KARADENİZ TEKNİK ÜNİVERSİTESİ
OF TEKNOLOJİ FAKÜLTESİ
YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ

OCS – TELEKOM ONLINE ÖDEME SİSTEMLERİNDE MİKROSERVİSLERİN KULLANILMASI VE SÜREKLİ ENTEGRASYON – SÜREKLİ DAĞITIMI

TASARIM PROJESİ

BERKİN DÜNDAR
RAMAZAN FIRAT AKDAĞ

2024 -2025 GÜZ DÖNEMİ

KARADENİZ TEKNİK ÜNİVERSİTESİ
OF TEKNOLOJİ FAKÜLTESİ
YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ

ÇALIŞMANIN ADI

OCS TELEKOM ONLINE ÖDEME SİSTEMLERİNDE MİKROSERVİSLERİN KULLANILMASI VE SÜREKLİ ENTEGRASYON – SÜREKLİ DAĞITIMI

ÖĞRENCİ ADI SOYADI

Berkin DÜNDAR
Ramazan FIRAT AKDAĞ

Bu projenin teslim edilmesi ve sunulması tarafımca uygundur.

Danışman : Öğr. Gör. Zeynep ŞAHİN TİMAR .....

2024-2025 GÜZ DÖNEMİ

IEEE Etik Kuralları
IEEE Code of Ethics

Mesleğime karşı şahsi sorumluluğumu kabul ederek, hizmet ettiğim toplumlara ve üyelerine en yüksek etik ve mesleki davranışta bulunmaya söz verdiğimi ve aşağıdaki etik kurallarını kabul ettiğimi ifade ederim:

1. Kamu güvenliği, sağlığı ve refahı ile uyumlu kararlar vermenin sorumluluğunu kabul etmek ve kamu veya çevreyi tehdit edebilecek faktörleri derhal açıklamak;
2. Mümkün olabilecek çıkar çatışması, ister gerçekten var olması isterse sadece algı olması, durumlarından kaçınmak. Çıkar çatışması olması durumunda, etkilenen taraflara durumu bildirmek;
3. Mevcut verilere dayalı tahminlerde ve fikir beyan etmelerde gerçekçi ve dürüst olmak;
4. Her türlü rüşveti reddetmek;
5. Mütenasip uygulamalarını ve muhtemel sonuçlarını gözeterek teknoloji anlayışını geliştirmek;
6. Teknik yeterliliklerimizi sürdürmek ve geliştirmek, yeterli eğitim veya tecrübe olması veya işin zorluk sınırları ifade edilmesi durumunda ancak başkaları için teknolojik sorumlulukları üstlenmek;
7. Teknik bir çalışma hakkında yansız bir eleştiri için uğraşmak, eleştiriyi kabul etmek ve eleştiriyi yapmak; hatları kabul etmek ve düzeltmek; diğer katkı sunanların emeklerini ifade etmek;
8. Bütün kişilere adilane davranmak; ırk, din, cinsiyet, yaş, milliyet, cinsi tercih, cinsiyet kimliği, veya cinsiyet ifadesi üzerinden ayırımcılık yapma durumuna girişmemek;
9. Yanlış veya kötü amaçlı eylemler sonucu kimsenin yaralanması, mülklerinin zarar görmesi, itibarlarının veya istihdamlarının zedelenmesi durumlarının oluşmasından kaçınmak;
10. Meslektaşlara ve yardımcı personele mesleki gelişimlerinde yardımcı olmak ve onları desteklemek.

IEEE Yönetim Kurulu tarafından Ağustos 1990'da onaylanmıştır.

İÇİNDEKİLER

İçindekiler

ÖNSÖZ

Telekomünikasyon sektörü, modern teknolojinin gelişiminde kritik bir role sahip olan bir alandır. Kullanıcı ihtiyaçlarının artan çeşitliliği ve bu ihtiyaçlara yanıt verebilecek esnek, ölçeklenebilir ve yüksek performanslı sistemlere duyulan gereksinim, bu projeyi gerçekleştirme fikrini doğurmuştur. Bu çalışmada, çevrimiçi ücretlendirme sistemleri (Online Charging Systems - OCS) için modern bir mimari tasarım ve sürekli entegrasyon - sürekli dağıtım (CI/CD) süreçlerinin etkin yönetimine yönelik yenilikçi bir çözüm önerilmektedir

Projemizin gerçekleştirilmesinde rehberlik eden, bize değerli kaynaklarını açarak fikir edinmemize katkı sağlayan ve telekomünikasyon sistemlerine dair bilgi birikimiyle ilham veren i2i Systems firmasına teşekkürlerimizi sunarız. Özellikle, projemiz boyunca bilgi ve tecrübelerini bizlerle paylaşan Sayın **Hacı Mehmet Atılgan** ve **Mennan Tekbir**, rehberlikleriyle projemizin şekillenmesine büyük katkı sağlamışlardır.

Bu çalışmayı tamamlamak, bir ekip çalışmasının, araştırmanın ve öğrenmenin sonucudur. Tüm bu süreci destekleyen akademik danışmanımıza, ekip arkadaşlarımıza ve katkıda bulunan tüm paydaşlara teşekkür ederiz. Çalışmamızın telekomünikasyon sektöründe daha verimli, modern ve sürdürülebilir sistemlerin inşa edilmesine katkı sağlamasını ümit ediyoruz.

ÖZET

Telekomünikasyon sektörü tüm dünyada teknoloji denilince akla gelen ilk sektörlerdendir. Ve her ülke gibi ülkemizde de birçok şirket bu alanda özveri ile çalışmakta, sektörümüzün gelişmesine katkı sağlamaktadır. Günümüzde Telekomünikasyon müşterilerinin sayılarındaki artış ve bu alandaki hizmetlere olan talep artmakta ve her geçen gün daha performanslı, daha stabil ve daha güvenilir sistemlere ihtiyaçlar da aynı oranda artmaktadır.

Bu ihtiyaçların başında da Telekomünikasyon firmalarının müşterilerinin kullanımlarını, ödemelerini ve aldığı hizmetleri yönetebilmesini, bu sayede müşterilerin memnuniyetini artırmayı hedefleyen daha etkili ödeme ve bilgi yönetim sistemlerinin geliştirilmesi gelmektedir. Yazılım ve son teknolojik gelişmelerin yardımıyla, bu ihtiyaçların karşılanabilmesi için yeterli ortamları sağlayabilmek mümkündür.

Bu çalışmanın amacı, günümüzün son teknolojik gelişmelerini oyuna dahil ederek; etkili, performanslı ve güvenilir sistemler inşa etmek ve bunların sürekli entegrasyon – sürekli dağıtım süreçlerini optimize edecek bir yordam sunmaktır. Mimari Tasarımlar oluşturup görselleştirmeler yapılarak, telekomünikasyon sektöründeki bu sistemlerin inşasında nasıl bir yol izlenebileceğine dair perspektif sunulacak ve devamında bu proje kurulan mimari doğrultusunda geliştirilecektir. Bu projenin mimari tasarımı ve telekomünikasyon yazılımlarında mikroservislerin kullanımı konusunda yenilikçi bir kaynak oluşturması planlanmaktadır.

Telekomünikasyon sektörü gibi karmaşık ve geniş ölçekli sistemlerde, CI/CD (Sürekli Entegrasyon ve Sürekli Dağıtım) süreçlerinin etkili bir şekilde yönetilememesi, işletmelerin performansını ve kullanıcı memnuniyetini olumsuz etkileyebilmektedir. Mikroservis mimarisi gibi modern yaklaşımlar, bu zorlukların üstesinden gelmek için ideal bir çözüm sunarken, bu sistemlerin doğru şekilde yapılandırılması ve CI/CD süreçlerine entegre edilmesi kritik bir öneme sahiptir.

