TURKISH TECHNOLOGY

SENSOR DATA INTERPRETER PROJESİ

KURULUM DOKÜMANI

18.12.2023

**İÇİNDEKİLER**

[**1.** **PROJE GENEL BİLGİSİ** 3](#_Toc153883088)

[**3.** **KURULUM ADIMLARI** 3](#_Toc153883089)

[**3.1.** **DOCKER DESKTOP KURULUMU** 3](#_Toc153883090)

[**3.2.** **POSTGRESQL, KAFKA VE MONGODB KURULUM ADIMLARI** 4](#_Toc153883091)

[**3.3.** **UYGULAMANIN ÇALIŞTIRILMASI** 7](#_Toc153883092)

[**3.4.** **PROJE TASARIM KARARLARI** 7](#_Toc153883093)

**ŞEKİLLER LİSTESİ**

[**Şekil 1:** Windows Özellikleri Aç veya Kapat Ekranı 4](#_Toc153883313)

[**Şekil 2:** Uyguların Kurulumu Sonrası Docker Desktop Containers Ekranı 5](#_Toc153883314)

[**Şekil 3:** MongoDB Compass New Connection Ekranı 5](#_Toc153883315)

[**Şekil 4:** MongoDB Compass Database Listesi Ekranı 6](#_Toc153883316)

[**Şekil 5:** MongoDB Compass New Database Ekranı 6](#_Toc153883317)

[**Şekil 6:** MongoDB Compass Import Data Ekranı 7](#_Toc153883318)

**TABLO LİSTESİ**

[**Tablo 1:** Sistem Özellikleri 3](#_Toc153883087)

1. **PROJE GENEL BİLGİSİ**

Proje kapsamında, periyodik olarak farklı miktarda sensor datası rastgele değerler ile üretilip mesajlaşma kuyruğuna eklenecektir. Mesajlaşma kuyruğundan okunan sensor datasından istatistiksel veriler elde edilip veritabanına gerçeğe yakın hızda kayıt edilecektir. Aynı sensor datasından operasyonel veriler de elde edilecek, bu verilerin bir kısmı kullanılarak başka bir veritabanından sorgu yapılacak, sorgu sonucu elde edilen değer ile matematiksel işlem yapılacak ve işlem sonucu ile operasyonel veri veritabanına kayıt edilecektir. Ayrıca, device\_id ve zaman aralığını parametre olarak alan bir webservisi, istatistiksel verilerden konum geçmişini dönecektir.

1. **SİSTEM ÖZELLİKLERİ**

Kurulumlar, geliştirmeler ve testler aşağıdaki sistem özelliklerine sahip bilgisayarda gerçekleştirilmiştir:

**Tablo 1:** Sistem Özellikleri

|  |  |
| --- | --- |
| **İşletim Sistemi** | Windows 11 Pro 64 Bit |
| **İşlemci** | Intel i7-1065G7 CPU 1.50 GHz |
| **Ram** | 32 GB |
| **IDE** | Intellij Idea 2023.2.1 (Community Edition) |
| **PostgreSQL Browser** | PgAdmin 4 version 7.5 |
| **MongoDB Browser** | MongoDB Compass version 1.40.4 |

