Görsel Çözümleme
Vaka Çalışması

11. 5. 2026

Vaka Çalışması

Orman yangınlarının atmosfere doğrudan etkisi, dumanın yangın bölgesinden uzaklaşmasıyla hava kalitesinin düşmesidir. Duman kirliliğinden oluşacak zararlı sonuçlarla canlıların sağlığı üzerinde kısa ve uzun vadeli olumsuz etkiler görülecektir. CAMS (Copernicus Atmosfer İzleme Servisi), atmosfere taşınan emisyonları ve bileşimlerini izleyerek dumanın hangi yöne doğru hareket edeceğini öngörür. Atmosfer kirliliğinin en yüksek seviyeleri yangına yakın bölgelerde meydana gelirken, bazı duman bileşenlerinin uzun bir fotokimyasal ömrü vardır; bu da duman kirliliğinin uzun mesafeli yayılımlara (kıtalararası ölçeklerde) neden olabileceği ve yangın bölgesinden binlerce kilometre uzaktaki hava kalitesini potansiyel olarak etkileyebileceği anlamına gelir.

Orman Yangını Duman İzleme ve Hava Kalitesi Tahmin Sistemi

- Sistemin amacı; orman yangını sırasında uydu ve sensörlerden gelen emisyon verilerini toplamak, atmosferik yayılım modelleriyle dumanın hareket yönünü tahmin etmek, hava kalitesi indeksini hesaplamak ve vatandaş, halk sağlığı uzmanı, çevre mühendisi gibi paydaşlara uyarı ve rapor sunmaktır.
- Vatandaş — bölgesel hava kalitesi uyarısı alır.
- Çevre Mühendisi — duman bileşenlerini analiz eder, raporlar üretir.
- Halk Sağlığı Uzmanı — risk seviyelerine göre kamu sağlığı bildirimi yayımlar.
- CAMS API (dış sistem) — atmosferik veri sağlayıcı.
- Uydu / IoT Sensör Ağı (dış sistem) — ham emisyon ölçümü.

Use Case Diyagramı

- Use Case diyagramı, sistemin kullanıcılarının (aktörler), sistemle etkileşimde bulunduğu durumları gösterir.
- Bu diyagram, kullanıcıların hangi eylemlere erişebileceğini ve sistemin bu eylemleri nasıl gerçekleştireceğini açıklar.

- **Aktörler:**

Kullanıcı (Araştırmacı)

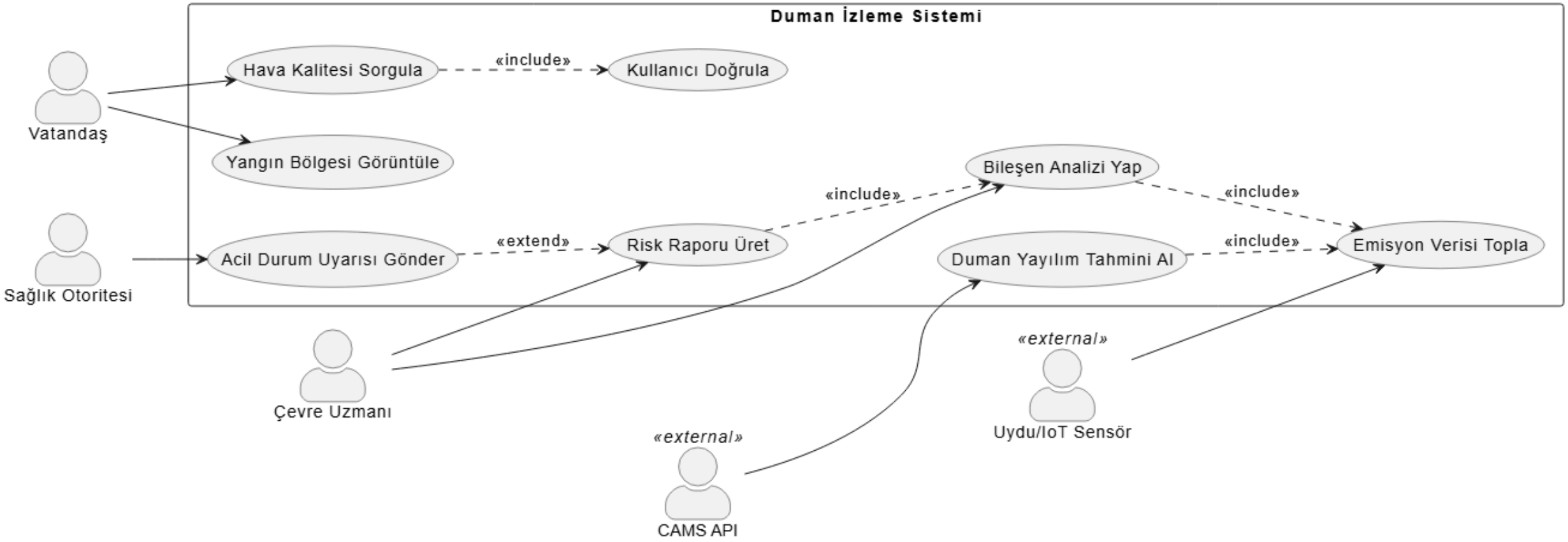
CAMS Sistemi

Halk Sağlığı Uzmanı

Çevre Mühendisi

Uydu/IOT Sensör

use case diyagramı



- Use case diyagramı, sistemin "ne yaptığını" kullanıcı bakış açısıyla değerlendirir.
- Amaç; aktörler ile sistem arasındaki fonksiyonel etkileşimi görsel olarak tasarlamaktır.
- <<include>> gerçekleşmesi zorunlu, <<extend>> ise gerçekleşmesi opsiyonel olan davranışları belirtir.

Use case açıklamaları

Hava Kalitesi Sorgulama use case: Vatandaşın konumuna göre AQI (Air Quality Index) döner. *Kullanıcı Doğrula* use case *include* edilir. Çünkü kimlik kontrolü zorunludur.

Duman Yayılım Tahmini use case : CAMS API'den gelen rüzgâr/atmosfer verisiyle çalışır; veri toplama olmadan çalışamayacağı *için Emisyon Verisi Topla use case include* edilir.

Risk Raporu Üret use case : Çevre Analizi Uzmanı tarafından hazırlanır; aynı ekibin gerçekleştirdiği bileşen analizini içerir.

Acil Durum Uyarısı use case: Risk eşiği aşıldığında işleme girer; bu durum <<extend>> ilişkisiyle gösterilir. Yani her zaman çalışmaz, koşullu davranıştır.

Sınıf Diyagramları

- Sınıf diyagramı sistemin statik yapısının modellenmesidir.
- Sınıflar, öznitelikler, metotlar ve aralarındaki ilişkiler olarak aggregation, composition, inheritance, dependency ilişkileri olarak çeşitlendirilir.
  - Nesne yönelimli tasarımın çatısını oluşturur.
- Her sınıfın özellikleri ve metotları da belirtilir.