Bu çalışma, telekomünikasyon sektörü için tasarlanan geniş kapsamlı ve mikroservis mimarisiyle inşa edilen bir projede, CI/CD süreçlerinin nasıl etkili bir şekilde yönetilebileceğini ortaya koymayı hedeflemektedir. Bu bağlamda, karmaşık sistemlerin sürekli entegrasyon ve sürekli dağıtım süreçlerinde karşılaşılan sorunları ele alarak, bu süreçlerin optimizasyonuna dair yenilikçi yöntemler sunulacaktır. Ayrıca, telekomünikasyon yazılımlarının tasarım süreçlerinde dikkat edilmesi gereken noktalar ve mimari tasarımlar üzerinde geliştirilen yaklaşımlar paylaşılacaktır.

Çalışmanın sonuçları, hem mimari tasarım prensipleri hem de CI/CD süreçlerinin entegrasyonu açısından sektöre değerli bir rehber sunmayı amaçlamaktadır. Özellikle, modern teknolojilerin kullanımıyla yüksek performanslı ve güvenilir telekomünikasyon sistemlerinin geliştirilmesine yönelik somut bir yol haritası ortaya konacaktır. Bu projenin, sadece teorik değil, aynı zamanda pratik bir bakış açısı sunarak sektördeki diğer projeler için referans niteliğinde olması beklenmektedir.

1. GENEL BİLGİLER

1.1 Giriş

Günümüzde mobil iletişim ağları hızla gelişmekte ve kullanıcılar, çeşitli servis sağlayıcılardan geniş bir yelpazede hizmet talep etmektedir. İlk başlarda sesli iletişim ve SMS gibi basit hizmetlerle sınırlı olan bu süreç, internetin mobil cihazlarda yaygınlaşması ile birlikte veri odaklı ve çoklu ortam içerikli servisleri kapsayacak şekilde genişlemiştir. Kullanıcılar artık video konferans, çevrimiçi oyun, bulut tabanlı hizmetler ve anlık mesajlaşma gibi çok çeşitli hizmetleri tek bir cihaz üzerinden kullanabilmektedir.

Bu çeşitlilik, mobil operatörler açısından büyük zorlukları beraberinde getirmektedir. Artan kullanıcı sayısı ve veri tüketiminin yoğunlaşması, operatörlerin ağ kaynaklarını etkin bir şekilde yönetme ve kullanıcıları adil bir ücretlendirme modeli ile faturalandırma gerekliliğini doğurmuştur. Bu noktada, çevrimiçi ücretlendirme sistemleri (Online Charging Systems - OCS), mobil ve sabit ağ hizmetlerinde kullanıcıya anlık olarak ücretlendirme yaparak denge sağlama işleviyle kritik bir öneme sahiptir.

Çevrimiçi ücretlendirme sistemleri, kullanıcının bir hizmeti tüketirken anlık olarak ücretlendirilmesine olanak tanımaktadır. Bu sistemler, yalnızca veri kullanımını izlemekle kalmayıp, aynı zamanda kullanıcının hesabındaki bakiye durumuna göre hizmetlerin kullanımını düzenlemektedir. Örneğin, veri paketi tükenmek üzere olan bir kullanıcıya otomatik olarak bildirim gönderilmesi veya yetersiz bakiye durumunda ek veri paketi satın alma imkânının sunulması gibi özellikler, kullanıcı deneyimini iyileştirmekte ve operatörler açısından gelir kaybını en aza indirmektedir.

Bununla birlikte, günümüz çevrimiçi ücretlendirme sistemlerinde bazı eksiklikler bulunmaktadır. Mevcut sistemler, farklı hizmet sağlayıcıların entegre edildiği çoklu ortam ve birleşik servislerde esneklik, ölçeklenebilirlik ve gerçek zamanlı performans açısından yetersiz kalabilmektedir. Bu bağlamda, daha yüksek performans ve esneklik sunan mikroservis tabanlı bir OCS mimarisine olan ihtiyaç giderek artmaktadır.

Ayrıca, bu tür kompleks sistemlerin sürekli entegrasyon (CI) ve sürekli dağıtım (CD) süreçlerinin yönetimi, artan karmaşıklık nedeniyle yeterince etkili yapılamamaktadır. Modern telekomünikasyon altyapıları, birden fazla servis sağlayıcı ve medya türünü (örneğin, ses, video konferans, çevrimiçi oyunlar) entegre ederken, CI/CD süreçlerinde karşılaşılan yavaşlık ve manuel müdahale gereksinimi gibi problemler, sistemlerin ölçeklenebilirliğini ve sürdürülebilirliğini olumsuz etkilemektedir. Bu durum, özellikle yazılım güncellemeleri, hata düzeltmeleri ve yeni özelliklerin hızlıca dağıtılmasını zorlaştırmakta, müşteri memnuniyeti ve sistem performansında kayıplara neden olmaktadır. Bu projede birden fazla servis sağlayıcı ve medya türünü entegre etmekten ziyade, daha basit medya türleri (sms, data, ses) üzerinde fiyatlandırma yapan modüler bir mimari geliştirilerek, bu özelliklerin sonradan kolayca farklı modüller eklenerek geliştiriminin devam edebileceği bir geliştirme süreci anlatılacaktır.

Yukarıda belirtilen sebeplerden ötürü, telekomünikasyon sektöründe kullanılan çevrimiçi ücretlendirme sistemlerinin geliştirilmesi ve modüler, esnek ve gerçek zamanlı çalışabilecek bir sistemin tasarlanması gerekliliği ortaya çıkmaktadır. Bu çalışma kapsamında, modern telekomünikasyon hizmetlerine uyum sağlayabilecek, CI/CD süreçlerini etkin bir şekilde yöneten, performansı yüksek ve kullanıcı deneyimini iyileştiren gelişmiş bir OCS prototipinin tasarlanması amaçlanmaktadır.

Bu projenin amacı, modüler bir sistem geliştirilerek farklı medya türlerinin (ses, video konferans, çevrimiçi oyun) entegrasyonunu kolaylaştırmayı, esneklik, ölçeklenebilirlik ve performans gibi metrikleri yükseltmeyi sağlayacak bir çözüm üretmektir. Ayrıca, CI/CD süreçlerinin optimize edilmesiyle, bu tür sistemlerin sürdürülebilirliğini artırmak ve hızlı bir şekilde değişen telekomünikasyon ihtiyaçlarına uyum sağlamalarını mümkün kılmak

hedeflenmektedir. Projenin tasarım aşamasında verilen her karar detaylı olarak işlenecek ve problemlere yaklaşımlar açısından farklı perspektifler sunulacaktır.

1.1.1. Projenin Amacı

Telekomünikasyon sektöründeki mevcut çevrimiçi ücretlendirme sistemleri (OCS), kullanıcıların artan ihtiyaçlarını karşılamak ve hizmet sağlayıcıların gelir yönetimini optimize etmek için kritik bir öneme sahiptir. Ancak, bu sistemlerde görülen performans, ölçeklenebilirlik ve sürekli entegrasyon süreçlerine ilişkin eksiklikler, hem müşteri memnuniyetini hem de hizmetlerin sürdürülebilirliğini olumsuz yönde etkilemektedir. Bu proje, yukarıda belirtilen sorunlara çözüm getirmeyi ve sektöre yenilikçi bir yaklaşım kazandırmayı amaçlamaktadır.

Projenin amacı, aşağıdaki temel hedefler doğrultusunda şekillenmektedir:

1. **Performansın İyileştirilmesi:**
   Kullanıcıların hizmet tüketimini izleyip yöneten, yüksek performanslı ve ölçeklenebilir bir OCS mimarisi oluşturmak.
2. **Sürekli Entegrasyon ve Sürekli Dağıtım (CI/CD) Süreçlerinin Etkin Yönetimi:**
   Karmaşık sistemlerin yazılım güncellemeleri ve dağıtım süreçlerini otomatikleştirerek, hata riskini azaltmak ve sistem değişikliklerini hızlı bir şekilde devreye almak.
3. **Esnek ve Modüler Mimari Tasarımı:**
   Farklı medya türlerinin (ses, SMS, veri) kolayca entegre edilebileceği ve sonradan genişletilebilir modüler bir yapı geliştirmek.
4. **Kullanıcı Deneyiminin İyileştirilmesi:**
   Kullanıcıların bakiye durumları ve hizmet tüketimi ile ilgili gerçek zamanlı bilgilendirme almasını sağlayarak müşteri memnuniyetini artırmak.
5. **Gelir Kaybını Önleyici Sistemler:**
   Aşım durumlarında kullanıcıları bilgilendiren ve gelir kaybını minimize eden özel modüller tasarlamak.