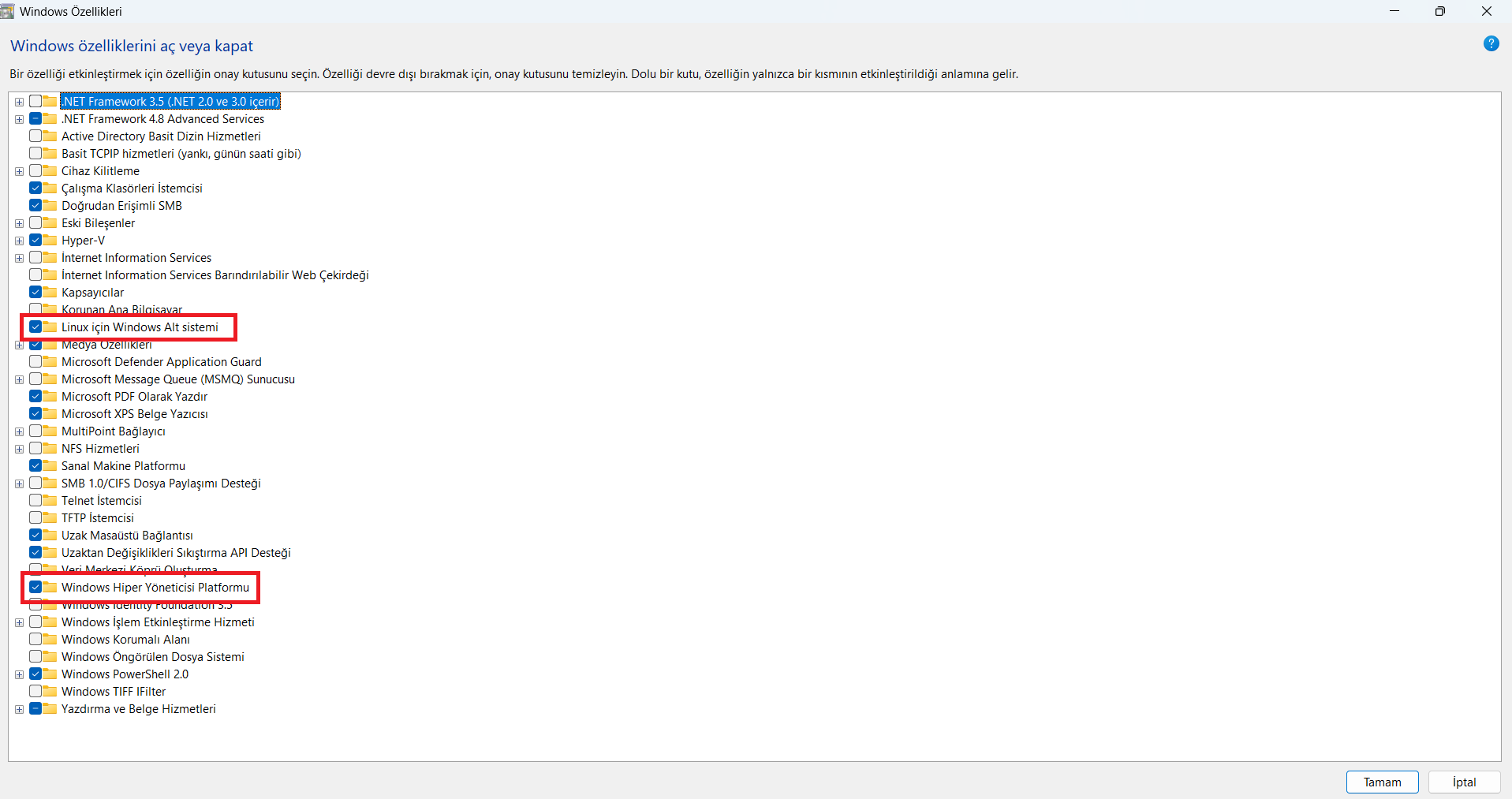
1. **KURULUM ADIMLARI**
   1. **DOCKER DESKTOP KURULUMU**

1- <https://docs.docker.com/desktop/install/windows-install/> adresinden Docker Desktop Installer.exe indirilir.

2- Denetim Masası altındaki Windows Özelliklerini Aç veya Kapat Ekranı açılır ve aşağıdaki seçenekler seçilir:

- Linux için Windows Alt sistemi

- Windows Hiper Yönetici Platformu

****

**Şekil 1:** Windows Özellikleri Aç veya Kapat Ekranı

3- Docker Desktop Installer.exe ile Docker Desktop kurulumu yapılır.

* 1. **POSTGRESQL, KAFKA VE MONGODB KURULUM ADIMLARI**

1- .zip dosyası açılır.

2- Docker Compose Files konumda Command Line veya Windows PowerShell açılır.

3- Açılan ekrana aşağıdaki komut yazılır ve enter tuşuna basılır; Kafka Mesaj Kuyruğu’nun kurulduğu görülür.

**docker-compose -f docker-compose-kafka.yml up -d**

4- Kafka Mesaj Kuyruğu’nun kurulumu tamamlandıktan sonra aşağıdaki komut çalıştırılarak PostgreSQL veitabanı kurulur.

