2025

Software Architectuur

BIOMETRISCH TOEGANGSSYSTEEM

PIOTR TADRALA | INFORDB

O-PP-CMK

INHOUDSOPGAVE

1.0 INLEIDING

1.3 DOEL VAN DIT DOCUMENT

Dit document omschrijft de software architectuur van het biometrisch toegangssysteem ontwikkeld voor InforDB. De focus van het systeem ligt op een schaalbaar systeem dat gebruik maakt van anti-spoofing technieken en multi-factor authentication. Doel van dit document is om inzicht te geven over de opbouw, werking, dependencies en de gemaakte keuzes van het systeem.

1.2 CONTEXT VAN HET SYSTEEM

Het systeem bestaat uit vijf onafhankelijke applicaties die samen het volledige systeem vormen. Elke applicatie vervult een specifieke taak en communiceert met de rest van het systeem om via gebruikersinteractie het slot te openen.

1. **Verification Node**: verzamelt beelden en PIN-invoer, stuurt raw data naar de hub.
2. **Hub**: Verzameling van microservices, kern van het systeem.
3. **Lock**: Microcontroller dat het slot mechanisme.
4. **Interface**: Interface van het systeem, maakt users en nodes management mogelijk.
5. **Backend**: Cloud applicatie voor API integraties en beheer op afstand.

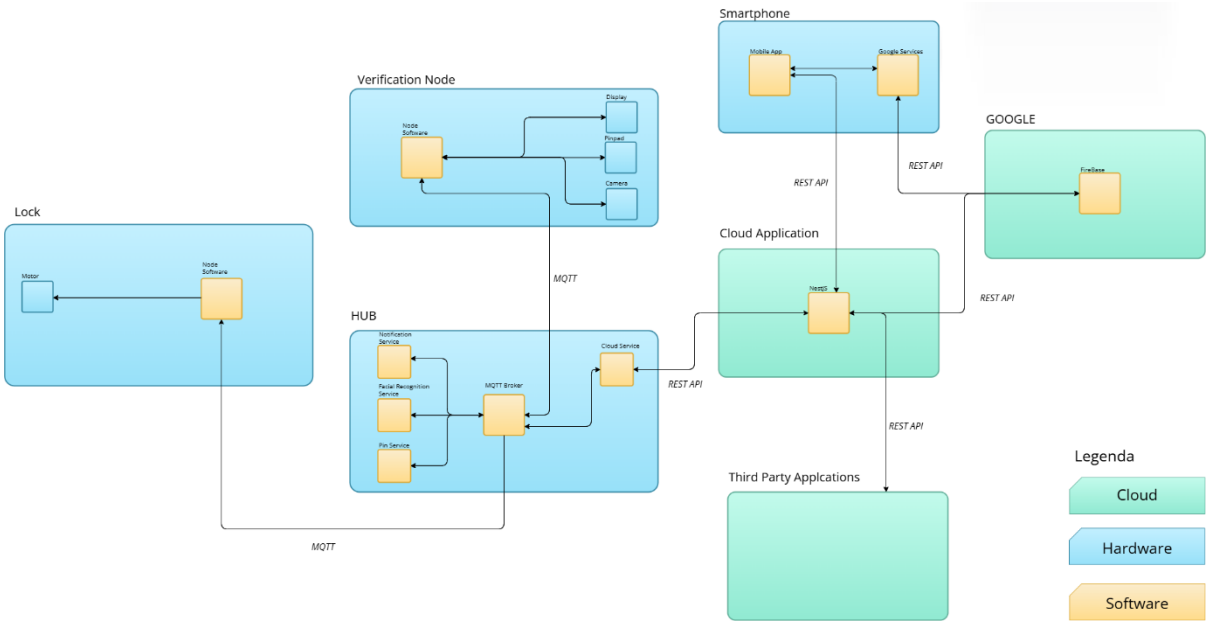
1.3 SCOPE

Dit document beschrijft de softwarearchitectuur van het biometrisch toegangssysteem en behandelt de vijf applicaties, namelijk: Backend, Interface, Hub, Verification Node en Lock. Voor elke component wordt de rol binnen het systeem toegelicht, de gebruikte technologieën en de gemaakte keuzes. Daarnaast wordt de volledige datastroom, van verificatie tot slotactiveringin kaart gebracht.