Bu proje, hem telekomünikasyon sektöründe çevrimiçi ücretlendirme sistemlerinin mevcut eksiklerini gidermek hem de sektörde referans oluşturabilecek modern bir çözüm geliştirmek amacıyla tasarlanmıştır. Sistemin, sektörel ihtiyaçlara göre uyarlanabilirliği ve gelecekteki yeniliklere açık bir mimari sunması beklenmektedir.

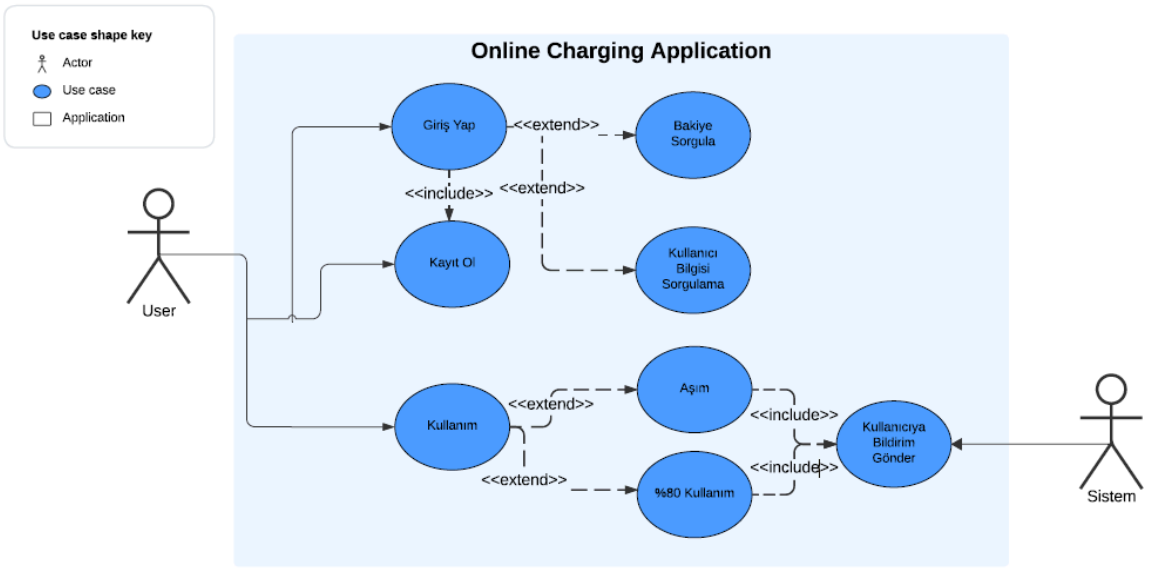
1.1.2. Projenin Kapsamı

Bu başlıkta da amacını detaylandırdığınız projenin neleri kapsayacağını, hangi özelliklere sahip olacağını yani projenizi detaylı olarak anlatmalısınız.