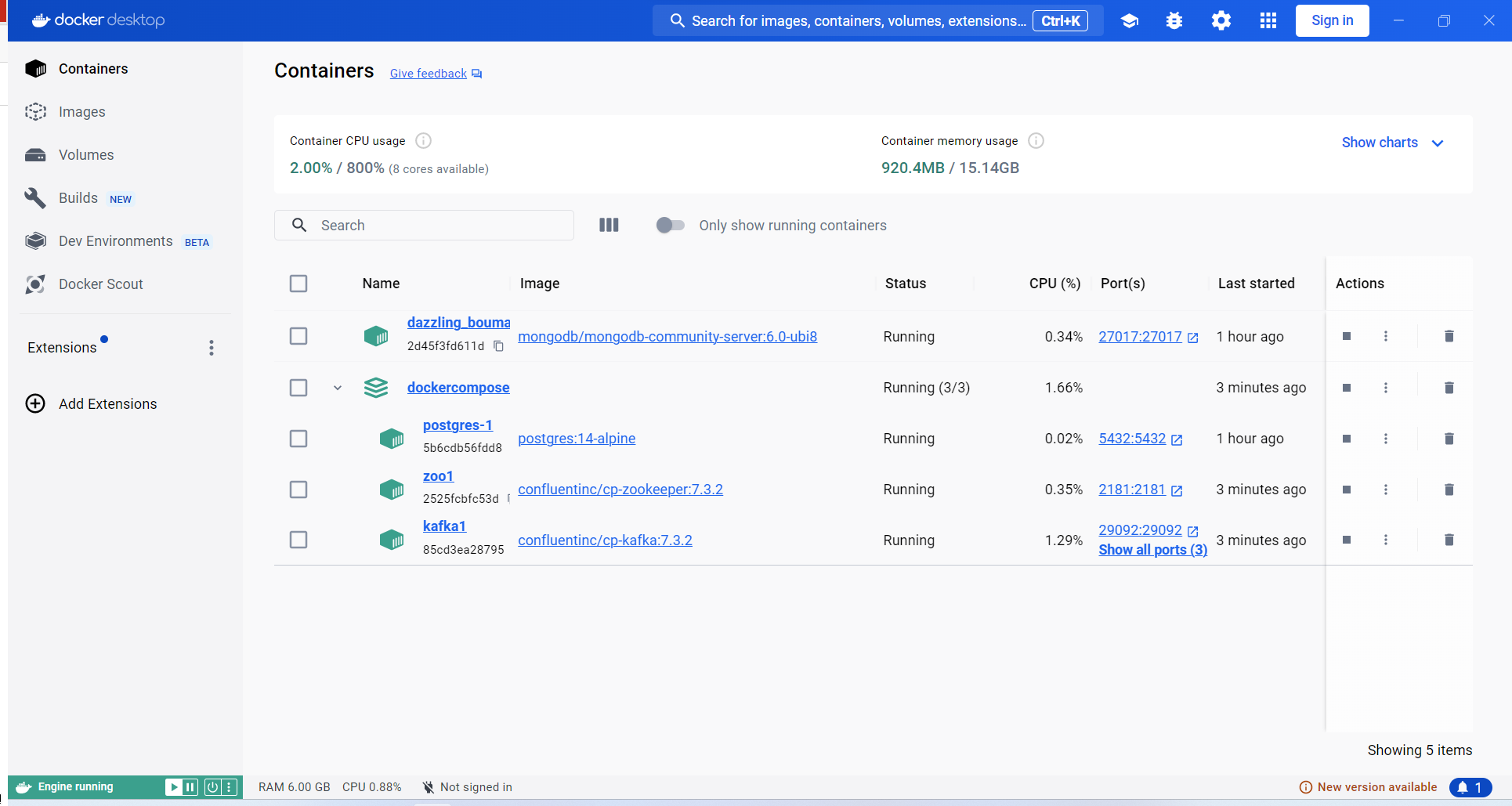
**docker-compose -f docker-compose-postgresql.yml up -d**

5- PostgreSQL kurulumu tamamlandıktan sonra aşağıdaki komut çalıştırılarak MongoDB veritabanı kurulur.

