ANKARA ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BLM3522

BULUT BİLİŞİM VE UYGULAMALARI

RAPORU

Proje 2

Ömer Asaf DEMİR

20290240

GitHub: AsafDemir/BulutBilisimVeUygulamalariBut

https://github.com/AsafDemir/BulutBilisimVeUygulamalariBut

Video:

https://drive.google.com/file/d/1qo1s6Zy9G3imAkj8BKh76dGcHcPRea7-/view?usp=sharing

1. Proje Genel Bakış

IoT Sensör Verisi İşleme Sistemi; sıcaklık ve nem gibi çevresel verileri gerçek zamanlı toplayan, işleyen ve görselleştiren modern bir bulut tabanlı IoT uygulamasıdır. MQTT protokolü ile sensörden gelen veriler Google Cloud Pub/Sub üzerinden Flask uygulamasına yönlendirilir ve MongoDB Atlas veritabanında saklanır. Sistem, ölçeklenebilirlik, güvenlik ve esneklik gibi bulut bilişim ilkelerini temel alarak tasarlanmıştır. Web tabanlı arayüz ile kullanıcılar verileri canlı olarak izleyebilir ve analiz edebilir.

2. Kullanılan Teknolojiler ve Uygulamalar

Backend ve IoT Teknolojileri

- Python 3
- Flask Framework
- Sensor Simulator (sensor_simulator.py)
- Data Consumer (data_consumer.py)

Python dili, sistemin tüm backend altyapısının geliştirilmesinde kullanılmıştır. Flask framework'ü ile RESTful API'ler tasarlanmış ve kullanıcıların veri erişimi sağlanmıştır. Sensor Simulator bileşeni ile gerçekçi sıcaklık ve nem verileri üretilmiş, Data Consumer modülü ise bu verileri alıp MongoDB'ye kaydetmiştir.

Mesajlaşma Katmanı

- MQTT Protokolü (Paho-MQTT Library)
- Eclipse Mosquitto (Lokal Broker)
- HiveMQ Public Broker (Test Ortamı)
- Google Cloud Pub/Sub

MQTT protokolü, IoT cihazları arasında hafif ve hızlı veri iletimi sağlamak amacıyla tercih edilmiştir. Paho-MQTT kütüphanesi ile Python ortamında MQTT işlemleri yürütülmüştür. Lokal testlerde Mosquitto kullanılmış, Pub/Sub ise üretim ortamında asenkron mesajlaşma ve servisler arası veri akışı için kullanılmıştır. Pub/Sub topic ve subscription yapıları, sensör verilerinin Flask sunucusuna HTTP push yöntemiyle iletilmesini sağlamıştır.

Veritabanı Teknolojileri

- MongoDB Atlas (Cloud NoSQL Database)
- MongoDB Compass

MongoDB Atlas, yapılandırılmamış sensör verilerinin JSON formatında depolanması için kullanılmıştır. NoSQL yapısı sayesinde sistem yatayda ölçeklenebilir bir veritabanına sahip olmuştur. MongoDB Compass aracı, verilerin okunabilirliğini artırmış ve canlı kontrol süreçlerinde kullanılmıştır.

Bulut Servisleri ve Deployment

- Google Cloud Run
- Google IAM & Service Account
- Google Cloud Container Registry
- Docker (Multi-stage Dockerfile)
- .env Ortam Dosyası

Flask uygulaması Docker ile container haline getirilmiş ve Google Cloud Run üzerinde çalıştırılmıştır. Cloud Run'ın otomatik ölçeklenebilir ve serverless yapısı sayesinde maliyet ve kaynak kullanımı optimize edilmiştir. GCP kimlik doğrulama için IAM ve service account yapıları yapılandırılmış, JSON kimlik dosyaları ortam değişkeni olarak .env içinde yönetilmiştir.

Frontend Teknolojileri

- Vanilla JavaScript
- CSS Grid Layout
- Font Awesome
- RESTful API ile Gerçek Zamanlı Veri Görselleştirme

Web arayüzünde kullanıcıya görsel olarak sade ama işlevsel bir panel sunulmuştur. Vanilla JS ile yazılmış dinamik içerik yapısı 30 saniyelik otomatik veri yenileme ile canlı güncellenmekte, CSS Grid ile responsive yapı sağlanmaktadır. İkon desteği için Font Awesome kullanılmıştır.