2. YAPILAN ÇALIŞMALAR

2.1. Planlama

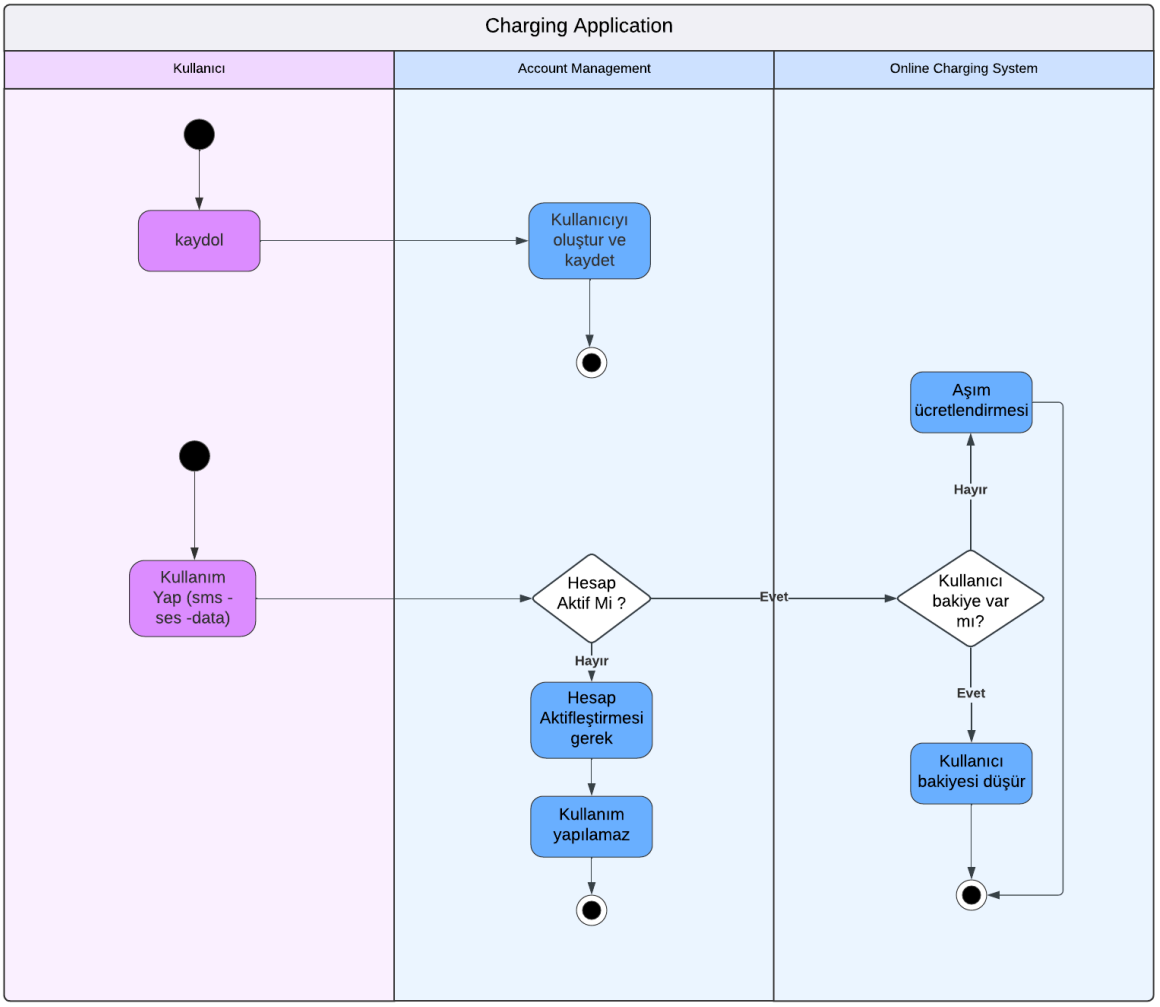
2.1.1. OCS Use Case Diyagramı



Use Case Diyagramının Açıklaması

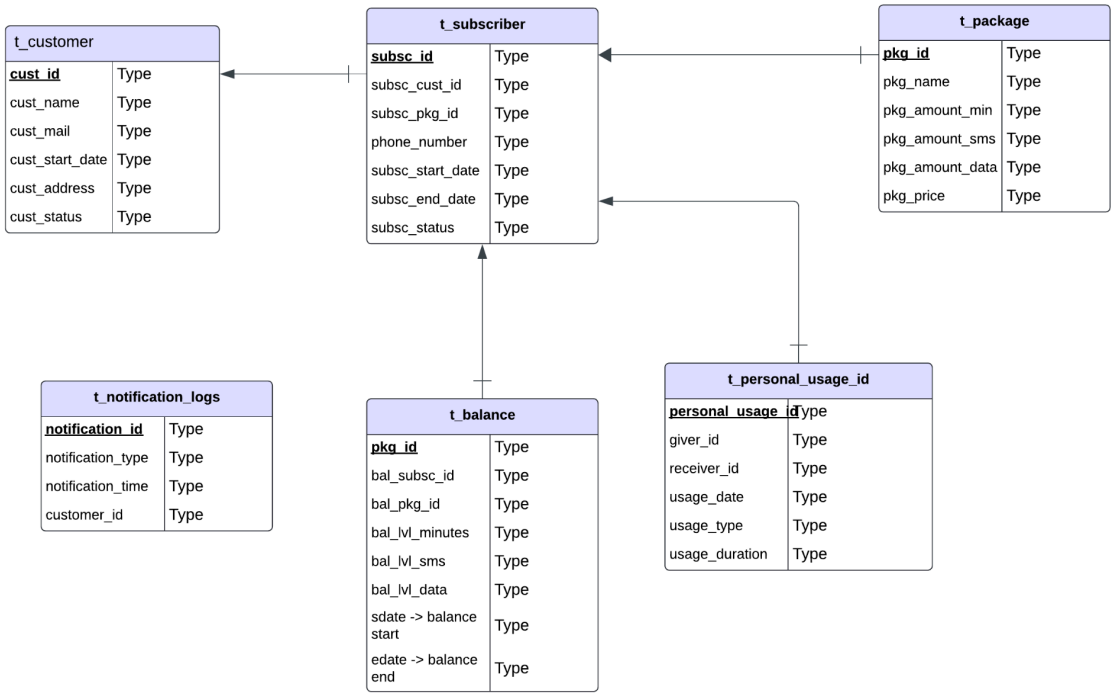
Bu **Use Case Diyagramı**, Online Charging System (OCS) projemizin temel işlevlerini ve bu işlevleri tetikleyen aktörleri görselleştirmek amacıyla tasarlanmıştır. Diyagram, sistemin işleyişini ve kullanıcı ile sistem arasındaki etkileşimleri tanımlayan senaryoları içermektedir.

2.1.2. Charging Application Aktivite Diyagramı



Bu aktivite diyagramı, **Charging Application** sistemindeki kullanıcı işlemlerini modellemektedir. Kullanıcı kaydolduktan sonra, hesap oluşturma ve doğrulama süreçleri gerçekleştirilir. Kullanıcı kullanım yapmadan önce hesap aktiflik durumu kontrol edilir. Eğer hesap aktif değilse, kullanıcı hesabını aktifleştirmek zorundadır. Aktif bir hesap varsa, bakiye durumu kontrol edilir. Yeterli bakiye bulunuyorsa işlem yapılır ve bakiye düşülür. Yetersiz bakiye durumunda ise aşım ücretlendirmesi devreye girer. Tüm bu süreçler, kullanımın tamamlanması veya işlemin durdurulmasıyla sonlanır. Bu diyagram, sistemin kullanıcı odaklı işlemlerini ve kontrol mekanizmalarını görselleştirmektedir.

2.1.3. Database Tasarımı



Bu veritabanı tasarımı, müşteriler, abonelikler, paketler, kullanım detayları, bakiye ve bildirimleri yönetmek için geliştirilmiştir.

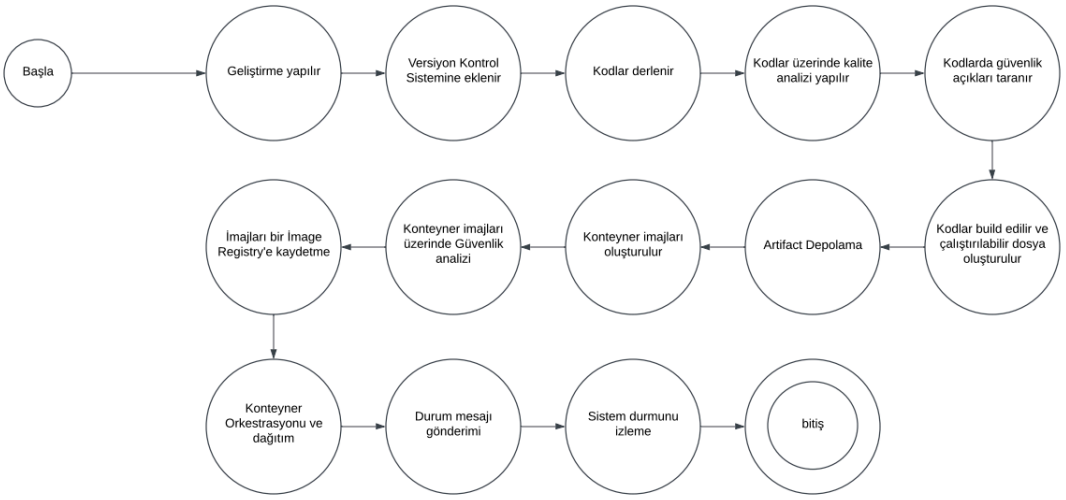
- **t_customer**: Müşteri bilgilerini saklar. (ID, ad, e-posta, başlangıç tarihi)
- **t_subscriber**: Müşterilerin telefon numaralarını ve paketlerini yönetir.
- **t_package**: Sunulan paketlerin ad, dakika, SMS ve veri limitlerini içerir.
- **t_balance**: Abonelik bazında kalan dakika, SMS ve internet bakiyelerini tutar.
- **t_personal_usage**: Kullanıcıların arama ve SMS detaylarını kaydeder.
- **t_notification_logs**: Müşterilere gönderilen bildirimlerin kaydını tutar.

Bu yapı, müşteri ilişkilerini ve kullanım detaylarını detaylı ve tutarlı bir şekilde yönetmek için tasarlanmıştır.

2.1.4. CI - CD Sürecinin Planlanması

2.1.4.1. Pipeline Akışının Planlanması

Durum Makinesi Diyagramı



Süreç Akış Diyagramı Açıklaması

Bu bölümde, yazılım geliştirme süreçlerinde **Sürekli Entegrasyon ve Sürekli Dağıtım (CI/CD)** yöntemlerini temel alan bir akış diyagramı açıklanmaktadır. Diyagramda, yazılımın geliştirilmesinden dağıtımına ve sistem izleme sürecine kadar olan adımlar detaylandırılmıştır. Süreç, yazılım projelerinde hatasız ve güvenilir bir geliştirme döngüsü sağlamak için tasarlanmıştır.

1. Geliştirme Yapılır

Yazılım geliştirme sürecinin başlangıç noktasıdır. Bu aşamada, geliştirme ekibi belirlenen gereksinimlere ve standartlara uygun olarak kodlama işlemini gerçekleştirir.

2. Versiyon Kontrol Sistemine Eklenir

Tamamlanan kod, bir versiyon kontrol sistemi (ör. Git) üzerine yüklenir. Bu, kodun sürüm kontrolünü sağlayarak iş birliği kolaylığı sunar ve geçmiş sürümlere dönülebilmesine olanak tanır.

3. Kodlar Derlenir

Yazılan kod, çalıştırılabilir bir hale getirilir. Derleme aşamasında, koddaki olası sözdizimsel hatalar tespit edilerek düzeltilir.

4. Kodlar Üzerinde Kalite Analizi Yapılır

Kodun okunabilirliği, bakım kolaylığı ve performans gibi kriterler değerlendirilir. Kalite analizi, yazılımın sürdürülebilirliğini artırmayı hedefler.

5. Kodlarda Güvenlik Açıkları Taranır

Kod, olası güvenlik açıklarını tespit etmek için taranır. Güvenlik analiz araçları, kodu tehditlere karşı koruyacak önlemleri belirler.

6. Kodlar Build Edilir ve Çalıştırılabilir Dosya Oluşturulur

Derlenen kod, çalıştırılabilir bir dosya haline getirilir (örn. JAR veya WAR). Bu dosya, yazılımın test edilmesi ve dağıtımı için kullanılır.

7. Artifact Depolama

Oluşturulan dosyalar ve diğer yazılım bileşenleri bir artifact deposuna yüklenir. Bu, dosyaların merkezi bir yerde saklanmasını ve yönetilmesini sağlar.

8. Konteyner İmajları Oluşturulur

Yazılım bileşenleri, Konteynerizasyon araçları kullanılarak bir konteyner imajına dönüştürülür. Bu adım, yazılımın farklı ortamlarda aynı şekilde çalışmasını garanti altına alır.

