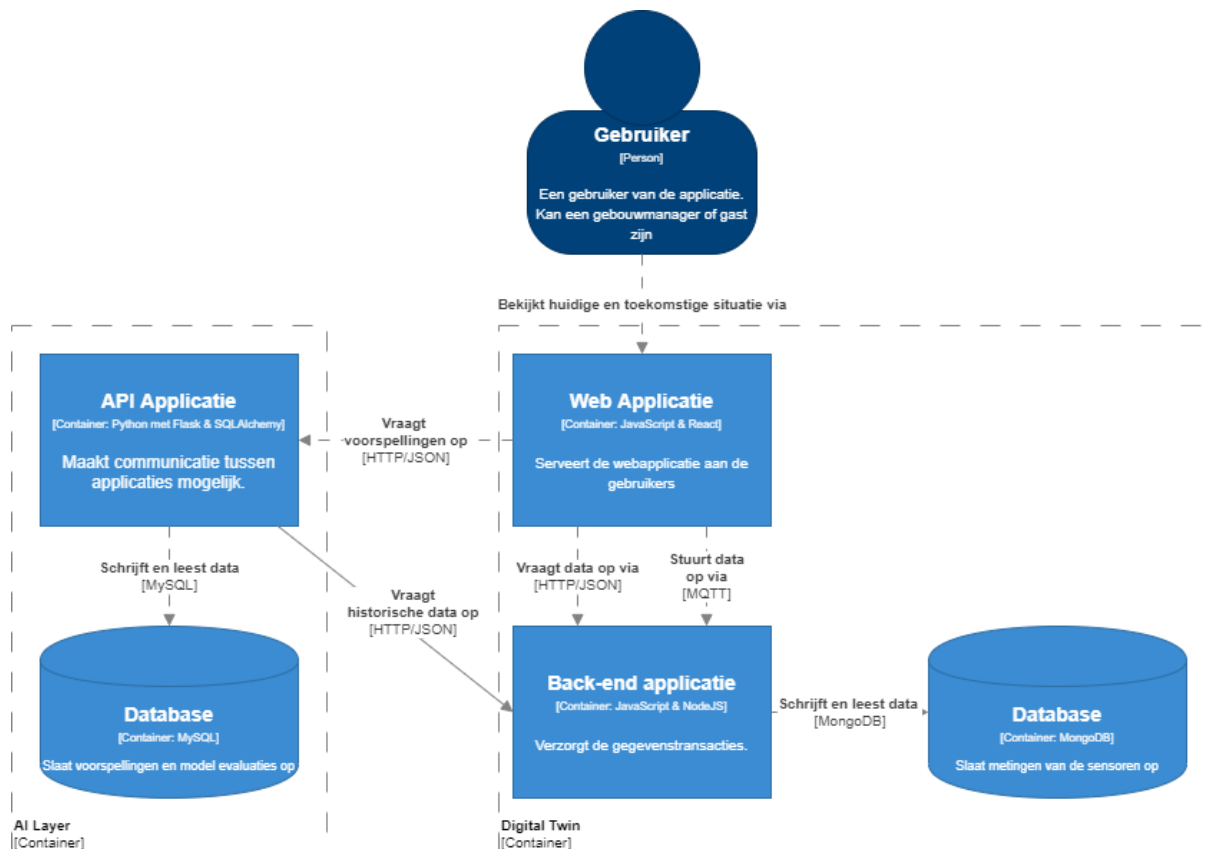
De twee gebruikersgroepen, gebouwmanager & gast, kunnen via de webapplicatie de huidige situatie binnen het gebouw bekijken. Hierop gebaseerd kan de manager, bijvoorbeeld, de verwarming wat lager zetten als het te warm wordt en de gast een melding maken wanneer de situatie voor hem oncomfortabel wordt.



## C2: Containers

In dit hoofdstuk zal er verder ingezoomt worden op de twee software systemen zoals deze zijn beschreven in het vorige hoofdstuk. Ze zullen verder onderverdeeld worden in containers. Dit zijn onderdelen van het systeem die apart van elkaar functioneren.