Sınıf Diyagramı

Sınıf diyagramı, sistemdeki sınıfları ve bu sınıflar arasındaki ilişkileri gösterir.

Her sınıfın özellikleri ve metotları da belirtili

Yangın

Özellikler: konum, tarih, emisyonMiktarı, şiddet
Metotlar: emisyonHesapla(), yangınBilgisiAl()

Duman

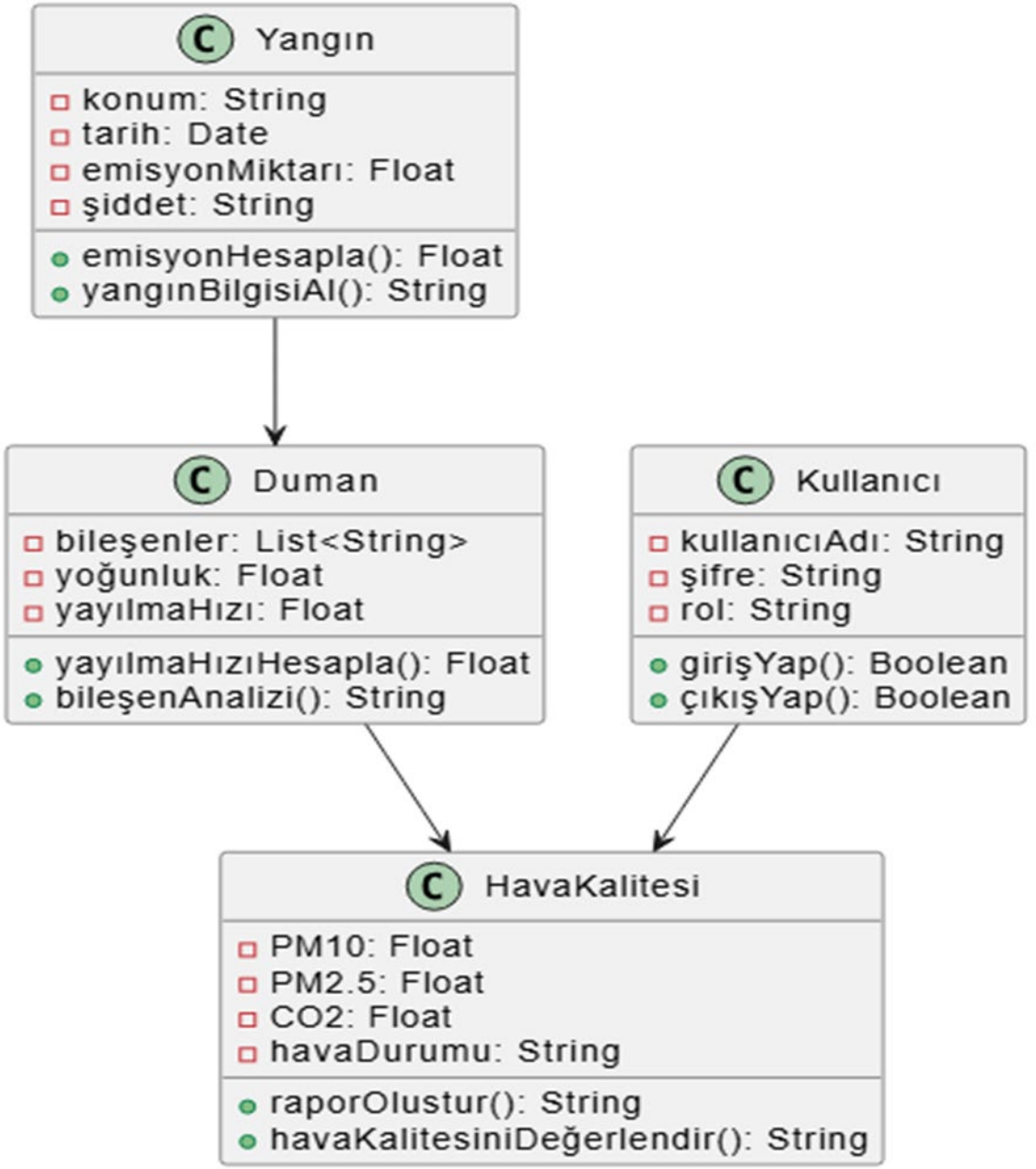
Özellikler: bileşenler, yoğunluk, yayılmaHızı
Metotlar: yayılmaHızıHesapla(), bileşenAnalizi()