2.0 OVERZICHT ARCHITECTUUR

2.1 SYSTEEMOVERZICHT

Het toegangssysteem bestaat uit vijf losse applicaties die samenwerken op basis van het MQTT-pub/sub-model en REST API. Iedere applicatie draait in een eigen container of embedded omgeving en is verantwoordelijk voor één deelstap in de verificatie- en toegangsflow. Samen vormen ze een modulair geheel dat eenvoudig schaalbaar is.



2.2 COMPONENTEN OMSCHRIJVING

2.2.1 HUB

De hub vormt de kern van het systeem. Het is geen applicatie op zichzelf, maar bestaat uit een reeks gedockeriseerde microservices die via Docker Compose als één geheel worden gestart en beheerd.

De hub bestaat uit de volgende services:

- **MQTT Broker**
  Een Mosquitto MQTT-broker die functioneert als centrale communicatielaag tussen alle services en componenten. In de config zijn gebruikers gedefinieerd met de bijhorende permissions.

- **Facial Recognition Service**
  Een Python-applicatie die verantwoordelijk is voor het uitvoeren van gezichtsherkenning op binnenkomende webcam feed. Daarnaast beheert deze service ook gebruikersdata, zoals het toevoegen van nieuwe gezichten voor autorisatie.

- **Kiosk Service**
  Een HTTP-server die draait binnen het lokale netwerk. Bereikbar Via mDNS (bijvoorbeeld http://accesscontrol.home:3000) biedt deze service een preview van de webcamfeed. De applicatie is gebouwd in Node.js en draait een Express-server. De server stuurt de camerafeed door naar de frontend via WebSockets.

- **MQTT Service**
  Een Node.js-service die functioneert als de centrale logica van de hub. Aangezien de hub zelf geen enkele applicatie is, representeert deze service de HUB. De service is bijvoorbeeld gesubscribed op de presence-topic en detecteert welke apparaten beschikbaar zijn binnen het systeem.

- **Cloud Service**
  Een Node.js applicatie die als gateway werkt voor de communicatie tussen het systeem, en de backend via REST API.

- **Auth Service**
  De Authentication Service verwerkt de binnenkomende data om te verifiëren of de gebruiker geautoriseerd is het slot te openen.

2.2.2 VERIFICATION NODE

De Verification Node is een Python-applicatie die op een microcontroller draait en is gekoppeld aan een camera en een pinpad. Deze service verzamelt de camerafeed en de input van het pinpad en publiceert die via MQTT op de broker in de hub. Andere microservices, zoals de Kiosk Service, Recognition Service, Authentication Service, kunnen deze data vervolgens gebruiken om hun specifieke taken uit te voeren. De Verification Node beslist niet over het openen van het slot, het is enkel bedoeld voor het verzamelen van gebruikersinput.

2.2.3 LOCK

De slotsoftware is een C++ applicatie gebaseerd op het Arduino-framework en draait op een ESP32-microcontroller. De applicatie is minimalistisch opgezet om batterij te besparen en luistert enkel naa 3 topics, namelijk het openen van het slot, batterijniveau en identity presence.

2.2.4 INTERFACE

De interface, in de vorm van een mobiele applicatie, biedt beheermogelijkheden. De app is ontwikkeld in Angular en Ionic en maakt via het lokale netwerk verbinding met het systeem. Met de app kun je gebruikers beheren, push-notificaties bevestigen voor 2FA en een live preview van de camerafeed van de Verification Node bekijken.