9. Konteyner İmajları Üzerinde Güvenlik Analizi

Oluşturulan konteyner imajları, güvenlik tarama araçlarıyla analiz edilir. Güvenlik açıkları bulunursa, düzeltici önlemler alınır.

10. İmajları Bir Image Registry'e Kaydetme

Güvenliği sağlanmış konteyner imajları, bir Image Registry'e yüklenir. Bu adım, imajların dağıtım için kullanılmasını sağlar.

11. Konteyner Orkestrasyonu ve Dağıtım

Konteyner imajları, orkestrasyon araçları ile dağıtılır. Bu adım, sistemin ölçeklenebilirliğini ve yönetilebilirliğini artırır.

12. Sistem Durumunu İzleme

İzleme araçları kullanılarak, sistemin performansı ve hataları izlenir. Bu süreç, sistemin kararlılığını sağlamak için kritik öneme sahiptir.

13. Durum Mesajı Gönderimi

İzleme araçlarından alınan sonuçlar doğrultusunda bildirimler oluşturulur. Durum mesajları, ilgili ekiplerle paylaşılır ve olası sorunlar hakkında hızlı aksiyon alınmasını sağlar.

2.1.4.2 Cloud Mimarisinin Planlanması

1. Cloud Mimarisinin Amacı

- Telekomünikasyon uygulamasının güvenli, ölçeklenebilir, yüksek performanslı ve sürekli erişilebilir bir altyapı üzerinde çalışmasını sağlamak.
- Kullanıcıların SMS, internet, arama gibi gerçek zamanlı verilerine kesintisiz erişim sunmak ve telekom operatörünün altyapısını destekleyen hizmetleri bulut üzerinden optimize etmek.
- Sistem bakım ve güncellemelerini kolaylaştırmak, daha düşük maliyetli ve hızlı bir şekilde yönetilebilir hale getirmek.

2. Başarı Ölçütleri

- Düşük gecikmeli veri akışı ve yüksek performans.
- Kullanıcı başına aylık en az %99,9 uptime garantisi.
- Artan kullanıcı yüküne göre sorunsuz ölçeklenme.
- Yedekli veri saklama ve hızlı kurtarma (disaster recovery).

3. Cloud Mimarisinin Hedefleri

Yüksek Ölçeklenebilirlik:

- Kullanıcı artışı ve veri yoğunluğuna göre dinamik ölçeklenebilir yapı.
- Coğrafi dağıtımlı erişim için CDN kullanımı.

Yüksek Performans:

- Gerçek zamanlı veri işleme ve düşük gecikmeli API yanıtları.

Güvenlik:

- TLS, AES gibi güçlü şifreleme yöntemleri ve WAF entegrasyonu.
- IAM ile erişim denetimi.

Yedeklilik ve Sürekli Kullanılabilirlik:

- Multi-region dağıtım ve otomatik hata kurtarma.
- %99.9 üzeri uptime garantisi.

Maliyet Etkinliği:

- Serverless ve autoscaling özellikleriyle kaynak optimizasyonu.
- Paylaşımlı altyapı ile maliyet düşürme.

Hızlı Entegrasyon:

- Telekom API'leri ile entegrasyon ve modüler yapı.

Veri Yönetimi ve Analitiği:

- Güvenli veri saklama ve büyük veri analitiği entegrasyonu.

Otomasyon ve DevOps Desteği:

- CI/CD süreçlerini destekleyen altyapı.
- Minimum kesintiyle güncelleme ve bakım.

Cloud Mimarisinin Temel Unsurları

1. Bulut Sağlayıcı Seçimi:

- AWS, Azure veya Google Cloud gibi sağlayıcılardan uygun olanı seçmek.

2. Servis Mimarisinin Planlanması:

- **Hesaplama Servisleri:** Sanal makineler, serverless işlevler, Kubernetes.
- **Veritabanı Servisleri:** SQL ve NoSQL veritabanları, cache çözümleri.
- **Depolama:** Nesne/dosya depolama, yedekleme ve arşivleme.
- **Ağ:** VPC, Load Balancer, CDN.
- **Güvenlik:** IAM, WAF, şifreleme.

3. Cloud Network Yapısı:

- **Ağ Tasarımı:** Public/Private subnet, NAT Gateway, VPN.
- **DNS Yönetimi:** Route 53 veya Azure DNS.

4. Depolama Sistemleri:

- **Anlık Veri:** NoSQL ve cache sistemleri.
- **Kalıcı Veri:** SQL tabanlı veritabanları.
- **Yedekleme:** Otomatik yedekleme ve geri yükleme stratejileri.
- **Şifreleme:** At-rest ve in-transit şifreleme.

5. Kubernetes ve Monitoring:

- Kubernetes tabanlı mikroservis yönetimi.
- AWS CloudWatch ve Azure Monitor gibi araçlarla izleme.

Bu mimari, ölçeklenebilir, güvenli ve performans odaklı bir altyapı sunarken telekom sistemleriyle entegrasyonu da kolaylaştırır.

3. Mobil Uygulamanın Geliştirilmesi

Mobil uygulamanın geliştirme süreci, kullanıcı gereksinimlerini karşılayan etkili bir ürün oluşturmayı hedefler.

1. Arayüz Tasarımı

- **Tasarım Araçları:**
  - Figma, Adobe XD veya Sketch gibi araçlar kullanılarak arayüz prototipi oluşturulur.
  - Renk paleti, tipografi, ikonlar ve diğer görsel ögeler belirlenir.
  - Kullanıcı deneyimini artıracak akışlar (user flow) ve ekranlar detaylandırılır.
- **Tasa**rım Süreci:
  - Wireframe: Temel ekran yapıları ve kullanıcı yolculuğu tasarlanır.
  - Mockup: Detaylı ve görsel olarak tamamlanmış tasarımlar hazırlanır.
  - Prototip: Ekranlar arasında etkileşim eklenerek bir demo oluşturulur.
  - Tasarım test edilerek, kullanıcı geri bildirimlerine göre iyileştirmeler yapılır.

2. Uygulamanın Geliştirilmesi

- **Programlama Dili ve Çerçeve Seçimi:**
  - Flutter veya React Native: Çapraz platform için tek bir kod tabanı kullanılır.
  - Swift (iOS) ve Kotlin/Java (Android): Yerel (native) uygulama geliştirme.
- **Backend Entegrasyonu:**
  - API'ler aracılığıyla telekomünikasyon sistemine bağlanarak kullanıcı ve kullanım verilerinin alınması.
  - Güvenli bir kimlik doğrulama (OAuth 2.0 veya JWT) entegrasyonu.
- **Geliştirme Süreci:**
  - Frontend Geliştirme: Kullanıcı arayüzlerinin kodlanması ve animasyonların eklenmesi.
  - Backend Entegrasyonu: Kullanıcı verilerinin ve API'lerin uygulamaya bağlanması.
  - Veri Yönetimi: Yerel veri depolama (SQLite, Hive) veya önbellekleme (Cache).
  - Test Süreci: Birim testleri, entegrasyon testleri ve kullanıcı kabul testleri ile uygulamanın hatasız çalışması sağlanır.

- **Sürüm ve Yayınlama:**
    - Uygulama, gerekli platform standartlarına uygun hale getirilir (App Store, Google Play).
    - Test sürecini başarıyla geçen uygulama, kullanıcılara sunulur.

2.2. İhtiyaç Analizi

2.2.1. Giriş

Online Charging System (OCS) projesinde, telekomünikasyon sektöründeki mevcut altyapının analiz edilmesi ve sektördeki modern gereksinimlerin karşılanması için kapsamlı bir ihtiyaç analizi yapılması gerekmektedir. Bu analiz, sistemin fonksiyonel ve fonksiyonel olmayan gereksinimlerini belirlemek için, kullanıcıların paket tüketimi, bakiye sorgulama, kullanım bildirimleri gibi süreçlerin detaylı şekilde incelenmesini içerir. Bu kapsamda, proje planlama aşamasında üretilen **use case diyagramları**, **süreç akış diyagramları** ve diğer görselleştirmeler, sistemin gereksinimlerini daha net bir şekilde ortaya koymaya yardımcı olacaktır. Özellikle telekomünikasyon sistemlerinde sürekli entegrasyon ve sürekli dağıtım (CI/CD) süreçlerinin etkili yönetimi, bu analizin önemli bir parçasını oluşturacaktır. Böylelikle, sistemin modüler yapısını destekleyen ve yüksek performans, esneklik ve ölçeklenebilirlik sağlayan bir çözüm geliştirilmesi hedeflenmektedir.