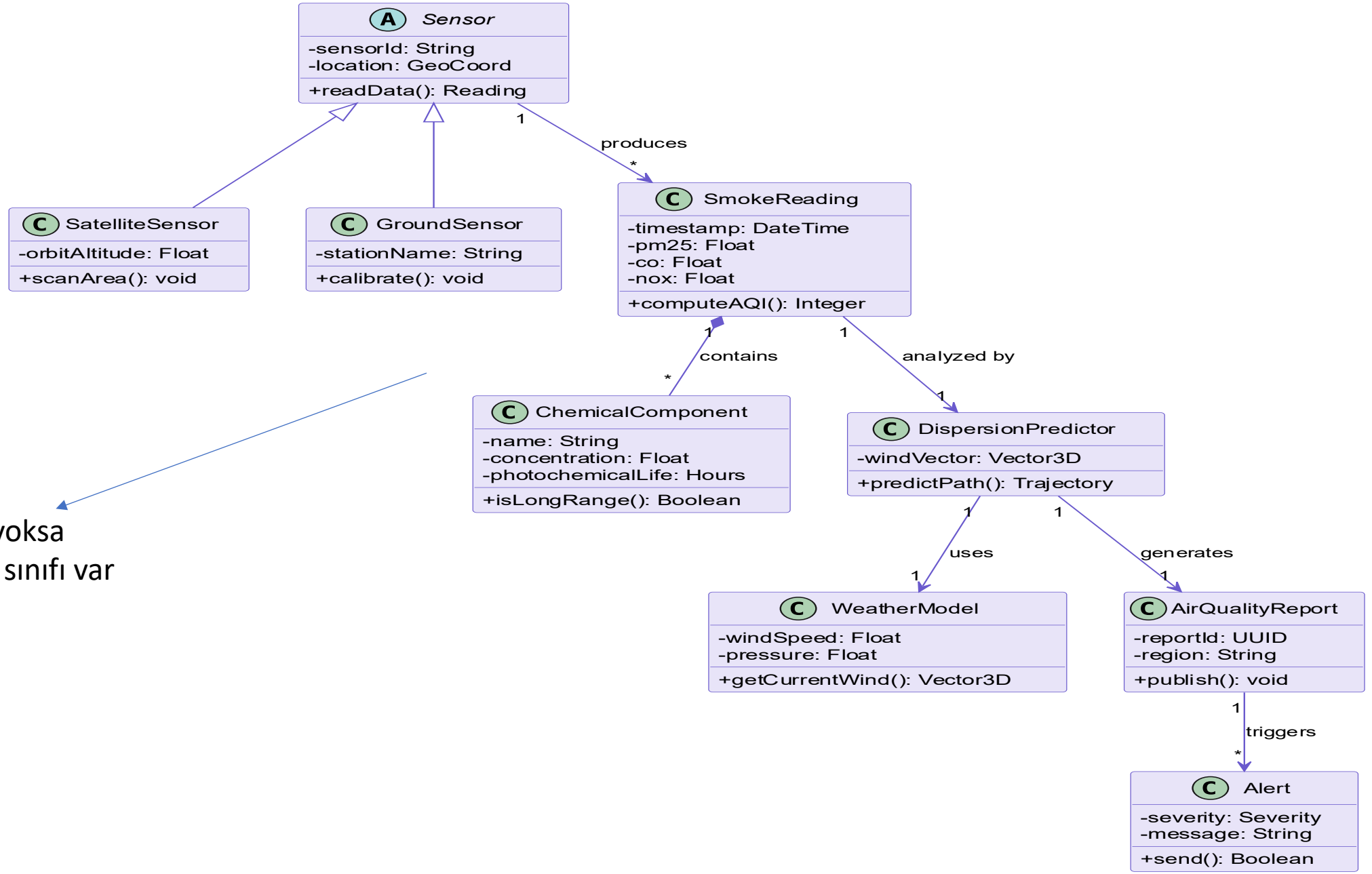
HavaKalitesi

Özellikler: PM10, PM2.5, CO2, havaDurumu
Metotlar: raporOlustur(), havaKalitesiniDeğerlendir()

Kullanıcı

Özellikler: kullanıcıAdı, şifre, rol
Metotlar: girişYap(), çıkışYap()





Composition ilişkisi SmokeReading sınıfı yoksa ChemicalComponent sınıfı var olmaz.

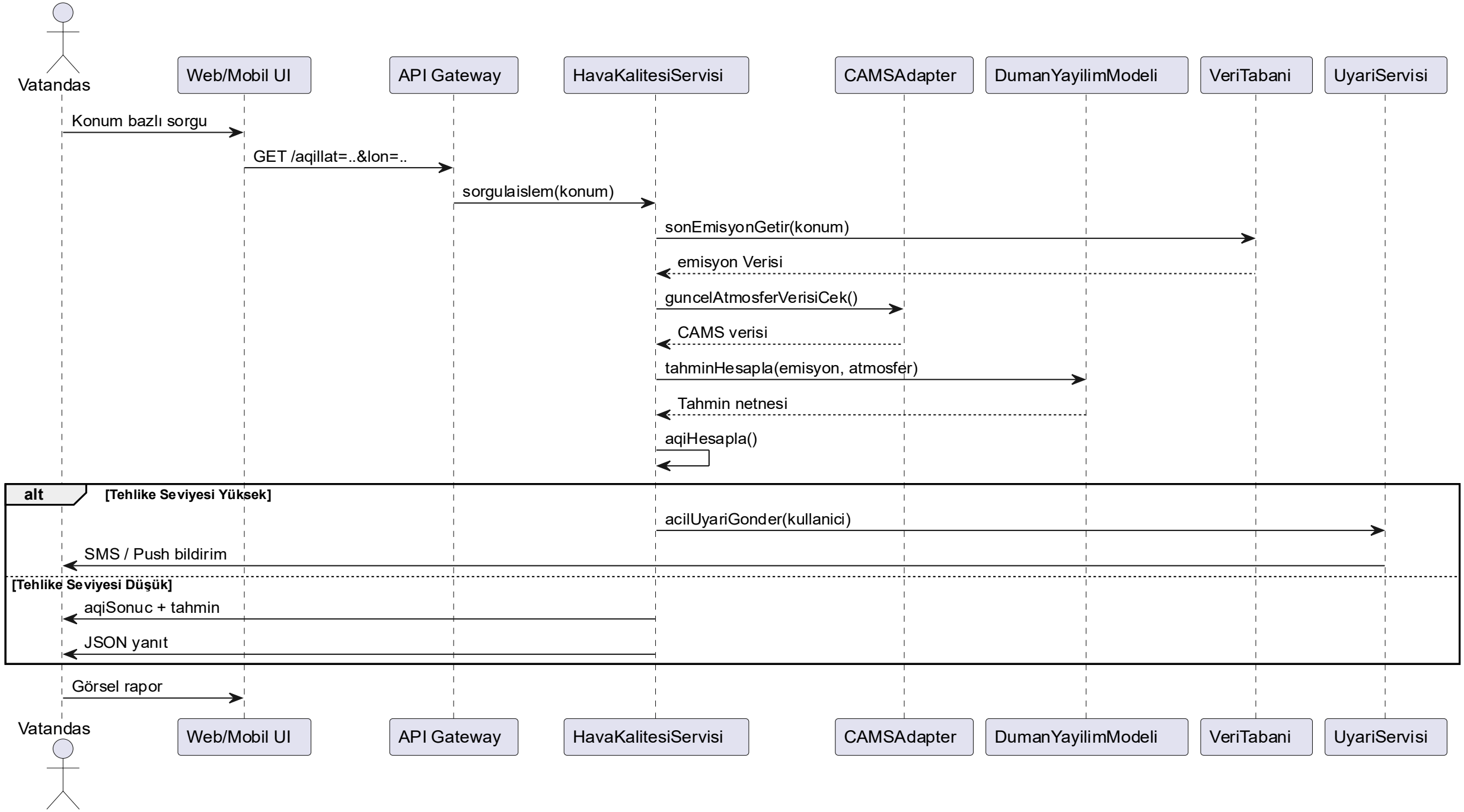
Sequence (Sıralama) Diyagramı

- Sequence diyagramı, belirli bir use case'in zaman ekseninde nasıl gerçekleştiği gösterilir.
- Nesneler arasındaki mesaj alışverişi ve sıralanışı modellenir.
- Senkron çağrılar düz okla, dönüş değerleri kesikli ok ile çizilir.
- Bir senaryo: Vatandaş hava kalitesini sorgular.

  Tahmin üretilir.

  Uyarı gönderilir.

Belirli bir kullanım senaryolarının zaman içindeki akışı gösterilmektedir. Bu diyagram, nesneler arasındaki etkileşimleri ve mesaj alışverişinü zaman sırasına göre düzenler.