2.2.5 BACKEND

Het backend is een gedockeriseerde Nest.js-applicatie. Het biedt functionaliteiten zoals API-integraties, push-notificaties, webhooks en remote management. Hiermee kunnen bepaalde aspecten van het systeem op afstand worden beheerd, zonder dat er een verbinding met het lokale netwerk nodig is.

2.3 COMMUNICATIE

2.3.1 MQTT

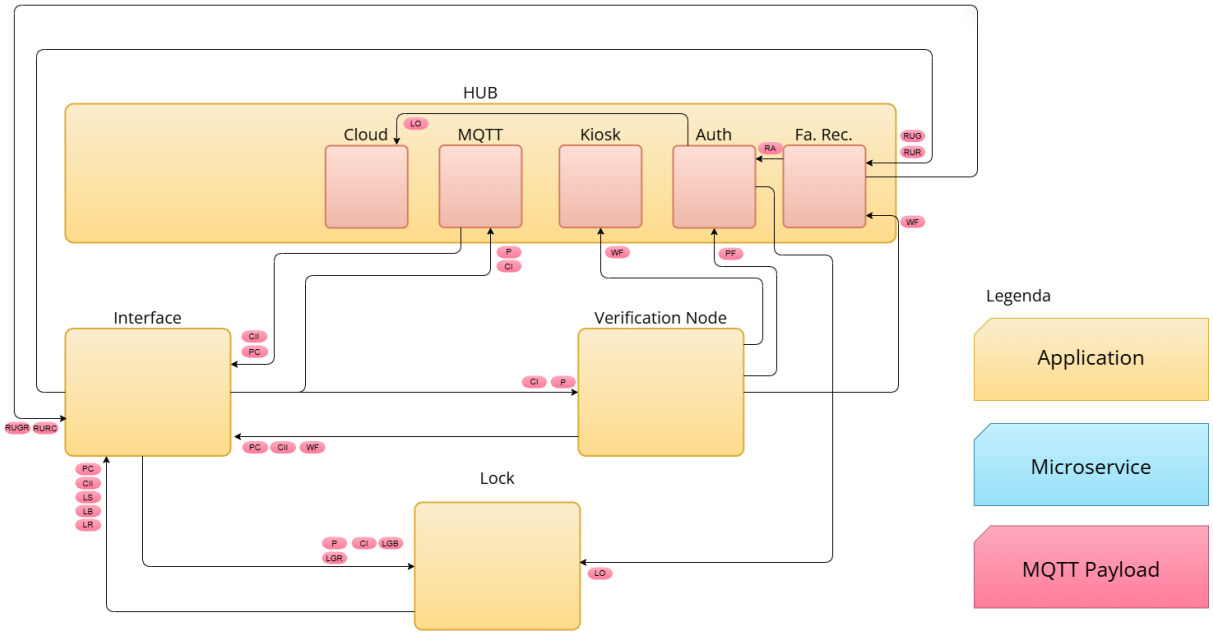
Communicatie binnen het lokale netwerk verloopt via MQTT. De MQTT-broker, die op de hub draait, beheert alle verbonden applicaties en services, inclusief hun permissions. De keuze voor MQTT is gebaseerd op uitgebreid onderzoek. Voor meer informatie zie *"Biometrisch Toegangssysteem Onderzoek"* en *"LoRa vs MQTT"*.