In dit diagram ligt de focus op de technologie keuzes en onderlinge communicatie tussen containers.

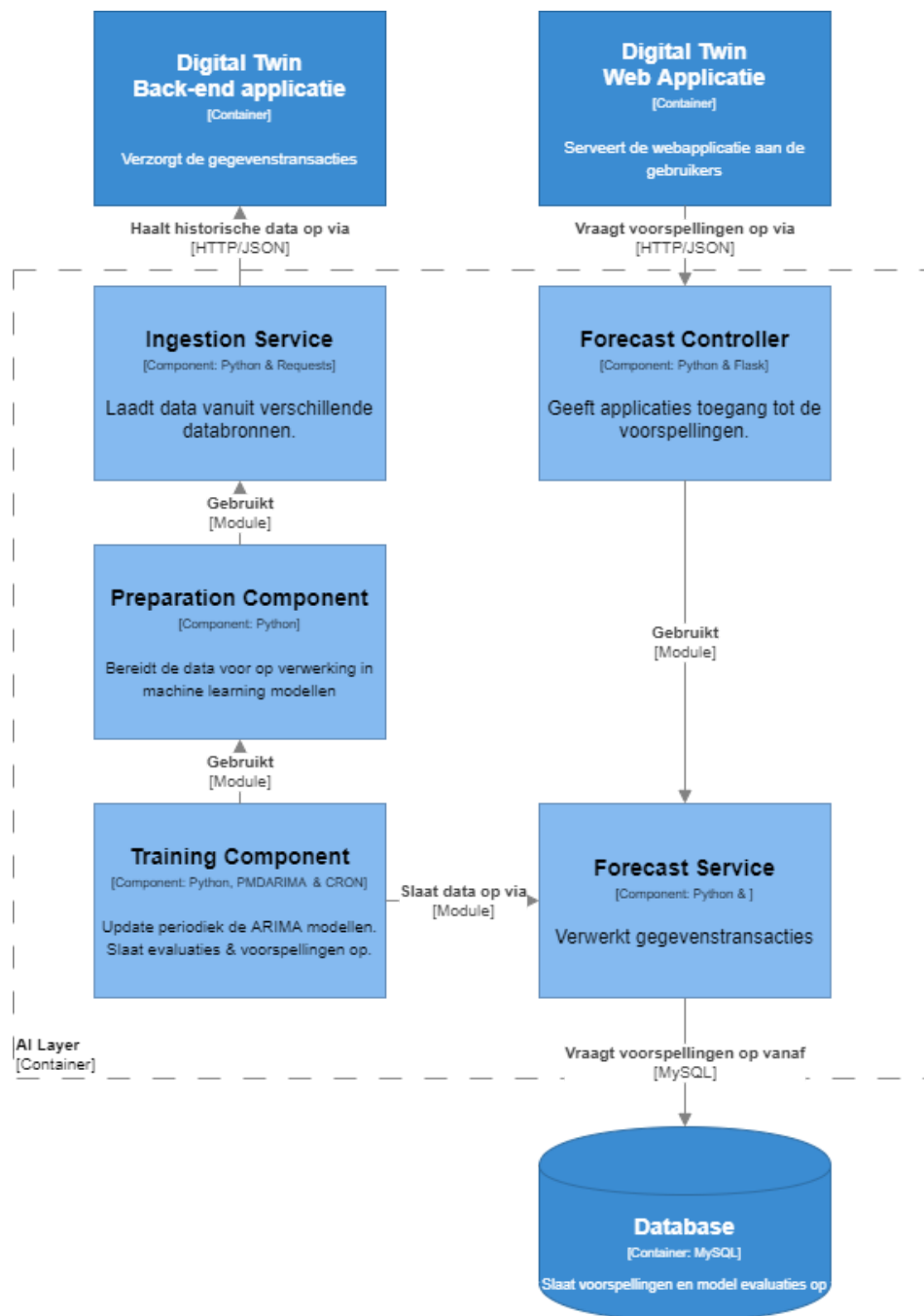


Container diagram voor de Digital Twin 3.0 applicatie

In het bovenstaande diagram kan gezien worden dat de webapplicatie voorspellingen ophaalt uit de AI-layer. Om deze voorspellingen te kunnen maken is historische data nodig die voorzien wordt door de back-end applicatie van de Digital Twin.