Vatandas
Web/Mobil UI
API Gateway
HavaKalitesiServisi
CAMSAdapter
DumanYayilimModeli
VeriTabani
UyariServisi
Konum bazlı sorgu
GET /aqillat=..&lon=..
sorgulaislem(konum)
sonEmisyonGetir(konum)
emisyon Verisi
guncelAtmosferVerisiCek()
CAMS verisi
tahminHesapla(emisyon, atmosfer)
Tahmin netnesi
aqiHesapla()
alt
[Tehlike Seviyesi Yüksek]
acilUyariGonder(kullanici)
SMS / Push bildirim
[Tehlike Seviyesi Düşük]
aqiSonuc + tahmin
JSON yanıt
Görsel rapor
Vatandas
Web/Mobil UI
API Gateway
HavaKalitesiServisi
CAMSAdapter
DumanYayilimModeli
VeriTabani
UyariServisi

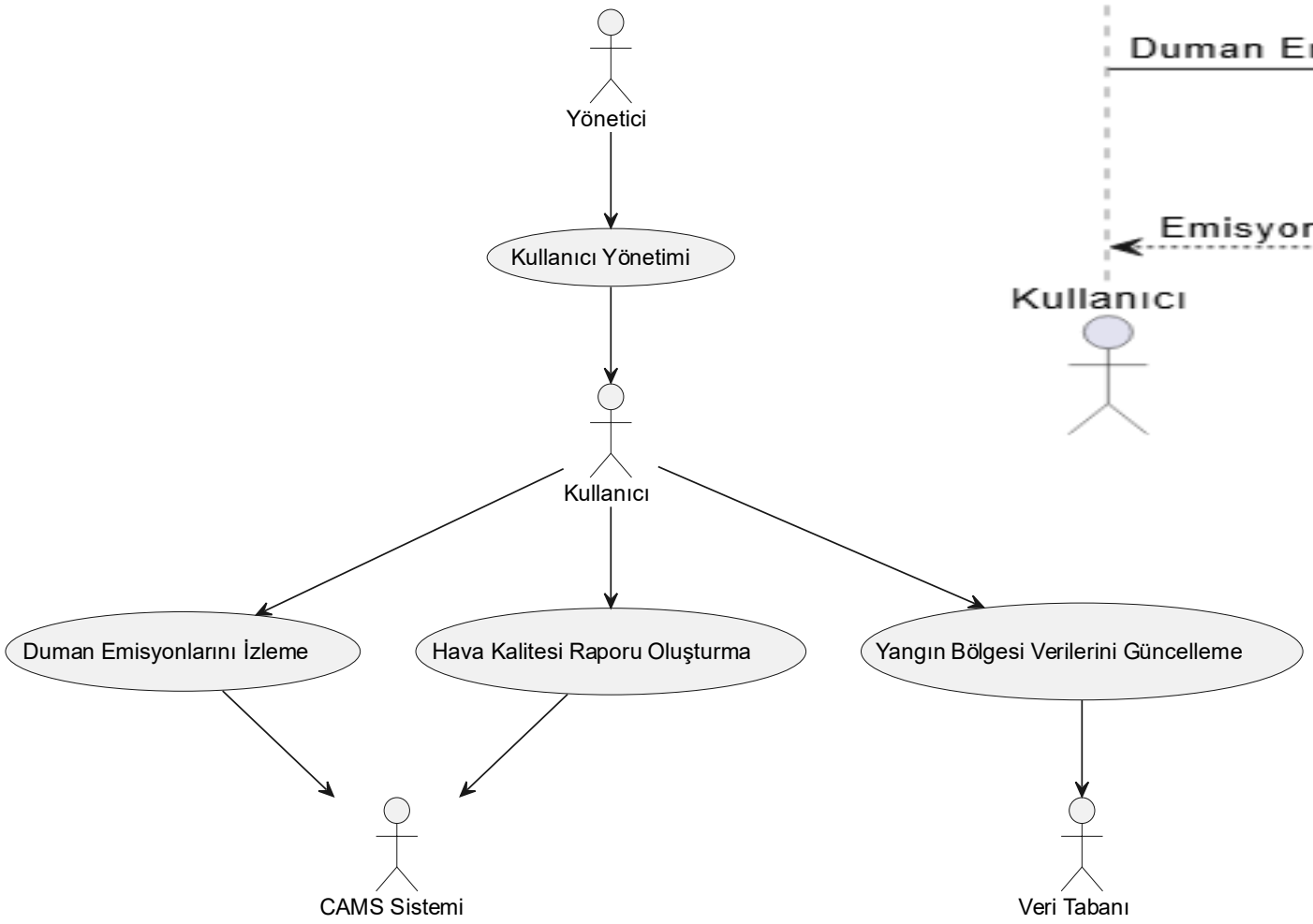
Sequence (Sıralama) Diyagramı

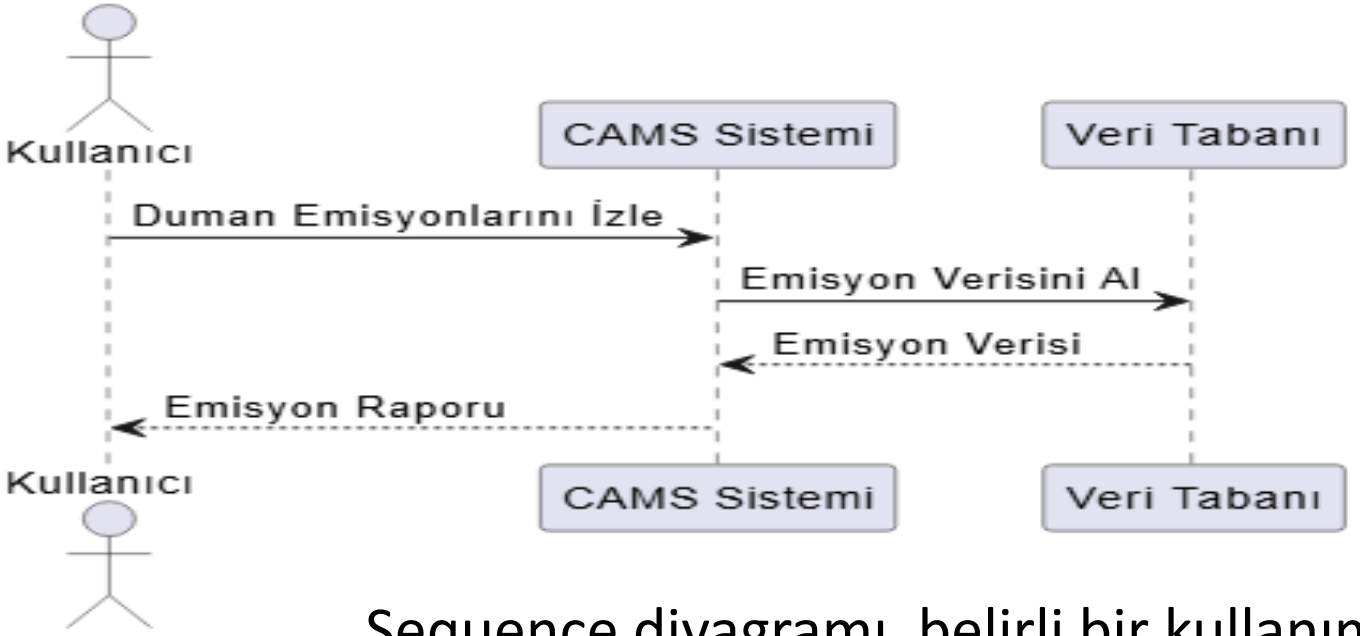
- API Gateway katmanı, mikroservis mimarisinde giriş noktası görevini görerek güvenlik ve yönlendirme işlerini gerçekleştirir.
- Alt blokta alternatif akış gerçekleştirilir: AQI değeri 150'yi aşarsa uyarı tetiklenir.
  - Bu, use case diyagramındaki <<extend>> davranışının sequence diyagramındaki karşılığıdır.
  - guncelAtmosferVerisiCek() çağrısı asenkron yapılabilir; performans iyileştirmesi için caching önerilir.
- Senkron çağrılar (->>) ve dönüşler (-->>) olarak ayrılmıştır; böylece hata izleme ve analiz kolaylaşır.