Topic	Index	Publishers	Subscribers	Omschrijving
Presence	P	Interface	MQTT Service (HUB), Verification Node, Lock	Gebruikt voor nodes discovery
Presence/confirm	PC	MQTT Service (HUB), Verification Node, Lock	Interface	Reactie op presence checks met unieke device identity
Client/identity	CI	Interface	MQTT Service (HUB), Verification Node, Lock	Gebruikt voor het opvragen van node identity
Client/identity/{id}	CII	MQTT Service (HUB), Verification Node, Lock	Interface	Specifieke client identity response
Webcam/feed	WF	Verification Node	Interface, Kiosk Service, Auth Service, Facial Recognition Service	Video feed van de camera
Pinpad/feed	PF	Verification Node	Auth Service	Input feed van de pinpad
Recognition/authorized	RA	Facial Recognition Service	Auth Service	Authorization trigger
recognition/user/register	RUR	Interface	Facial Recognition Service	Registratie van nieuwe gebruikers
recognition/user/register/confirm	RURC	Facial Recognition Service	Interface	Bevestiging van gebruikersregistratie
Recognition/users/get	RUG	Interface	Facial Recognition Service	Opvragen van gebruikers
Recognition/users/get/response	RUGR	Facial Recognition Service	Interface	Response met de gebruikers
Lock/open	LO	Auth Service	Lock, Cloud Service	Request om het slot te openen
Lock/get/battery	LGB	Interface	Lock	Opvragen van batterijstatus
Lock/get/rssi	LGR	Interface	Lock	Opvragen van wifi sterkte
Lock/status	LS	Lock	Interface	Statusupdate van het slot
Lock/battery	LB	Lock	Interface	Batterijstatus updates
Lock/rssi	LR	Lock	Interface	WIFI sterkte updates

2.4 DATA FLOW

In het onderstaand diagram is de route van elk MQTT topic weergegeven. De pijlen visualiseren de dependencies tussen publishers en subscribers. Zo loop er bijvoorbeeld een pijl van de interface naar de verification Node met de index P. In de bovenstaande tabel zie je dat P voor het presence topic staat, dat gebruikt wordt voor devices discovery op het lokale netwerk.

De authenticatieflow van de gebruiker verloopt als volgt: de Verification Node verzamelt camerabeelden en stuurt deze continu door naar de Facial Recognition Service. Zodra een gebruiker wordt herkend, stuurt de Facial Recognition Service een trigger naar de Auth Service. Vanaf dat moment wacht de Auth Service op invoer van de pinpad voor multi-factor authenticatie (indien nodig). Wanneer alle vereiste gegevens zijn ontvangen en de gebruiker succesvol is geverifieerd, stuurt de Auth Service een message naar de Lock om het slot te openen. Tegelijkertijd gebruikt de Cloud Service dit message ook om een webhook naar de cloud te sturen voor eventuele API integraties.



CI/CD

Het systeem maakt gebruik van een CI/CD-pipeline om testen en deployment te automatiseren via GitHub Actions. Bij het mergen van een feature branch met de production branch moet de nieuwe branch voldoen aan alle tests. Zodra een feature branch met de production branch wordt gemerged, start er automatisch een deployment flow.