## C3: Components

Iedere container uit het vorige hoofdstuk bestaat uit een of meerdere componenten. In dit hoofdstuk zal de technologiekeuze, communicatie en verantwoordelijkheden van deze componenten worden toegelicht.



## C4: Code

### Routes

#### AI-layer

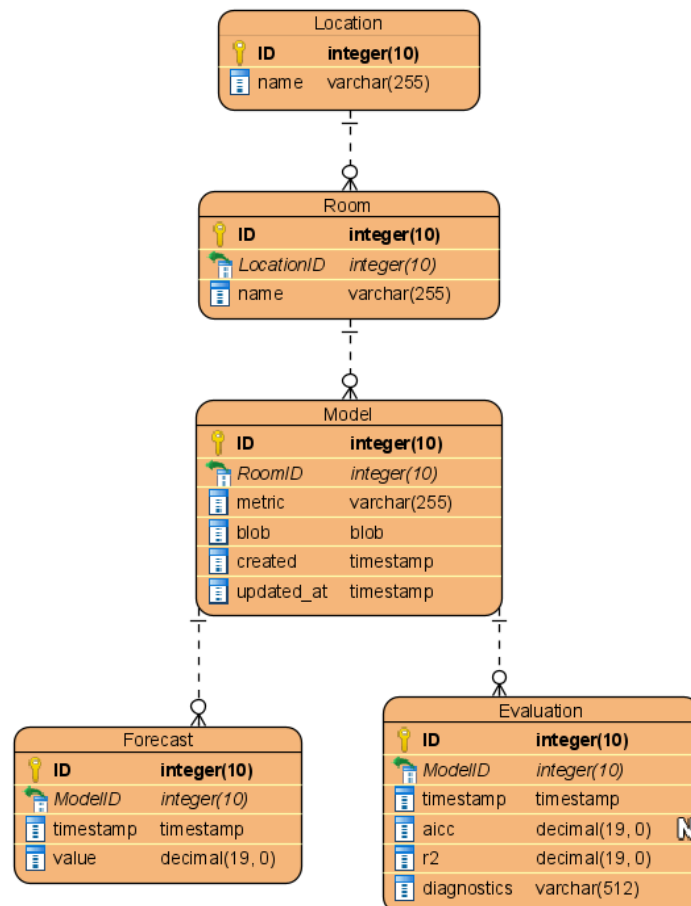
Method	Route	Parameters	type	Response
GET	/forecast/<location>/<room>/<metric>		number	<pre>[   {     "timestamp": "date",     "value": "float"   }   ... ]</pre>

#### Twindle backend

Method	Route	Parameters	type	Response
GET	/metrics	location room start end	string string datetime datetime	<pre>[   {     "timestamp": "date",     "tvoc": "float",     "temperature": "float",     "humidity": "float",     "co2": "float"   }   ... ]</pre>

## Database diagram

Naar aanleiding van de inzichten van de voorgaande hoofdstukken is het onderstaande database diagram opgesteld. In dit diagram worden de entiteiten en bijbehorende attributen en connecties weergegeven.



# Bronnen

Odegua, R. (2020, 22 oktober). How to put machine learning models into production. Stack Overflow Blog.

<https://stackoverflow.blog/2020/10/12/how-to-put-machine-learning-models-into-production/>

Purkayastha, S. (2021, 8 januari). Top 15 Python REST API Frameworks in 2021. The Last Call - RapidAPI Blog. <https://rapidapi.com/blog/best-python-api-frameworks/>