




# IoT Sensör Takip Sistemi (NestJS + MQTT + Next.js + Socket.io + Supabase)

Bu proje, **fabrikalardaki IoT sensörlerinden gelen verileri MQTT protokolü üzerinden toplayan**, bu verileri **gerçek zamanlı yayınlayan**, **kullanıcı ve şirket yönetimi sağlayan**, **loglama ve güvenlik mekanizmaları içeren** kapsamlı bir sistemdir.

**Bu proje Görkem Kurtkaya tarafından geliştirilmiştir.**

## Canlı Linkler

Servis	URL
 Backend (NestJS)	<a href="https://iot-backend-559293271562.europe-west1.run.app">https://iot-backend-559293271562.europe-west1.run.app</a>
 Frontend (Next.js)	<a href="https://iot-frontend-559293271562.europe-west1.run.app">https://iot-frontend-559293271562.europe-west1.run.app</a>
 Test Microservice	<a href="https://testmicroservice-hydust6b3a-ew.a.run.app">https://testmicroservice-hydust6b3a-ew.a.run.app</a>

## Proje Amacı

- Fabrikalardaki **IoT sensörlerinden gelen sıcaklık ve nem verilerini** toplamak
- Bu verileri **gerçek zamanlı olarak yayınlamak**
- Kullanıcıların yalnızca yetkili oldukları cihaz verilerine erişmelerini sağlamak
- Loglama ve davranış takibi** ile kullanıcı aktivitelerini izlemek
- EMQX MQTT Broker** üzerinden güvenli veri alışverişi sağlamak

## Teknoloji Stack

### Backend

- Framework:** NestJS (Node.js)
- Veri Tabanı:** Supabase (PostgreSQL)
- MQTT Broker:** EMQX Cloud (mqttps:// ile TLS)
- Gerçek Zamanlı Veri Yayını:** MQTT + Socket.io
- Güvenlik:** JWT Authentication
- Logging:** Structured JSON Logging
- Containerization:** Docker + Google Cloud Run
- Test:** MQTT simülasyon servisi

### Frontend

- Next.js**
- WebSocket üzerinden anlık veri güncelleme**



## Projeyi Başlatma



### Docker Kullanımı

- Projeyi klonladıktan sonra sadece tek komutla tüm servisleri ayağa kaldırabilirsiniz:

```
docker-compose up --build
```

Bu komut:

- Backend (NestJS) (iot-backend)
- Frontend (Next.js) (iot-frontend)
- Test Microservice (testmicroservice)

üçünü de aynı anda ayağa kaldırır.



### Lokal Geliştirme

- Tüm servisleri ayrı ayrı localde çalıştırmak isterseniz:



### Backend (NestJS)

```
cd backend
npm install
npm run start
```



### backend/.env

```
# Supabase bağlantı bilgileri
SUPABASE_URL=https://your-supabase-url.supabase.co
SUPABASE_KEY=your-supabase-service-key
# MQTT Broker bilgileri (TLS yoksa mqtt://, TLS varsa mqtt://)
MQTT_BROKER_URL=mqtt://your-broker.emqxcloud.com
MQTT_USERNAME=mqtt
MQTT_PASSWORD=your-password
# JWT Secret
JWT_SECRET=your_jwt_secret
# WebSocket CORS izinli frontend URL
FRONTEND_URL=https://your-frontend.run.app
```



### Frontend (Next.js)

```
cd frontend
npm install
npm run dev
```

#### frontend/.env.local

```
# Backend API adresi
NEXT_PUBLIC_BACKEND_URL=https://your-backend.run.app
```

#### Test Microservice

```
cd testmicroservice
npm install
node app.js
```

#### testmicroservice/.env

```
# EMQX MQTT bağlantı bilgileri (TLS ile)
MQTT_BROKER_URL=mqtts://your-broker.emqxcloud.com:8883
MQTT_USERNAME=mqtt
MQTT_PASSWORD=your-password
```

### MQTT Broker'ı Localde Çalıştırmak İstiyorsan

-Projeyi localde test etmek ve kendi broker'ını çalıştırmak istiyorsan Eclipse Mosquitto kullanarak local MQTT broker kurabilirsin.

```
docker run -it -p 1883:1883 -p 9001:9001 eclipse-mosquitto
```

- 1883: MQTT bağlantısı (mqtt://)
- 9001: WebSocket bağlantısı (ws://)

### 2. .env Ayarında MQTT Broker'ı Local Olarak Ayarla

```
MQTT_BROKER_URL=mqtt://localhost:1883
MQTT_USERNAME= # Gerekliyse bırak
MQTT_PASSWORD= # Gerekliyse bırak
```

## Sistem Mimarisi

### Kullanıcı Roller

- **System Admin:** Şirket, kullanıcı, cihaz ekleyebilir. Tüm loglara ve verilere erişebilir.
- **Company Admin:** Kendi şirketi için kullanıcı ve cihaz atamaları yapabilir, verileri ve logları görüntüleyebilir.
- **User:** Sadece yetkili olduğu cihazlardan gelen sensör verilerini görüntüleyebilir.