Sequence (Sıralama) Diyagramı

- Hava Kalitesi Servisi use case tehlike seviyesine göre farklı akışlar izler.
- Tehlike seviyesi yüksek ise, UyariServisi use case üzerinden kullanıcıya SMS/Push bildirimi gönderilir.
- Tehlike seviyesi düşük ise, kullanıcıya aqiSonuc ve tahmin bilgileri JSON yanıtı olarak iletililir.

Ayrıntılı use case & Sequence Diyagramı Örneği



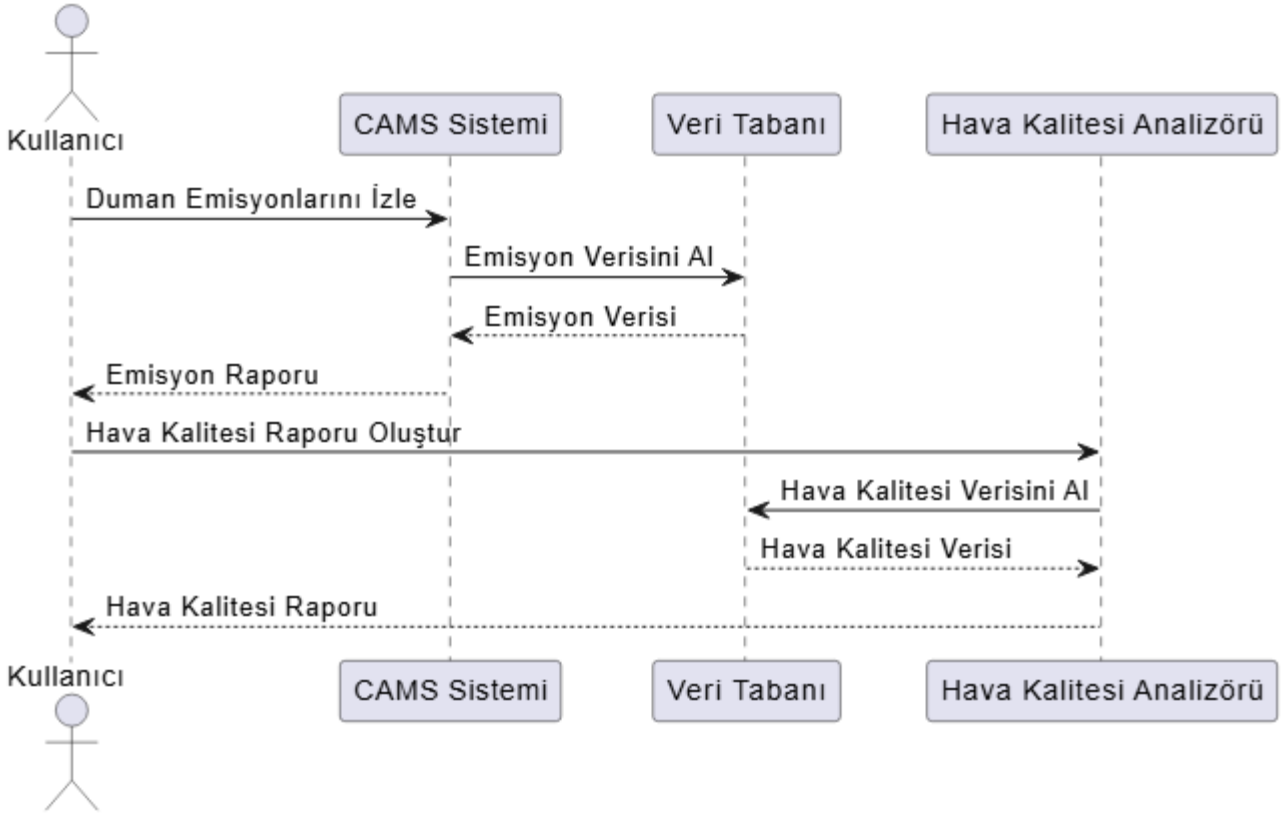


Sequence diyagramı, belirli bir kullanım senaryosunun zaman içindeki akışını gösterir.

Nesneler arasındaki etkileşimleri ve mesaj alışverişini zaman sırasına göre düzenler.

Ayrıntılı Senaryo ise bir kullanıcının duman emisyonlarını izleme süreci sürecidir.