**docker-compose -f docker-compose-mongodb.yml up -d**

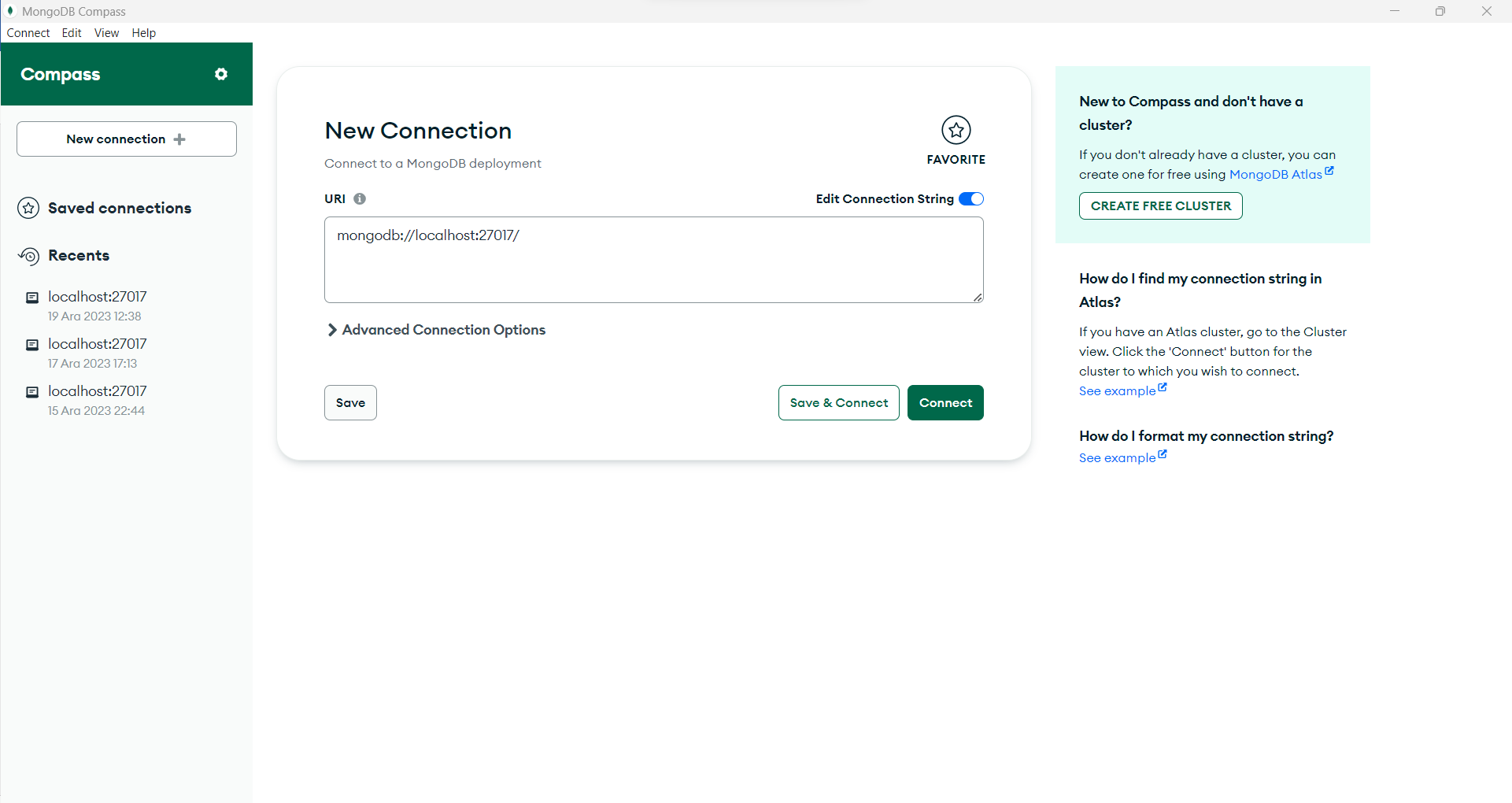
6- <https://www.pgadmin.org/download/pgadmin-4-windows/> sayfasından PgAdmin 4 uygulamasının son sürümü indirilip kurulur.

7- <https://www.mongodb.com/try/download/compass> sayfasından MongoDB Compass uygulamasının son sürümü indirilip kurulur.