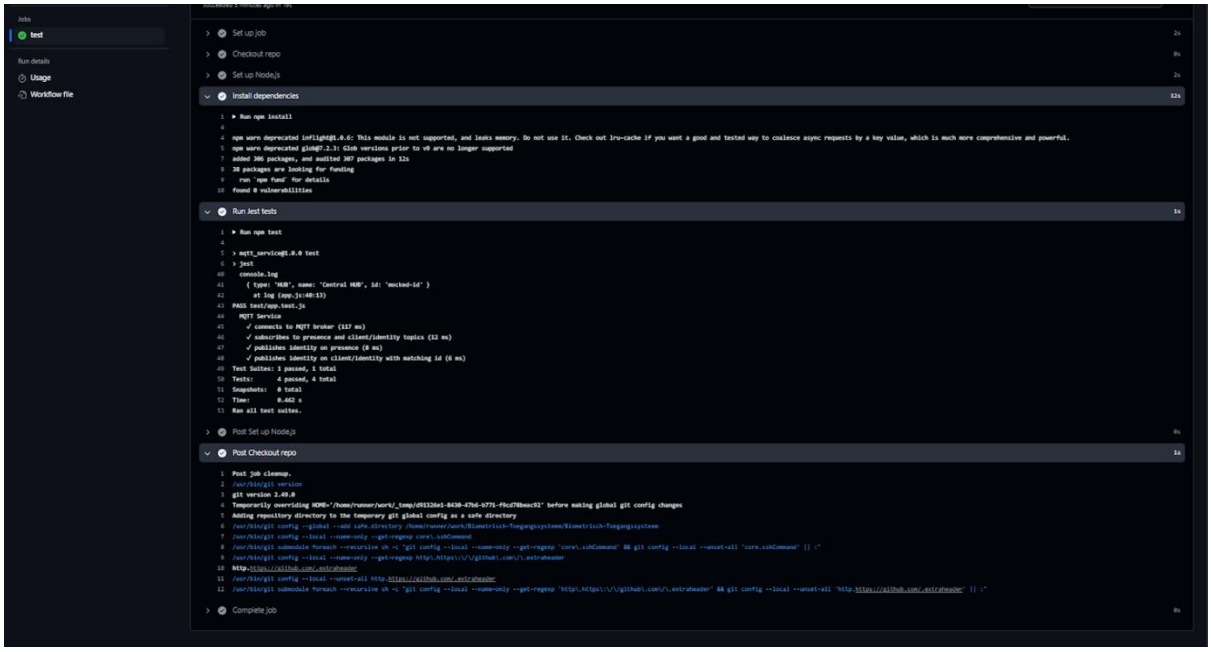
Voor de backend wordt de Docker-image gebuild en gepusht naar de container registry in Azure. Deze image wordt vervolgens gebruikt als de nieuwste versie.

Voor de microservices vergelijkt de hub om de vijf minuten de lokale commit hash met de laatste commit hash op GitHub. Zodra deze niet overeenkomen, haalt de hub automatisch de nieuwste code binnen en voert een build uit van de Docker Compose.

```
 PASS  test/app.test.js
  MQTT Service
    √ connects to MQTT broker (98 ms)
    √ subscribes to presence and client/identity topics (21 ms)
    √ publishes identity on presence (16 ms)
    √ publishes identity on client/identity with matching id (19 ms)

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        0.403 s, estimated 1 s
Ran all test suites.

C:\Users\piotr\Documents\infordb\Biometrisch-Toegangssysteem\Hub\mqtt_service (main -> origin) (mqtt_service@1.0.0)
λ
```



3.0 STACKS

3.1 HUB

De hub is opgebouwd volgens een microservicearchitectuur waarbij de services onderling communiceren via MQTT. Alle microservices worden afzonderlijk gedockeriseerd en vervolgens als één geheel opgebouwd en gestart met Docker Compose.

GEBRUIKTE TECHNOLOGIEËN

- **Docker + Docker Compose**
  Alle microservices worden gedockeriseerd als Docker-containers en als een stack beheerd via docker-compose.

- **Node.js**
  Gebruikt voor de MQTT Service, Kiosk Service, Cloud Service en Auth Service, vanwege snelle I/O-afhandeling.

- **Python**
  Gebruikt in de Facial Recognition Service vanwege de optimale ondersteuning voor AI- en beeldverwerkings libraries.

- **MQTT (Mosquitto Broker)**
  De centrale communicatielaag tussen alle services. Draait in een eigen container en is beveiligd met gebruikersauthenticatie en permissions.

- **WebSockets**
  Wordt gebruikt door de Kiosk Service om de webcamfeed in real-time door te sturen naar de frontend via een browserinterface.

- **REST API (Cloud Service)**
  Verzorgt de communicatie tussen de hub en de backend, bijvoorbeeld voor logboekverzending, webhook-acties en beheer op afstand.

3.2 VERIFICATION NODE

De Verification Node is geschreven in Python en draait op een microcontroller zoals een Raspberry Pi. De node is direct gekoppeld aan een camera en een pinpad en is verantwoordelijk voor het verzamelen van input.
De node stuurt deze input real-time via MQTT naar de broker op de hub. De Verification Node bevat geen eigen logica voor authenticatie of autorisatie, en functioneert puur als input collection.