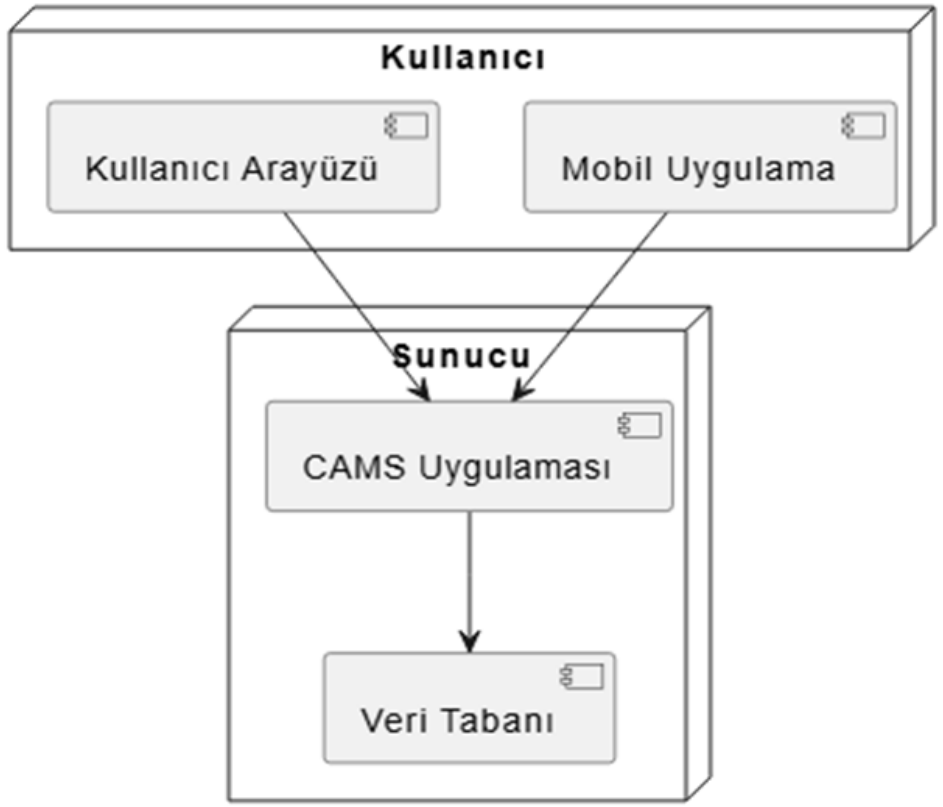
Daha Ayrıntılı Sequence Diyagramı



Daha Ayrıntılı senaryo ise bir kullanıcının duman emisyonlarını izleme süreci ile hava kalitesi raporu oluşturma sürecidir

Deployment (Dağıtım) Diyagramı

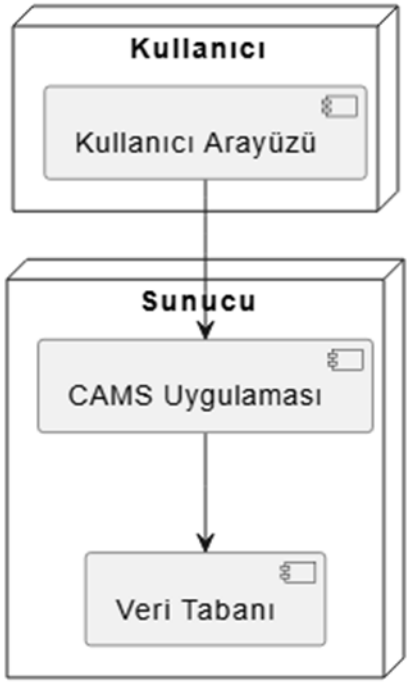
- Yazılım bileşenlerinin fiziksel dağılımını ve bu bileşenlerin hangi donanım üzerinde çalıştığını gösterir.
- Sunucular (node), cihazlar (device), yürütülebilir parçalar (artifact) ve protokoller (HTTP, MQTT, TCP) bu diyagramda yer alır.

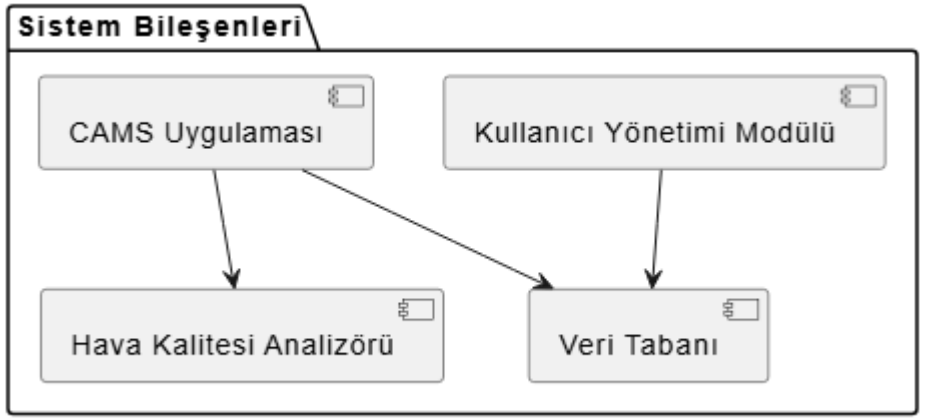


- Bileşenler:
  - Sunucu
  - Veri Tabanı
  - Kullanıcı Arayüzü

- Detaylı Bileşenler:
  - Sunucu: CAMS uygulaması ve veri tabanı
  - Kullanıcı: Kullanıcı arayüzü ve mobil uygulama.

Detaylı Orman Yangınları Deployment Diyagramı





Sistem; uydu, yer istasyonu, IoT ağ geçidi, bulut uygulama sunucusu, veritabanı, web ve mobil istemcileri kapsayan çok katmanlı (multi-tier) dağıtık bir mimaridir.

Veri Toplama Katmanı: Satellite Node Ground Station , IoT Gateway

İşleme Katmanı :

Application Server (AWS EC2) ve Database

Sunum Katmanı Web Dashboard ve Mobile Client

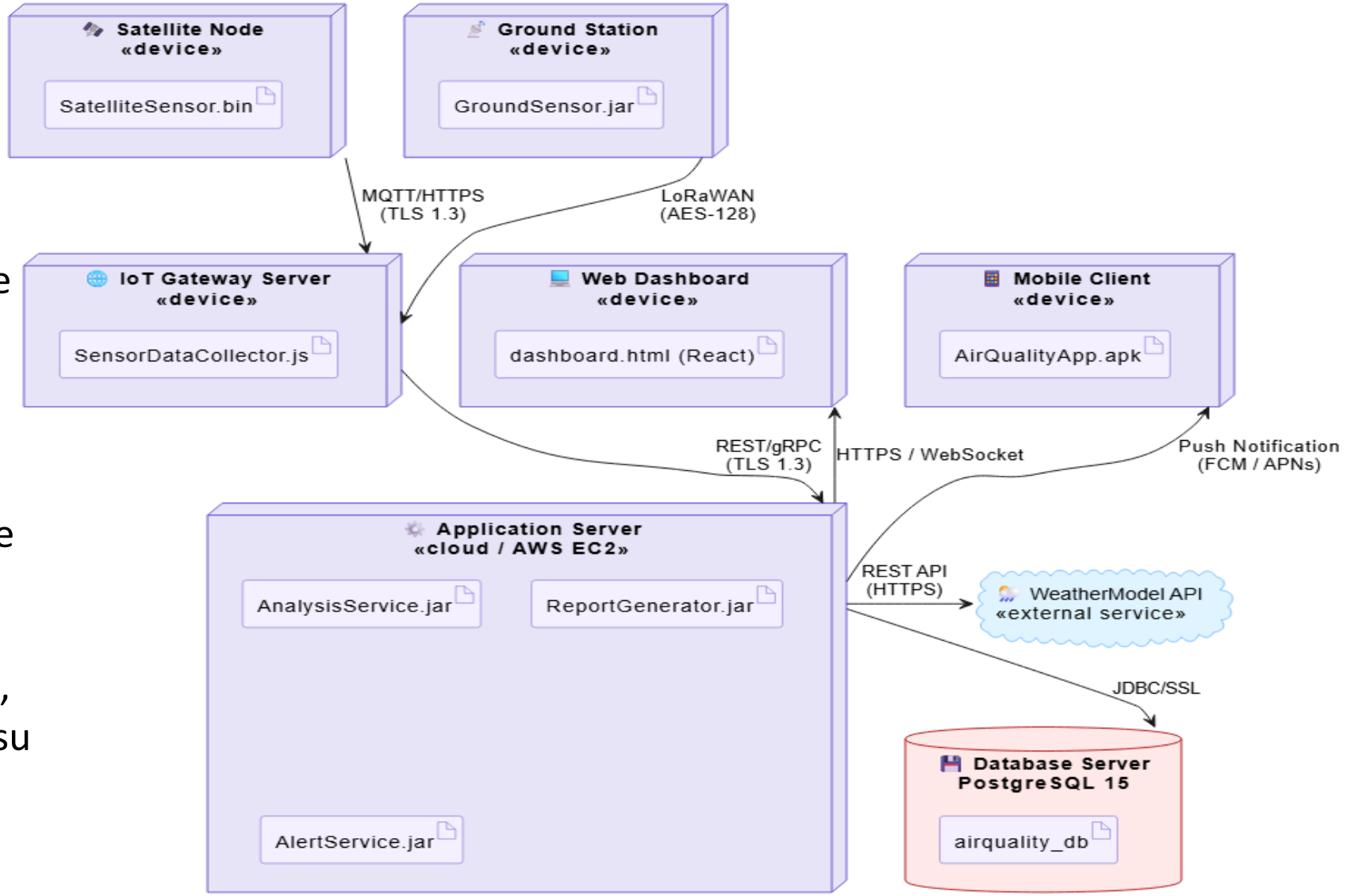
Veri akışı:

i)Sensörler ile Iot Gateway arasında,

ii) IOT Gateway ile Uygulama Sunucusu arasında,

iii) Uygulama Sunucusu ile Veritabanı arasında ,

iv) Uygulama Sunucu ile Kullanıcı Arayüzleri arasında gerçekleşir.



Düğümlerin (Nodes) Ayrıntılı Açıklamaları

1.Satellite Node «device» : Uzaydan atmosferik veri (hava kalitesi, partikül, gaz oranları vb.) toplayan uydu donanımı.

IoT Gateway'e MQTT/HTTPS (TLS 1.3) üzerinden şifreli veri gönderir.

2. Ground Station «device»: Yerdeki sensör istasyonu.

IOT Gateway'e LoRaWAN (AES-128) protokolüyle düşük güçte, geniş alan kablosuz iletim yapar.

3. IoT Gateway Server «device» : Uydu ve yer istasyonundan gelen verileri birleştirip Application Server'a iletir.

REST/gRPC (TLS 1.3) protokolüyle bulut sunucusuna bağlanır.

4. Application Server «cloud / AWS EC2» Sistemin beynidir.

Düğümlerin Ayrıntılı Açıklamaları

5. Database Server – PostgreSQL 15 : Uygulama sunucusuna JDBC/SSL üzerinden bağlı.

Tüm sensör ölçümleri, raporlar ve kullanıcı verileri burada saklanır.

6. WeatherModel API «external service» Harici hava durumu servisi.

Application Server, REST API (HTTPS) üzerinden meteorolojik verileri çekerek analizi zenginleştirir.

7. Web Dashboard «device» : Yöneticiler/analistler için web arayüzü.

Sunucuya HTTPS / WebSocket ile bağlanır. Gerçek zamanlı veri akışı için WebSocket kritiktir.

8. Mobile Client «device» : Android tabanlı son kullanıcı uygulamasıdır.

Application Server'dan Push Notification (FCM / APNs) ile anlık uyarı alır.

İletişim Protokolleri ve Güvenlik

Bağlantı	Protokol	Güvenlik
Satellite → Gateway	MQTT / HTTPS	TLS 1.3
Ground Station → Gateway	LoRaWAN	AES-128
Gateway → App Server	REST / gRPC	TLS 1.3
App Server → Database	JDBC	SSL
App Server → WeatherModel	REST API	HTTPS
App Server → Web Dashboard	HTTPS / WebSocket	TLS
App Server → Mobile Client	Push (FCM/APNs)	Sağlayıcı şifreleme

Özet olarak: Örnek Deployment Diyagramı

- Application Server katmanında Analysis, Report, Alert servislerinin ayrılması mikroservis yapısının özelliğidir.
  - Faydası bağımsız olarak ölçeklenebilirlik sağlamasıdır.
- Bulut tabanlı AWS EC2 kullanılmasıyla yüksek erişilebilirlik ve esnek kapasite sağlanmıştır.
- Çok kanallı sunum: Webi Mobil, Push bildirimleri çok kanallı sunumlar sağlar.
  - Böylece farklı kullanıcı senaryolarına hizmet verilebilir.
- WeatherModel API ile analiz doğruluğunun artırılabilmesi harici (external) entegrasyon örneğidir.

Özet olarak: Örnek Deployment Diyagramı

- Gerçek zamanlı hava kalitesini izlemede IoT , Cloud , Client mimarileri modelleniyor.
- Veri uzaydan/yerden toplanıyor, bulutta işleniyor, veritabanında saklanıyor ve hem web hem mobil üzerinden kullanıcılara ulaştırılıyor.
- Güvenlik, ölçeklenebilirlik ve modülerlik bakımından oldukça güçlü bir tasarım örneğidir.

Component Diyagram

- Sistemin bileşenlerini ve bu bileşenler arasındaki ilişkileri gösterir.
- Sistem mantıksal yazılım modüllerine ayrılır ve aralarındaki arayüz (interface) ilişkileri gösterilir.
- Component diyagram, sistemin mantıksal yapısını (hangi bileşen hangi arayüzü sağlıyor/tüketiyor) gösterir.
- Deployment diyagramından farkı, fiziksel donanım yerine yazılım bileşenleri ve arayüzlerine odaklanmasıdır.
- Modülerlik, yeniden kullanılabilirlik ve bakım kolaylığını ortaya koyar.