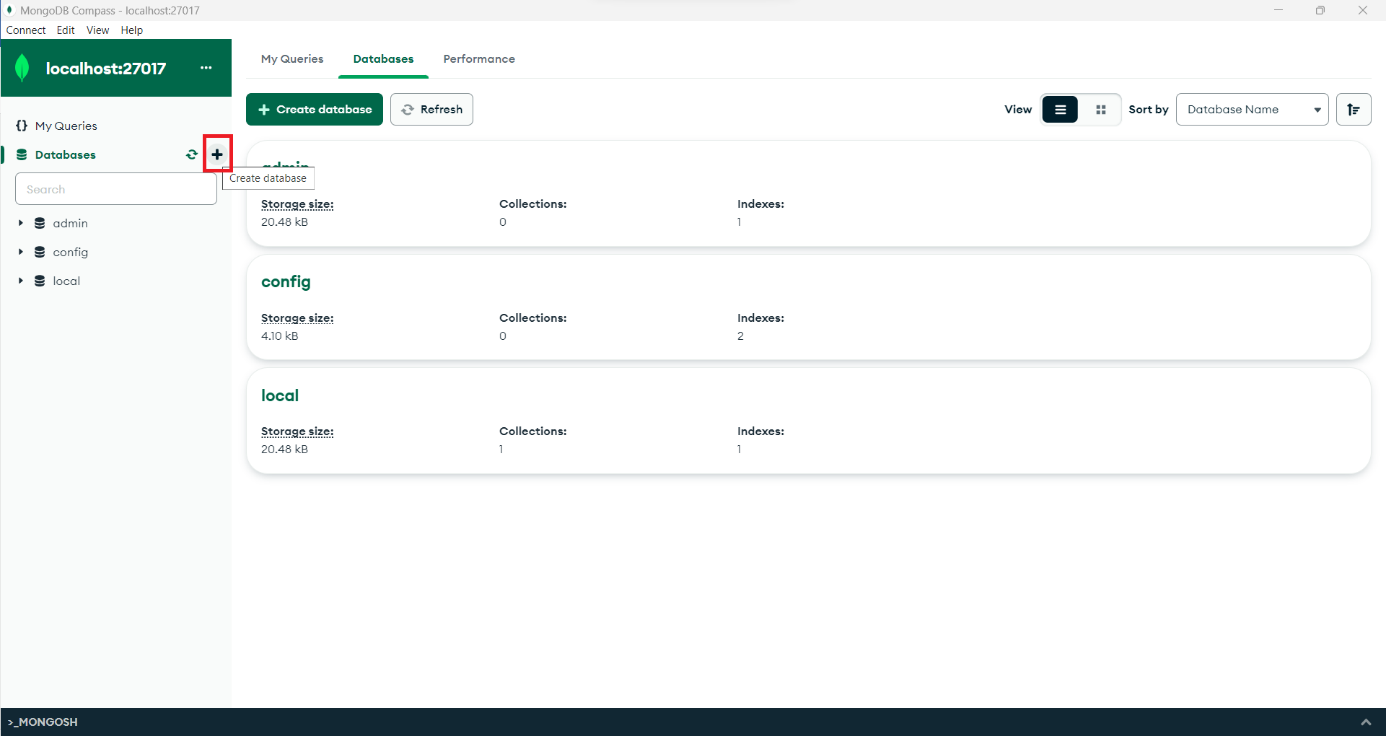
**Şekil 2:** Uyguların Kurulumu Sonrası Docker Desktop Containers Ekranı

8 – MongoDB Compass uygulaması açılır ve Connect butonuna basılarak MongoDB’ye bağlanılır.