Bu bölümde, planlama aşamasında elde edilen çıktılar doğrultusunda, projenin gereksinimleri detaylandırılarak, sistemin hangi problemlere çözüm üreteceği ve bu çözümü nasıl gerçekleştireceği açıklanacaktır.

2.2.2. Gereksinimlerin Tanımlanması

Bu bölümde projede ele alınması gereken ihtiyaçlar listelenir ve sınıflandırılır.

2.2.2.1. Fonksiyonel Gereksinimler

Kullanıcı Yönetimi

- Kullanıcıların sisteme kayıt olabilmesi sağlanmalıdır.
    - Kayıt işlemi sırasında, kullanıcının kişisel bilgileri (ad, soyad, e-posta, telefon numarası) ve seçtiği tarife paketi alınmalıdır.
- Kullanıcılar, mevcut kullanıcı bilgilerini sorgulayabilmeli ve gerektiğinde güncelleyebilmelidir (örneğin, adres değişikliği).
- Kullanıcı, giriş yaparak sisteme erişim sağlayabilmelidir.

Paket ve Tarife Yönetimi

- Kullanıcılara farklı tarife ve paket seçenekleri sunulmalıdır (örneğin, ses, veri ve SMS paketleri).
- Kullanıcı, mevcut paketlerini sorgulayabilmeli ve yeni paketler satın alabilmelidir.

- Paket kullanım detayları (örneğin, kalan dakika, veri miktarı) anlık olarak görüntülenebilmelidir.

Bakiye Yönetimi

- Kullanıcının bakiyesi gerçek zamanlı olarak yönetilmelidir.
- Sistem, kullanıcının mevcut bakiyesini sorgulamasına olanak sağlamalıdır.
- Kullanıcı, mevcut bakiyesini aşması durumunda bildirim almalıdır ve ek bakiye yükleyebilmelidir.

Kullanım Yönetimi

- Kullanıcı, belirli hizmetleri (örneğin, sesli arama, veri kullanımı, SMS gönderimi) başlatabilmelidir.
- Kullanıcının tüketim miktarları (örneğin, gönderilen SMS sayısı, kullanılan veri miktarı) kaydedilmelidir.
- Sistem, kullanıcıya gerçek zamanlı olarak kullanım durumunu bildirebilmelidir.

Limit ve Aşım Yönetimi

- Sistem, kullanıcıya paket veya bakiyesinin %80'ine ulaşıldığında bildirim göndermelidir.
- Kullanıcı bakiyesini tamamen tükettiğinde, aşım durumunu belirterek kullanıcıya bildirim gönderilmelidir.
- Aşım durumunda, kullanıcıya ek paket veya hizmet satın alma seçenekleri sunulmalıdır.

Faturalama ve Bildirim Yönetimi

- Sistem, her ayın sonunda kullanıcıya fatura bilgilerini göndermelidir.
- Kullanıcıya aşım durumu, bakiye tükenmesi ve kullanım detayları gibi konularda bildirimler gönderilmelidir.
- Bildirimler e-posta, SMS veya mobil uygulama üzerinden gönderilebilmelidir.

Gerçek Zamanlı Ücretlendirme

- Kullanıcının hizmet kullanımı sırasında gerçek zamanlı olarak ücretlendirme yapılmalıdır.
- Kullanıcının kullandığı her bir birim (örneğin, 1 MB veri, 1 SMS, 1 dakika arama) ücretlendirme sistemi üzerinden değerlendirilmelidir.
- Ücretlendirme, sistemde tanımlı tarifelere ve paketlere göre hesaplanmalıdır.

Entegrasyon ve Veri Senkronizasyonu

- Kullanıcının kullanım bilgileri, bakiye ve paket durumları senkronize edilmelidir.
- kullanıcı fatura ve kullanım bilgilerini faturalama sistemine aktarmalıdır.
- Bildirimler ve aşım durumları ilgili yerlere iletilmelidir.

Güvenlik ve Veri Gizliliği

- Kullanıcı hesap ve kullanım bilgileri güvenli bir şekilde saklanmalı ve yalnızca yetkilendirilmiş kişiler tarafından erişilebilir olmalıdır.

- Veri güvenliği için şifreleme ve erişim kontrol mekanizmaları uygulanmalıdır.

Sistem Performansı ve Ölçeklenebilirlik

- Sistem, aynı anda birden fazla kullanıcıdan gelen talepleri karşılayacak şekilde ölçeklenebilir olmalıdır.
- Yüksek veri trafiği altında gerçek zamanlı performans sağlamalıdır.

Paket ve Hizmet Entegrasyonu

- Sistem, farklı medya türleri (örneğin, ses, SMS, veri) için entegre bir hizmet sunmalıdır.
- Yeni hizmet veya paketlerin sisteme eklenmesi kolaylaştırılmalıdır.

Android Uygulama Arayüzü

1.Kullanıcı Hesap Yönetimi

- Kullanıcılar hesap oluşturabilmelidir.
- Kullanıcılar mevcut hesaplarıyla giriş yapabilmelidir.
- Kullanıcılar şifrelerini unutmaları durumunda şifre sıfırlama talebinde bulunabilmelidir.

2.Paket ve Fatura Yönetimi

- Kullanıcılar, mevcut internet, dakika ve mesaj haklarını görüntüleyebilmelidir.
- Kullanıcılar, paket kullanım detaylarını yüzdesel olarak görebilmelidir.
- Kullanıcılara, fatura detayları sunulmalıdır.

3.Bildirim ve Uyarılar

- Kullanıcılar, paket kullanım oranları kritik seviyeye ulaştığında bilgilendirilebilmelidir.

2.2.2.2. Fonksiyonel Olmayan Gereksinimler

1. Performans Gereksinimleri

Projenin performans gereksinimleri, sistemin hız, ölçeklenebilirlik ve güvenilirlik açısından modern telekomünikasyon ihtiyaçlarını karşılayabilecek bir altyapı sunmayı hedeflemektedir. Aşağıda, diyagramda yer alan bileşenler göz önüne alınarak, kullanılan teknolojilerin kategorileri üzerinden belirlenen performans gereksinimleri açıklanmıştır:

1. Mesaj Kuyruğu

- **Gereksinim**: Sistem, modüller arasında veri aktarımı için mesaj kuyruğu yapısını kullanacaktır. Bu yapı sayesinde, mesajların yüksek hızda, güvenilir bir şekilde iletilmesi sağlanacaktır.
- **Ölçeklenebilirlik**: Mesaj kuyruğu, eşzamanlı binlerce mesajı işleyebilecek şekilde yapılandırılmalı ve yüksek yük altında da performans kaybı yaşamamalıdır.

2. Ön Bellekleme

- **Gereksinim**: Kullanıcı verilerinin düşük gecikmeli erişimi için bir ön bellekleme mekanizması kullanılacaktır. Bu yapı, sık erişilen verilere hızlı erişim sağlamak amacıyla tasarlanacaktır.
- **Güvenilirlik**: Ön bellekteki verilerin tutarlılığı sürekli olarak sağlanmalı ve olası sistem arızalarında veritabanı ile eşzamanlı bir şekilde çalışabilmelidir.

3. Dağıtılmış Sistem Yönetimi

- **Gereksinim**: Dağıtılmış bir sistem yapısı ile yük dengelemesi ve ölçeklenebilirlik sağlanmalıdır. Sistem, modüller arasında yük paylaşımı yaparak performans artışı sağlamalıdır.
- **Yedeklilik ve Güvenilirlik**: Sistem, olası bir modül hatasında diğer modüllerin çalışmasını etkilemeyecek şekilde tasarlanmalıdır.