GEBRUIKTE TECHNOLOGIEËN:

- **Python**
  Gebruikt vanwege optimale ondersteuning voor beeldverwerking, GPIO's en compatibiliteit met microcontrollers.
- **MQTT (Paho Client)**
  Verantwoordelijk voor het publiceren van webcambeelden en pinpad-input naar de hub.
- **Picamera2**
  Gebruikt voor het aansturen van de Raspberry PI Camera en het maken van video snapshots

- **GPIO**
  Gebruikt voor het uitlezen van de pinpad via de GPIO-pinnen

3.3 LOCK

De software die op het slot mechanisme draait, is een lightweight C++ applicatie gebaseerd op het Arduino-framework en draait op een ESP32-microcontroller. De applicatie is ontworpen met energie-efficiëntie in gedachten en voert slechts enkele eenvoudige taken uit.

GEBRUIKTE TECHNOLOGIEËN

- **C++ & Arduino Framework**
  Gebruikt voor het aansturen van de microcontroller
- **ESP32Servo**
  Gebruikt voor het aansturen van de MG90S Servo die het slot opent.
- **MQTT (ArduinoMqttClient)**
  Verantwoordelijk voor het ontvangen van de commando's om het slot te openen en het versturen van de status van het slot en de batterij.
- **WiFi (WiFi.h)**
  Voor verbinding met het lokale netwerk via WiFi.

3.4 INTERFACE

De interface is een cross-platform applicatie die gebruikers in staat stelt om het toegangssysteem te beheren. De app biedt functionaliteiten zoals gebruikersbeheer, push notificaties MFA en live Verification Node preview.

GEBRUIKTE TECHNOLOGIEËN

- **Angular & Ionic Framework**
  Gebruikt voor het ontwikkelen van cross-platofrm applicaties met één codebase.
- **TypeScript**
  Een type-safe alternatief voor javascript.
- **Capacitor**
  Native runtime voor het builden van Ionic applicaties.
- **MQTT (Via WebSockets)**
  Zorgt voor de communicatie met de hub via WebSockets.
- **Axios**
  Gebruikt voor REST communicatie met het backend.
- **Push Notifications (FCM)**
  API voor het publishen en ontvangen van push notificaties.

3.5 BACKEND

De backend is een cloudgebaseerde NestJS-applicatie, gedockeriseerd in een Docker-container en gedeployed naar Azure. Het backend functioneert als remote beheerinterface en maakt externe integraties mogelijk.

GEBRUIKTE TECHNOLOGIEËN

- **NestJS (TypeScript & Node.js)**
  Javascript framework voor het ontwikkelen scalable server sided node applicaties.
- **Docker**
  Voor containerisatie van de applicatie voor consistente ontwikkeling.
- **PostgreSQL**
  Database voor het opslaan van logs en API configs.
- **TypeORM**
  ORM voor database interacties en migraties
- **Azure & GitHub**
  Gebruikt voor CI/CD pipeline die automatisch docker images build en pusht naar ACR bij branch merges.

4.0 DOCKER

Het biometrisch toegangssysteem maakt uitgebreid gebruik van Docker voor containerisatie van de verschillende microservices en applicaties. De Hub, bestaat uit meerdere gecontaineriseerde services die via Docker Compose als één geheel worden beheerd. Elke microservice (MQTT Broker, Facial Recognition Service, Kiosk Service, MQTT Service, Cloud Service en Auth Service) draait in een geïsoleerde container met eigen dependencies, wat het mogelijk maakt om het modulair te scalen. Voor strage worden Docker volumes gebruikt, vornamelijk voor de MQTT broker configuratie en gebruikersdata. De containers communiceren via een shared netwerk, met exposed ports voor externe toegagn. Voor hardware-afhankelijke componenten zoals de Verification Node zijn speciale configuraties toegevoegd om toegang tot camera's en GPIO's mogelijk te maken. De Backend wordt verder via de CI/CD pipeline automatisch gebuild en gedeployed naar Azure Container Registry bij elke merge naar de production branch.