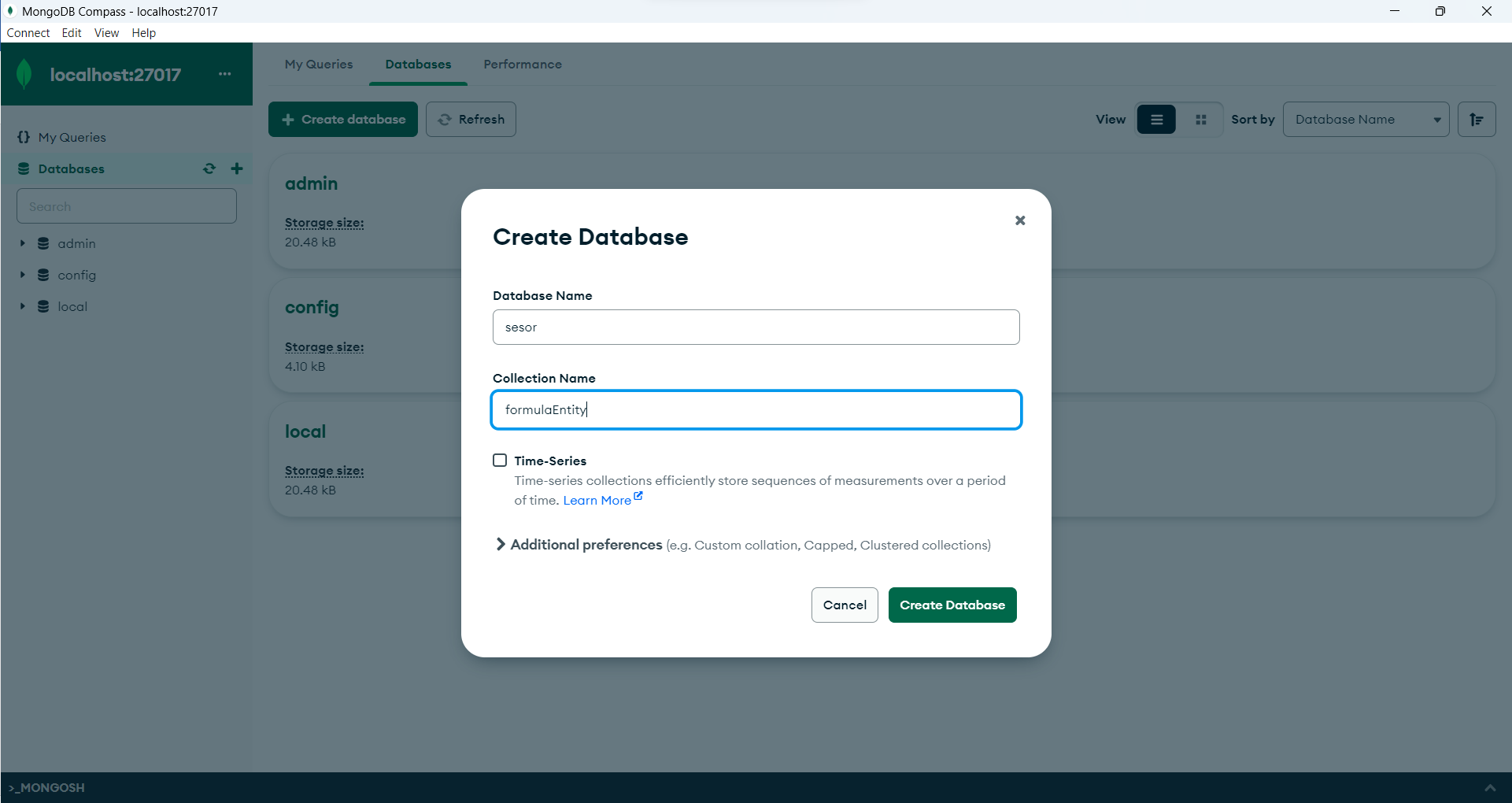
**Şekil 3:** MongoDB Compass New Connection Ekranı

9 – Create Database butonuna basılır.