4. Veritabanı Yönetimi

- **Gereksinim**: Kullanıcı ve hesap bilgileri için güvenilir bir veritabanı yönetim sistemi kullanılacaktır. Bu yapı, hem okuma hem de yazma işlemlerinde yüksek performans sağlamalıdır.
- **Veri Tutarlılığı**: Anlık işlem yapılan verilerin tutarlılığı garanti edilmelidir.

5. Sürekli Entegrasyon ve Sürekli Dağıtım (CI/CD) Yönetimi

Gereksinim: Sistem, geliştirilen yazılımların güvenilir ve hızlı bir şekilde üretim ortamına aktarılmasını sağlayacak bir CI/CD altyapısına sahip olmalıdır. CI/CD süreçleri, yazılım geliştirme sürecindeki kod değişikliklerinin sürekli olarak test edilmesi, birleştirilmesi ve dağıtılması işlevlerini yerine getirmelidir.

Performans Hedefi:

- Kod değişikliklerinin üretim ortamına aktarılması sürecinin toplam süresi 5 dakika ile sınırlı olmalıdır.
- Otomatik testlerin, %95 başarı oranı ile çalışması ve minimum manuel müdahale gerektirmesi sağlanmalıdır.

Ölçeklenebilirlik ve Hız:

- CI/CD altyapısı, artan geliştirici sayısı ve değişiklik frekansı ile uyumlu olacak şekilde ölçeklenebilir olmalıdır.
- Paralel dağıtım süreçleri desteklenerek aynı anda birden fazla modülün güncellenmesi sağlanmalıdır.

Güvenilirlik:

- CI/CD süreçleri sırasında oluşabilecek hatalar anında raporlanmalı ve sistem geri dönüşü (rollback) için otomatik mekanizmalar bulunmalıdır.
- Tüm dağıtım süreçleri boyunca yazılım versiyonları izlenebilir olmalı ve herhangi bir hata durumunda bir önceki stabil versiyona dönülebilmelidir.

Avantajları:

- Daha hızlı dağıtım süreçleri ile sistemin üretkenliğinin artırılması.
- Geliştirme ekipleri arasında koordinasyonu sağlayarak yazılım değişikliklerinin entegre edilme süresinin kısaltılması.
- Yeni özelliklerin ve hata düzeltmelerinin kullanıcıya daha hızlı ulaşmasını sağlayarak müşteri memnuniyetinin artırılması.

CI/CD yönetimi, yalnızca sistemin geliştirme ve dağıtım süreçlerini optimize etmekle kalmaz, aynı zamanda genel sistem performansını ve güvenilirliğini de artırarak sürekli güncel bir altyapı sunar. Bu, özellikle telekomünikasyon gibi dinamik bir sektörde önemli bir gereksinimdir.

2. Teknik Gereksinimler

Teknik gereksinimler, sistemin altyapısının ve çalışma ortamının doğru bir şekilde yapılandırılmasını sağlamak için belirlenmiştir. Projede kullanılan donanım, yazılım ve bulut hizmetleri gibi bileşenlerin gereksinimleri aşağıda açıklanmıştır.

Donanım Gereksinimleri

Sistem altyapısı, ölçeklenebilirlik ve yüksek performans gereksinimlerini karşılayabilmek için Amazon Web Services (AWS) bulut platformu üzerinde çalışacaktır.

AWS ile Sağlanan Donanım:

- EC2 Sunucuları: Mikroservislerin çalıştırılması ve yük paylaşımı için elastik bulut sunucuları kullanılacaktır.
- Elastic Load Balancer (ELB): Trafik yönetimi ve yük dengeleme için kullanılacaktır.

Yazılım Gereksinimleri

OCS Mikroservis Gereksinimleri

Sistem altyapısında kullanılan yazılımlar ve entegrasyon gereksinimleri şunlardır:

- **Mesaj Kuyruğu:**
  Apache Kafka, sistem modülleri arasında veri iletişimi için kullanılacaktır.
  - **Özellikler:**
    - Her mesaj kuyruğu için 3 replikalı bir yapı sağlanacaktır.
    - Yüksek trafik durumlarında gecikmeyi minimumda tutacak konfigürasyon.
- **Ön Bellekleme:**
  Hazelcast, sık erişilen verilerin hızlı erişimi için tercih edilecektir.

    - **Özellikler:**
        - Bellek tabanlı işlem desteği.
        - Dağıtılmış yapı ile ölçeklenebilirlik.
- **Veritabanı Yönetimi:**
  Oracle Veritabanı, kullanıcı ve hesap bilgilerinin güvenli bir şekilde depolanması için kullanılacaktır.
    - **Özellikler:**
        - ACID uyumluluğu ve yüksek tutarlılık.
        - Tablolar arası ilişkisel yapı desteği.
- **Dağıtık Veri İşleme ve Cache**
  Yazılım: Apache Ignite, yüksek performanslı ve gerçek zamanlı veri işleme için kullanılacaktır.

    - **Özellikler**:
        - Bellek içi veri işleme (in-memory computing) desteği.
        - Veri tutarlılığı ve yatay ölçeklenebilirlik.
        - Dağıtık ön bellek (distributed cache) mekanizması ile performans artışı.
        - Oracle veritabanı ile kolay entegrasyon ve veri eşzamanlılığı.
        - Gerçek zamanlı analiz işlemleri için işlem gücü sağlama.

CI-CD Süreç Gereksinimleri

1. Kod Yönetimi

- **Gereksinim:** Geliştiricilerin yazdığı kodların güvenli bir şekilde saklanması ve sürüm kontrolünün sağlanması için merkezi bir kod yönetim platformuna ihtiyaç vardır.
- **Amaçlar:**
    - Kod versiyon kontrolü yapılması.
    - Kod incelemeleri ve kalite artırımı için işbirlikçi bir mekanizma sağlanması (örneğin, kod birleştirme talepleri).

2. Sürekli Entegrasyon ve Dağıtım Otomasyonu

- **Gereksinim:** Sürekli entegrasyon ve sürekli dağıtım süreçlerini otomatikleştirecek merkezi bir araç gereklidir.
- **Teknik Özellikler:**
    - Süreçlerin esnek ve ölçeklenebilir şekilde yapılandırılması.
    - Derleme, test ve dağıtım işlemlerinin zamanlanabilir ve otomatik olarak tetiklenebilir olması.
- **Performans Gereksinimi:**
    - Artan yük altında bile işlemlerin gecikmesiz şekilde çalışması.

3. Kod Derleme ve Test

- **Gereksinim:** Uygulama kodlarının yapılandırılması, derlenmesi ve birim testlerinin gerçekleştirilmesi.
- **Teknik Özellikler:**
    - Derleme sırasında hataların minimize edilmesi.
    - Kodun en az %90 birim test kapsamına ulaşması.

4. Kod Kalitesi ve Güvenlik Analizi

- **Gereksinim:** Kodun kalitesinin artırılması ve olası güvenlik açıklarının tespit edilmesi için bir analiz süreci gereklidir.
- **Teknik Özellikler:**
    - Kod analizinde hata oranının %5'in altında tutulması.
    - Güvenlik açıklarının otomatik tespiti ve raporlanması.

5. Paket Yönetimi

- **Gereksinim:** Derleme sonrası oluşturulan uygulama paketlerinin merkezi bir depoda saklanması ve kolay erişilebilir olması.
- **Teknik Özellikler:**
    - Paketlerin güvenli ve hızlı bir şekilde yüklenmesi ve indirilmesi.
    - Versiyonlama desteği ile farklı sürümlerin yönetimi.

6. Uygulama İzolasyonu

- **Gereksinim:** Uygulamaların bağımsız ve taşınabilir ortamlarda çalıştırılması için bir izolasyon çözümüne ihtiyaç vardır.
- **Teknik Özellikler:**
    - Çalıştırılabilir görüntülerin oluşturulması ve optimize edilmesi.
    - Güvenlik taramalarının entegre edilmesi.