• **Restart Policy:** Alle containers zijn geconfigureerd met ‘resetart: unless-stopped’ voor automatische startup.

• **Netwerk:** Alle containers maken gebruik van het ‘iot-network’ voor interne communicatie

Component	Container	Image	Volumes	Exposed Ports	Dependencies
MQTT Broker	mqtt	Eclipse Mosquitto	/mosquitto/config /mosquitto/data /mosquitto/log	1883 (MQTT) 9001 (WS) 9002 (WSS)	-
Facial recognition Service	facial_recognition_service	Python 3.11-slim	/users	-	mqtt
Kiosk Service	kiosk_service	Node.js	-	3000 (HTTP)	mqtt
MQTT Service	mqtt_service	Node.js	-	-	mqtt
Cloud Service	cloud_service	Node.js	-	-	mqtt
Auth Service	auth_service	Node.js	/app/config /app/log	-	Mqtt
Verification Node	Picamera	Python 3.11-slim	/dev/shm /dev/video0 /dev/vchiq	-	-
Backend	nestjs_backend	Node.js	-	3000 (HTTP)	-

5.0 AUTHENTICATIE

5.1 INTERNAL AUTH

Voor de interne communicatie tussen microservices en applicaties binnen het netwerk wordt gebruikgemaakt van de authenticatie- en autorisatiemogelijkheden van MQTT, beheerd via de Mosquitto MQTT-broker. Elke microservice beschikt over eigen inloggegevens (username en password), die zijn vastgelegd in de passwd-config van de broker. De toegang tot topics wordt geregeld via een Access Control List (ACL), waarin per service is vastgelegd op welke topics die mag subscriben en publishen.

ACL Voorbeeld:

```
29  user lock
30  topic readwrite lock/status
31  topic write lock/open
32
```

Passwd Voorbeeld:

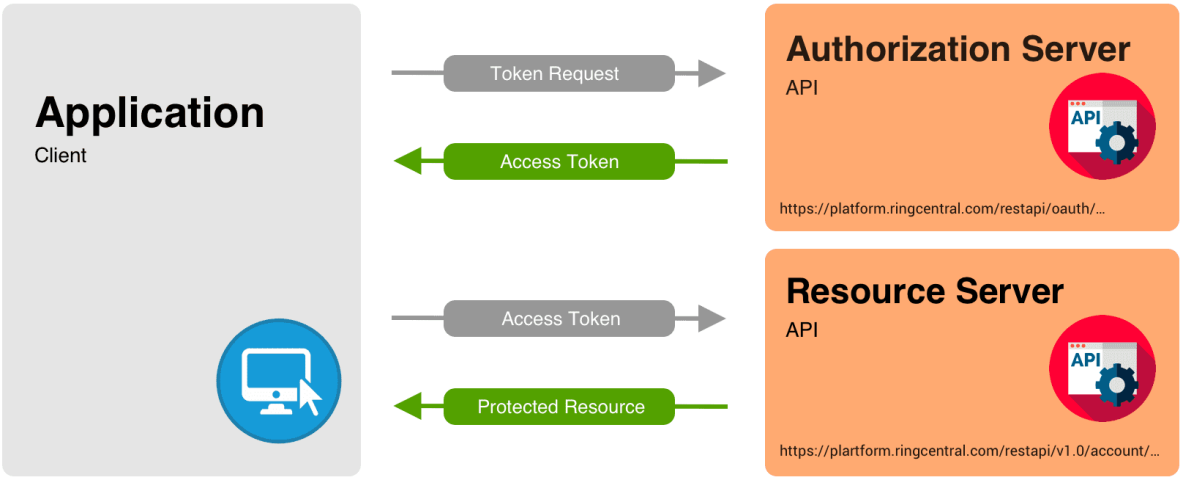
```
1  facial_recognition_service:$7$101$nzKdxrwfzedha6z9$KrOCbl/H3DjBLamogr1+GaYtqdOARfCG8Fosmf4uYknk5YPoQbEmI7qHsMFORxh
2  lock:$7$101$UNVcJ6BKTsuN/aMb$y3bdKs9U+EaBl2Npm7Mt/4gQamx/ML/Mv+BNcAngDt1d2McsCzh0Zssn8jDAvX97g7pVs+VecA2/3Qm7ilwwn
3  verification_node:$7$101$3Adxlzl9XQmhaBKP$fIIw8Bu5tk9zk4yCONPWOqvQmM2KpmBuzRVmMB93YWYUSZx4mSzaNurb27k0xxvYR2x2bdgy
4  kiosk_service:$7$101$qBW9uJ81GYVfufhs$ynxa/woN1iUYJkZMgaRRj3WicAuayhqEAbvdE8kNfLA6f8t1gJpKfNrN5eufR2fOAs9AVMAFAblg
5  debug:$7$101$TF79UN219znEJv3K$RqY9/kUDiA1g5EswcPoKLgrTKVSnCbhV0HdyTBC7ZJFubzSDz73fq8G1I1SDspCxEC2KImhxQNqsR9BZNy6G
6  mqtt_to_lora_service:$7$101$JEzbAYmNjC8+SPed$wPdHCFsPaLY/DNPY2xiVW0pF72q05MvRxPnIVg6lxDQR8w9FfOqpKbjqYCn9fk3HYZkuM
7  interface:$7$101$jYq3FN80mzaLvQga$PnO5/gd7jdW+WxN/JpYVo40Iuqxkt05pj48NUut5IGBnF1llJclkUoTJICcu1kqDbWfzRHZ0BGXKZ/ea
8  mqtt_service:$7$101$N4uG6eYyd5xVBA0b$qUoIU7O2yvn9H1qRugiYsoq4puWmq4dsfM5Ia76SSQlsEeAkKdQYDq3weelBQK+kYf4sQBQ/1zmMd
```

5.2 REMOTE AUTH

Voor de gebruikersauthentiatie buiten het MQTT netwerk, zal er gebruik worden gemaakt van JWT auth (@nestjs/jwt).

1. Gebruiker logt in via de mobiele app met zijn inloggegevens.
2. De backend geeft een JWT terug.
3. Alle volgende api requests van de app bevatten dit token.

De backend valideert het token, en voegt gebruikers data toe aan de api request pipeline.



6.0 FAILSAFE

Het slotmechanisme wordt aan de binnenkant van de deur geplaatst, waardoor het mogelijk blijft om het slot van buitenaf te openen. Daarnaast kunnen er technieken worden toegepast zoals een watchdog timer, die het systeem automatisch herstartals die vastloopt, en een health status overview, waarmee direct ingegrepen kan worden als er iets uitvalt.

7.0 OVERIGE OVERWEGINGEN

- **Webcamfeed niet via MQTT:** Alternatieve oplossingen onderzoeken om webcamfeed niet via MQTT te versturen om overbelasting van de broker te voorkomen.
- **Geen samenvoeging van Lock & Verification Node mogelijk:** Vanwege functionele en architecturale redenen is ervoor gekozen deze componenten toch gescheiden te houden. Dit heeft er mee ta maken dat de verification node niet perse in de buurt van het slot hoeft te hangen.
- **Webcamfeed framerate beperken:** Een framerate van 15–20 FPS (of lager) zou voldoende moeten zijn voor een live preview.
- **FFMPEG Servier:** Gebruik maken van een FFMPEG server zou mogelijk een betere oplossing zijn voor de webcamfeed.