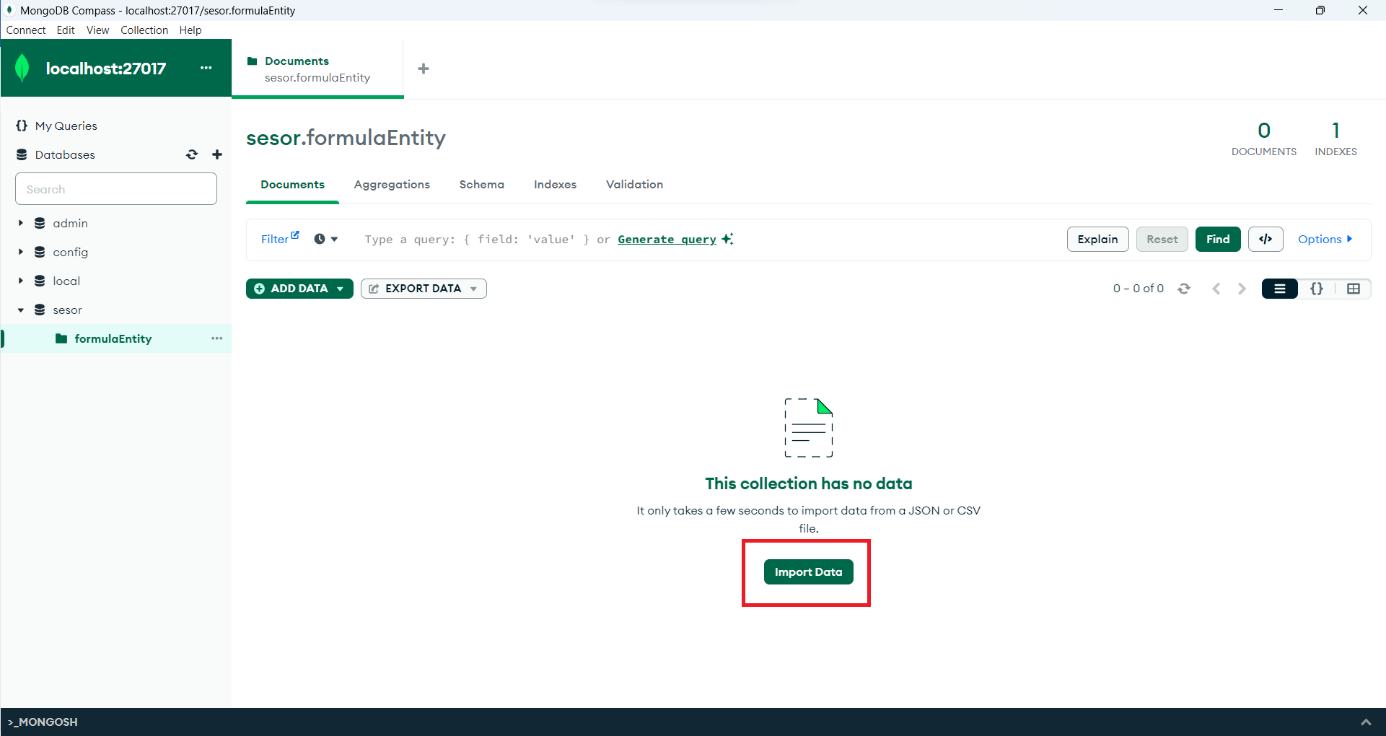
**Şekil 4:** MongoDB Compass Database Listesi Ekranı

10 – Database Name olarak sensor ve Collection Name olarak formulaEntity girilir ve Create Database butonuna basılır.



**Şekil 5:** MongoDB Compass New Database Ekranı

11- Import Data butonuna basılarak sensor.formulaEntity.json dosyası import edilir.



**Şekil 6:** MongoDB Compass Import Data Ekranı

* 1. **UYGULAMANIN ÇALIŞTIRILMASI**

1- Docker Desktop Uygulaması açılır ve Containers Ekranı’ndaki Actions sütunu altındaki Run butonlarına basılarak uygulamaların çalışması sağlanır.

2- .zip klasörü altındaki Çalışan Uygulamalar Klasörü açılır.

3- sensorDataInterpreter.bat dosyası mouse çift tık ile çalıştırılır.

4- sensorDataGeneratorSimulator.bat dosyası mouse çift tık ile çalıştırılır.

**NOT:** Uygulamaların kaynak kodları sensorDataGenerator ve sensorDataInterpreter klasörleri altındadır.

* 1. **PROJE TASARIM KARARLARI**

Sensor Data Interpreter Uygulaması’nın test edilebilmesi için ayarlanabilir periyodik sürede ayarlanabilir miktarda rastgele değerler ile sensor datası üreten simülatör uygulamasına ihtiyaç vardır. Bu uygulamanın geliştirilmesi için Java 17 sürümü, Spring Boot 3.2.0 sürümü, Maven ve gerekli kütüphane bağlılıkları seçilerek <https://start.spring.io/> sitesi üzerinden proje oluşturulmuştur.

Rastgele üretilecek değerlerin daha homojenize üretilebilmesi için Random sınıfı yerine SecureRandom sınıfını kullandım.

Periyodik işler için Spring Boot’un @Scheduled annotation’ını kullanarak periyodik işlerin yönetimini Spring Boot’a bıraktım.

Aspect Oriented Programming ile sınıf bağlılıkları arasında loose coupling kurup aynı zamanda obje oluşturulmasını ve yönetimini Spring Boot’a bıraktım.

Mesajlaşma kuyruğu olarak Apache Kafka seçtim.

Kafka üzerinden mesaj gönderirken direk objenin kendisini serialize edip göndermek yerine Sensor objesini String’e çevirip gönderdim. Objeyi String’e çevirip, kuyruktan mesajı okuduktan sonra String’i tekrar objeye çevirmek maliyet oluştursa da farklı objeler için özel Serializer ve KafkaTemplate gibi özel Config’ler oluşturmak yerine genel bir yapı kurarak SOLID prensiplerine uyum sağlamış oldum.