Sistem Bileşenleri

Sensör Bileşenleri (veri üreticileri)

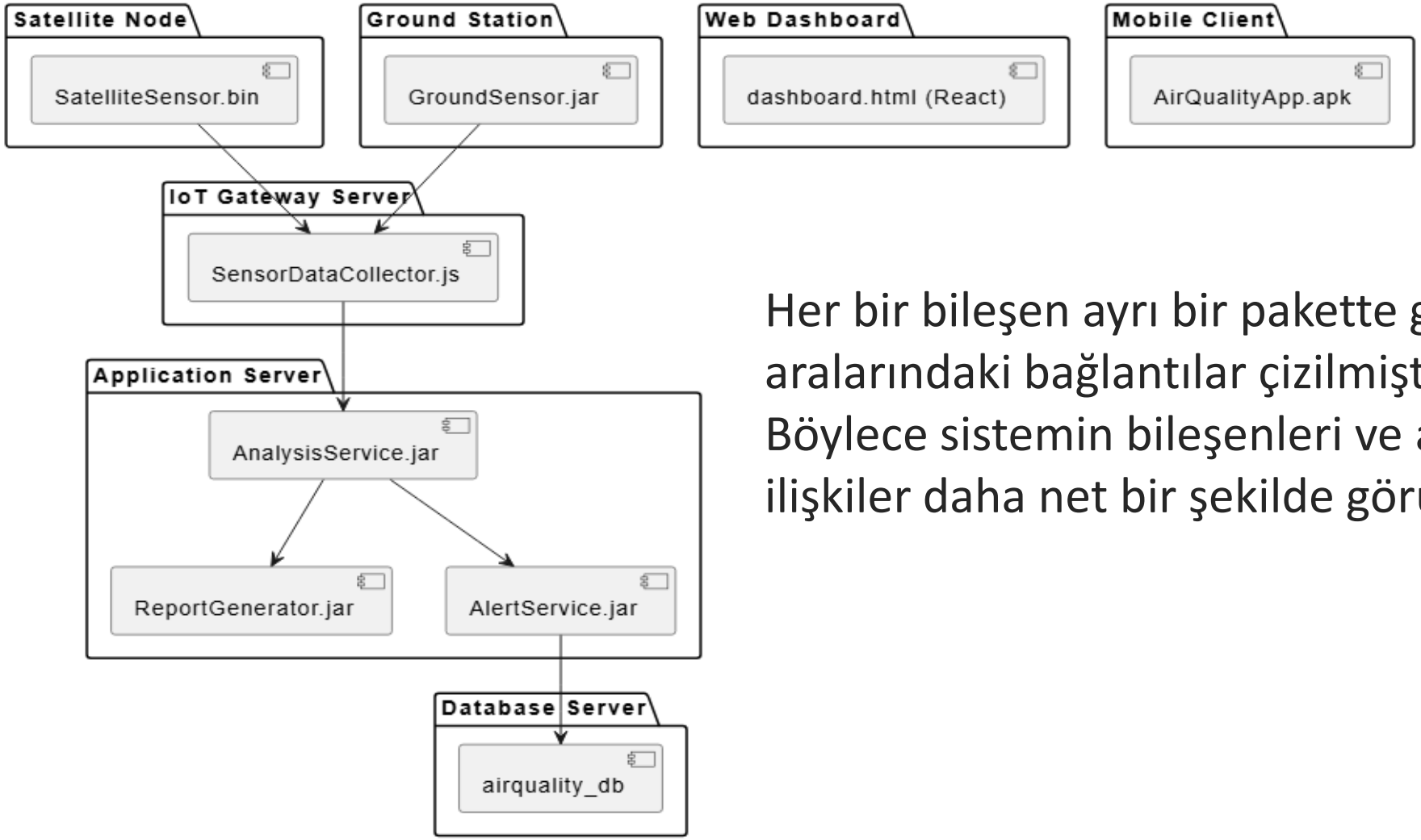
Veri Toplama Bileşeni (gateway)

Çekirdek İşleme Bileşenleri (analiz, rapor, uyarı)

Veri Erişim & Entegrasyon Bileşenleri (DB, harici API)

Sunum Bileşenleri (web, mobil)

Component Diyagramı



Her bir bileşen ayrı bir pakette gösterilmiş ve aralarındaki bağlantılar çizilmiştir.
Böylece sistemin bileşenleri ve aralarındaki ilişkiler daha net bir şekilde görülebilmektedir.

Dependency Inversion İlkesi Açıklamaları

- SensorDataCollector.js, SatelliteSensor.bin ve GroundSensor.jar bileşenlerinden bağımsız çalışmaktadır.
  - SensorDataCollector.js, alt bileşenlerin somut sınıflarına değil, **soyut arayüzlerine** bağımlıdır.

- AnalysisService.jar, ReportGenerator.jar ve AlertService.jar bileşenleri, SensorDataCollector.js'den bağımsız çalışmaktadır.
  - Bu bileşenler, SensorDataCollector.js'nin somut sınıflarına değil, **soyut arayüzlerine** bağımlıdır.

- AlertService.jar, airquality_db bileşenine bağımlıdır.
  - Bağımlılık yönü yüksek seviyeli bileşenden (AlertService.jar) düşük seviyeli bileşene (airquality_db) doğrudur.
  - Burada da **dependency inversion** prensibi uygulanmıştır.

Dependency Inversion Principle –DIP İlkesinin Önemi

- DIP, yazılım tasarımında ve geliştirilmesinde kullanılan SOLID ilkelerinden sonuncusudu.
- Yüksek düzeydeki modüllerin düşük düzeydeki modüllere bağlı olmamasını sağlar.
  - Her ikisinin de soyut kavramlara bağlı olmasını önerir.
  - yüksek düzeydeki bileşenler, düşük düzeydeki bileşenlerin somut uygulamalarına değil, *soyut arayüzlerine* bağlıdır.
  - Bileşenler arasındaki bağımlılıklar azaltılarak, modüler ve esnek bir yazılım mimarisi elde edilir.

SOLID İLKESİ: Dependency Inversion Principle -DIP

Bağımlılığı Tersine Çevirme İlkesi (DIP), üst düzeydeki bileşenlerin alt düzeydeki bileşenlere bağımlı olmamasıdır.

Her iki düzeyin de soyutlamalara (arayüzlere) bağımlı olmasını gerektiren bir yazılım tasarımı ilkesidir. Diğer bir ifade ile:

Bağımlılıkların Tersine Çevrilmesi ilkesi, üst düzeydeki kodun alt düzeydeki kodu doğrudan içe aktarması yerine, her ikisinin de bir ara soyutlamaya (arayüz) bağlı olmasıdır.

Böylece bileşenler ayrıştırılmıştır; üst düzeydeki bileşenler, alt düzeydeki bileşenlerin değişikliklerinden etkilenmez.

Örneğin yüksek düzeyli bir Rapor Oluşturucu sınıfının, düşük düzeyli bir SQLDatabase sınıfına doğrudan bağlı olmayıp, bir IDataSource arayüzüne bağlı olmasıdır.

Avantajları: Yüksek düzeyli iş mantığını değiştirmeden uygulamaların kolayca değiştirilebilmesi ile ESNEKLİK (örneğin, veritabanı sağlayıcılarını değiştirme), belirli bileşenleri güncellemedeki veya değiştirmedeki kolaylık (bakım), birim testleri sırasında bağımlılıkların mocking işlemini gerçekleştirebilmesi verilebilir.

Yazılım Testinde Mocking nedir?

Birim testleri (unit test) yazılırken, test edilen kodun bağımlı olduğu veritabanları, API servisleri, dosya sistemleri gibi dış bileşenlerinin gerçek halleri yerine, onların davranışlarını taklit eden sahte nesneler (mock objects) kullanılması tekniğidir.