Güvenlik ve İzleme

- Google IAM ve Rol Bazlı Erişim
- Environment Variable İzolasyonu
- MongoDB Encryption at Rest / In Transit
- Google Cloud Monitoring & Logging

Uygulamada kimlik bilgilerinin korunması için environment dosyaları kullanılmış ve erişim kontrolü IAM aracılığıyla sağlanmıştır. MongoDB tarafında hem at rest hem de in transit şifreleme aktif hale getirilmiştir. Google Cloud Operations ile sistemin gecikme süresi, hata oranları ve mesaj throughput gibi metrikleri izlenmiştir.

3. Sistem Mimarisi

Kullanılan Yapılar ve Bileşenler

- MQTT + Google Pub/Sub hibrit veri akışı
- Mosquitto MQTT Broker
- Flask Web Servisi (Cloud Run)
- MongoDB Atlas Veritabanı
- REST API Endpoint'leri
- Web Tabanlı Arayüz (JS + API)

Sensörden gelen veriler önce MQTT ile Mosquitto broker'a iletilir. Bu veriler ya doğrudan veri tüketici modülü ile MongoDB'ye yazılır ya da Google Pub/Sub ile Cloud Run'daki Flask API'ye gönderilir. Flask uygulaması gelen veriyi işler ve MongoDB'ye kaydeder. Web arayüzü, bu verileri sorgulayarak kullanıcılara canlı olarak sunar.

4. Güvenlik ve Hata Toleransı

Uygulanan Önlemler

- MQTT TLS veri şifreleme
- MongoDB şifreli veri aktarımı ve saklama
- Google IAM ile rol tabanlı erişim
- .env ile gizli bilgilerin korunması
- Cloud Run health check endpoint
- MQTT QoS ve yeniden bağlanma desteği

- GCP log takibi

MQTT ile veri iletiminde TLS kullanılarak güvenlik sağlanmıştır. MongoDB, verileri hem aktarımda hem saklamada şifreli tutar. IAM sayesinde yalnızca yetkili servisler erişim sağlar. Kimlik bilgilerinin sızmaması için .env dosyaları kullanılmıştır. Cloud Run, sistemin çalışır durumda olup olmadığını otomatik kontrol eder. MQTT'de QoS 1 ile veri kaybı engellenmiş, bağlantı kesilirse yeniden bağlanma sağlanmıştır.

5. Performans ve Ölçeklenebilirlik

Kullanılan Yöntemler

- Cloud Run autoscaling (0–10 instance)
- MongoDB indexing (timestamp, sensor_id)
- Query optimizasyonu
- GCP Monitoring ile izleme
- 1000 msg/s MQTT veri akışı
- <150ms dashboard yanıt süresi

Uygulama trafik yoğunluğuna göre otomatik olarak yeni instance başlatır. MongoDB'de yapılan indekslemeler sorgu hızını artırır. Cloud Monitoring ile sistem performansı düzenli takip edilir. Testlerde saniyede 1000 mesaj işlenmiş, arayüzde 150ms altında cevap süreleri elde edilmiştir.

6. Test ve Kalite Güvencesi

Uygulanan Testler

- Sensor Simulator → MQTT → MongoDB testi
- Pub/Sub → Flask → Veritabanı girişi testi
- Dashboard → API → Arayüz güncelleme testi
- Bağlantı kesintisi ve veri hatası senaryoları
- Yük altında sistem kararlılığı testi

Sistem uçtan uca testlerle doğrulanmıştır. Sensörden MongoDB'ye kadar olan veri akışı ve arayüzdeki veri güncellemeleri test edilmiştir. Hatalı veri girişleri, bağlantı sorunları ve yoğun trafik altında sistemin dayanıklılığı gözlemlenmiş ve güvenilirliği sağlanmıştır.

7. Proje Sonuçları

Elde Edilen Kazanımlar

- Gerçek zamanlı veri işleme başarıyla sağlandı
- Otomatik ölçeklenebilir mimari oluşturuldu
- MQTT ve Pub/Sub hibrit yapısı uygulandı
- IoT sistemlerinin temel ihtiyaçları karşılandı

Proje hedeflerine ulaşarak; veri akışının güvenli, hızlı ve ölçeklenebilir bir şekilde yönetilmesi sağlanmıştır. Hem geleneksel hem de bulut-native yöntemler desteklenerek esnek bir yapı kurulmuştur. Modern IoT uygulamaları için güçlü bir temel sunulmuştur.