### Sensör Verisi Akışı

1. Sensörler şu formatta veri gönderir:

```
{
  "sensor_id": "temp_sensor_01",
  "timestamp": 1710772800,
  "temperature": 25.4,
  "humidity": 55.2
}
```

2. Backend, EMQX MQTT Broker'a abone olur.
3. Gelen veri doğrulanır → hatalıysa loglanır.
4. Doğru veri veritabanına yazılır.
5. Socket.io üzerinden ilgili kullanıcıya gerçek zamanlı iletilir.

### Güvenlik Katmanları

- Tüm API'ler JWT ile korunur.
- MQTT bağlantısı TLS (mqtt://) ile güvenli hale getirilmiştir.
- Structured JSON loglama dosya sistemine yapılır.
- Kullanıcıların ve Şirketlerin sadece yetkili olduğu cihazlara erişimi vardır.

### Loglama & Kullanıcı Takibi

- Her kullanıcı log sayfasını her görüntülediğinde şu formatta kayıt oluşturulur:

```
{
  "user_id": "user_123",
  "timestamp": 1710772800,
  "action": "viewed_logs"
}
```

- Bu kayıtlar üzerinden kullanıcıların hangi saatlerde aktif olduğu analiz edilebilir.

## Test Microservice

Gerçek sensörlere gerek kalmadan test amacıyla hazırlanmış bir mikroservistir. İsterseniz anlık olarak veya her 15 saniyede bir rastgele sensör verisi üretir ve EMQX MQTT Broker'a yayımlar.

Canlı test etmek için: <https://testmicroservice-hydust6b3a-ew.a.run.app>

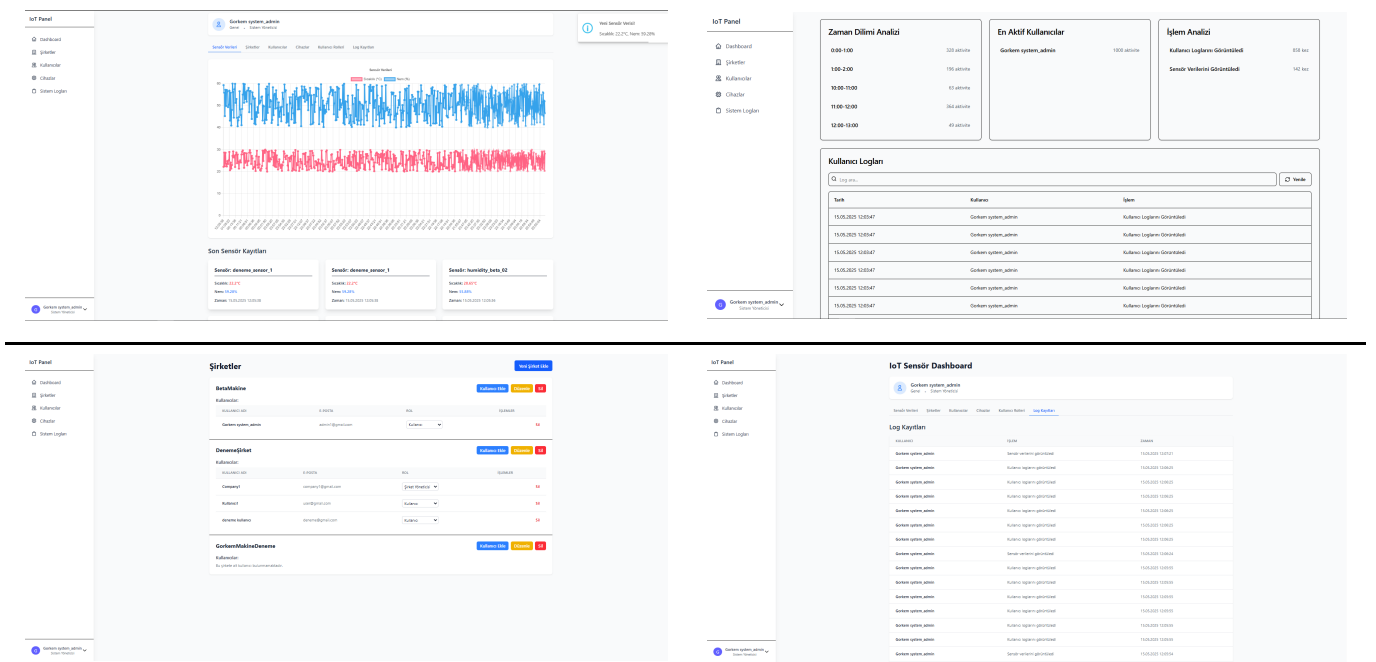
## Deployment

Tüm servisler Google Cloud Run üzerinde deploy edildi. Cloud NAT + VPC Connector kullanılarak TCP çıkış (MQTT 8883) sağlandı.

## Uygulama Görselleri

### Frontend Arayüzü

### Sistem Logları



## Testmicroservice

## MQTT Sensör Test Paneli

### Manuel Veri Gönderme

Sensör Verisi Gönder

✓ Başarılı: Veri gönderildi

### Otomatik Veri Gönderme

Otomatik Göndermeyi Başlat

Otomatik Göndermeyi Durdur

Durum: Otomatik gönderme kapalı

### Son Gönderilen Veri

```
{  "sensor_id": "deneme_sensor_1",  "temperature": 22.2,  "humidity": 99.98,  "timestamp": "2025-05-15T12:05:38+03:00"}
```