Sensor objesi içerisindeki benzer tipteki alanların değerlerinin üretimini ortaklaştırıp Util sınıfına alarak hem OOP hem de SOLID prensiplerine uyum sağlamış oldum.

JUnit testleri yazarak rastgele değerleri doğru ürettiğimi test ettim.

Application.yml dosyasındaki random-sensor-data-generator-fixed-delay-in-ms parametresinin değeri değiştirilerek sensor datası üretim periyodu ayarlanabilir.

Application.yml dosyasındaki random-generated-sensor-data-max-number parametresinin değeri değiştirilerek 1 periyodda üretilecek max. sensor datası miktarı ayarlanabilir. Max. sensor sayısı hesaplama formülü:

Max. sensor datası = random-generated-sensor-data-max-number \* 100 + 100. Örneğin random-generated-sensor-data-max-number 50 girilirse her periyodda 100 ile 5100 arasında rastgele bir sayıda sensor datası üretilir ve mesaj kuyruğuna gönderilir.

Her periyodda gönderilen sensor datası ile uygulama çalışmaya başladığı andan itibaren toplam mesaj kuyruğuna gönderilen sensor datası console’a log olarak yazdırılır.

Mesaj kuyruğunu dinleyip istatistiksel ve operasyonel verileri yorumlayıp veritabanına kayıt eden, veritabanındaki istatistiksel verileri webservisi ile dönen Sensor Data Interpreter uygulaması oluşturmak için Java 17 sürümü, Spring Boot 3.2.0 sürümü, Maven ve gerekli kütüphane bağlılıkları seçilerek <https://start.spring.io/> sitesi üzerinden proje oluşturulmuştur.

Mesaj kuyruğundaki yük dalgalanmalarını karşılayabilecek ve gerçek zamana yakın hızda verileri yorumlayıp veritabanına kayıt etmek için Kafka listener methoduna concurrency parametresi eklenerek (Application.yml dosyasında spring.kafka.concurrency-number parametresi ile değeri ayarlanabilir) aynı anda birden fazla thread ile kuyruktan veri okunması sağlandı. Yorumlanan istatistiksel ve operasyonel verilerin single insert ile PostgreSQL veritabanına kayıt hızının kuyruğa veri eklenme hızına oranla çok yavaş kalması sebebiyle batch insert işlemi gerçekleştirildi. List of CopyOnWriteArrayList tutuldu. Kuyruktan okunan veriler CopyOnWriteArrayList’e eklendi. Her liste 500 adet veriye ulaşınca List<CopyOnWriteArrayList>’e size’ı 500 olan CopyOnWriteArrayList eklendi ve yeni bir CopyOnWriteArrayList oluşturularak yeni gelen verilerin bu listeye eklenmesi sağlandı. 1 saniyelik periyodda çalışan scheduled job ile CopyOnWriteArrayList’deki verilerin batch insert ile veritabanına kayıt edilmesi sağlandı.

Thread-safe çalışmayı sağlamak ve veri kaybını önlemek için Arraylist yerine CopyOnWriteArrayList kullanıldı.

Veritabanına kayıt sırasında hata oluşması durumunda (PostgreSQL’de aynı anda max. connection sayısına ulaşılırsa veya veritabanına erişim koparsa) veri kaybı olmaması için MongoDB’ye ve dosyaya mesajlar kayıt edildi.

Saniyede 2100 mesajın üretilip mesaj kuyruğuna eklenme hızına geliştirmeleri yaptığım bilgisayarımın donanımları yükü kaldırıp gerçek zamana yakın hızda çalıştı. Daha fazla miktarda üretilen dataların hızına yetişmek için PostgreSQL yerine MongoDB’ye verileri kayıt etmeyi denedim ama performansları yakın oldu. Daha fazla miktarda üretilen dataların yorumlanmasının gerçek zamana yakın olabilmesi için CPU’su daha güçlü ve çekirdek sayısı daha fazla olan bir sunucuya veya birden fazla sunucuda cluster olarak çalışan Sensor Data Interpreter uygulamalarına ihtiyaç vardır.

Intellij Idea 2023.2.1 (Community Edition) içerisinde gelen code review rule’ları ile kod gözden geçirmesi ve düzenlemesi yaptım.