7. İzolasyon Ortamlarının Yönetimi

- **Gereksinim:** Bağımsız çalışan ortamların yönetimi ve ölçeklenebilir bir altyapı oluşturulması gereklidir.
- **Teknik Özellikler:**
    - Ortamların otomatik olarak dağıtılması ve ölçeklenmesi.
    - Hatalı çalışan ortamların tespit edilmesi ve yeniden başlatılması.

8. Sistem İzleme ve Gözlemleme

- **Gereksinim:** Sistem performansını ve durumunu gerçek zamanlı izlemek için bir izleme çözümüne ihtiyaç vardır.
- **Teknik Özellikler:**
    - Performans metriklerinin görselleştirilmesi.
    - Anlık bildirimlerle sorunların hızlı bir şekilde fark edilmesi.

9. Bildirim ve Geri Bildirim

- **Gereksinim:** Geliştirme ve dağıtım süreçleriyle ilgili geri bildirimlerin hızlı bir şekilde ekiplere iletilmesi.
- **Teknik Özellikler:**
    - Başarılı veya başarısız işlemlerle ilgili otomatik raporlama.
    - Geliştiricilere gerçek zamanlı bildirimlerin iletilmesi.

10. Dağıtım Süreci

- **Gereksinim:** Üretilen uygulama paketlerinin hatasız ve otomatik bir şekilde çalışır ortamlara dağıtılması.

- **Teknik Özellikler:**
    - Süreçlerin otomatikleştirilmesi ve minimum manuel müdahale ile tamamlanması.
    - Farklı dağıtım stratejilerinin (örneğin, mavi/yeşil dağıtım) desteklenmesi.

Bu gereksinimler, CI/CD sürecinin güvenilir, hızlı ve otomatik bir şekilde gerçekleştirilmesini sağlayacak şekilde tasarlanmıştır. Teknik gereksinimlerin sağlanması, sistemin ölçeklenebilirliğini artıracak ve geliştirme süreçlerinde zamandan tasarruf sağlayacaktır.

Cloud Mimarisi Gereksinimleri

1. Ağ Yapısı ve Trafik Yönetimi

- **Kullanıcı Trafiği Yönetimi:**
    - Kullanıcılardan gelen tüm taleplerin optimize edilmesi ve doğru hedefe yönlendirilmesi için bir trafik yönlendirme katmanına ihtiyaç vardır.
    - Trafik şifreleme ve güvenliği için bir güvenlik sertifikası yönetim sistemine gereksinim duyulabilir.
- **Yük Dengeleme:**
    - Trafiğin dengeli bir şekilde dağıtılması, sistem performansının artırılması ve ölçeklenebilirlik sağlanmalıdır.
    - Sistem, yük altındayken bile kesintisiz hizmet verebilmelidir.

2. Kapsayıcılaştırma ve Mikroservis Yönetimi

- **Servislerin Yönetimi:**
    - Servislerin kapsayıcı bazlı yönetimi yapılmalıdır.
    - Kapsayıcılar için bir orkestrasyon aracı gereklidir (örneğin, bir küme yönetimi aracı).
- **Bağımsız Çalışma:**
    - Her servis birbirinden bağımsız çalışabilmeli ve ölçeklenebilir olmalıdır.

3. Depolama ve Veri Yönetimi

- **Dosya Depolama:**
    - Kullanıcı verileri ve uygulama kaynaklarının ölçeklenebilir bir depolama sistemine ihtiyacı vardır.
- **Veritabanı İhtiyacı:**
    - Yapısal (relational) ve yapısal olmayan (NoSQL) veri için uygun bir veritabanı yapısı gereklidir.
    - Veritabanı, düşük gecikme süresi ve yüksek performans sağlayacak şekilde optimize edilmelidir.

4. Yedeklilik ve Yüksek Erişilebilirlik

- **Erişim Sürekliliği:**
    - Sistem, kesinti anında hizmeti sürdürebilmek için birden fazla veri merkezi arasında yedekleme yapmalıdır.

    - Bir yedekleme ve geri yükleme mekanizması uygulanmalıdır.
- **Felaket Kurtarma:**
    - Olası veri kaybına karşı, sistemin hızlı bir şekilde kurtarılmasını sağlayacak bir plan olmalıdır.

5. Güvenlik

- **Veri Güvenliği:**
    - Uygulama, ağ ve veri tabanı güvenliği için şifreleme, erişim kontrolü ve güvenlik duvarları gibi önlemler alınmalıdır.
- **VPN ve Güvenli Erişim:**
    - Özel ağda çalışan bileşenlere yalnızca belirli kullanıcıların güvenli bir şekilde erişmesi sağlanmalıdır.

6. Ağ ve İletişim

- **Public ve Private Ağ Ayrımı:**
    - Sistem, hem genel (kullanıcıya açık) hem de özel (dahili servisler için) ağlarda çalışmalıdır.
- **İnternet Geçişi:**
    - Özel ağdaki servislerin internetle iletişimi kontrollü bir şekilde sağlanmalıdır.

7. Performans ve Ölçeklenebilirlik

- **Dinamik Ölçekleme:**
    - Trafiğe göre sistem kaynaklarının otomatik olarak artırılması veya azaltılması gereklidir.
- **Yük Testi:**
    - Yüksek kullanıcı trafiği altında sistemin nasıl davranacağını test eden bir performans analiz aracı kullanılmalıdır.

8. Dağıtık Sistemler ve Veri İşleme

- **Veri Senkronizasyonu:**
    - Dağıtık veri işleme ve servislerin uyumlu çalışması için bir mekanizma gereklidir.
- **Kuyruklama ve Mesajlaşma Sistemi:**
    - Servisler arası asenkron iletişim sağlamak için bir kuyruklama/mesajlaşma aracı uygulanmalıdır.

9. İzleme ve Loglama

- **Gerçek Zamanlı İzleme:**
    - Sistem performansını, kaynak kullanımını ve hata oranlarını izlemek için bir izleme platformu gereklidir.
- **Log Yönetimi:**
    - Tüm servislerden gelen logların merkezi bir şekilde toplanması ve analiz edilmesi için bir araç kullanılmalıdır.

Android Uygulama Gereksinimleri

- Uygulama hem Android platformlarında çalışmalıdır.
- Uygulama, çevrimdışı erişim için temel verileri önbelleğe alabilmelidir.

2.2.3 Kullanıcı Beklentileri

Bu bölümde, sistemin son kullanıcılar tarafından nasıl kullanılacağı ve karşılaması gereken temel ihtiyaçlar açıklanmıştır. Kullanıcıların beklentileri, projenin işlevsel gereksinimlerini şekillendiren önemli unsurlardır. Telekomünikasyon sektöründeki müşterilerin deneyimlerini iyileştirmek ve operatörlere değer katmak amacıyla aşağıdaki beklentiler belirlenmiştir:

1. **Anlık Bakiye Görüntüleme**
    - Kullanıcılar, mevcut bakiyelerini ve hizmet paketlerini gerçek zamanlı olarak görüntüleyebilmelidir.
    - Bakiye bilgisi, mobil uygulama veya web portalı üzerinden kolayca erişilebilir olmalıdır.
2. **Limit Aşım Bildirimleri**
    - Kullanıcılar, veri veya hizmet limitlerinin %80'ine ulaşıldığında ve limit aşıldığında bildirim almalıdır.
    - Bu bildirimler SMS veya e-posta yoluyla hızlı bir şekilde iletilmelidir.
3. **Hizmet Kullanımı ve Paket Yönetimi**
    - Kullanıcılar, mevcut paketlerini kontrol edebilmeli, yeni paket satın alımı yapabilmeli ve paket detaylarını görebilmelidir.
    - Hizmet kullanım bilgileri (örneğin, dakika, SMS, veri tüketimi) detaylı bir şekilde sunulmalıdır.
4. **Kullanıcı Odaklı Arayüz**
    - Sistem, kullanıcı dostu bir arayüzle tasarlanmalı ve kolay kullanım sağlamalıdır.
    - Arayüz, kullanıcıların hızlı ve zahmetsiz bir şekilde işlem yapmalarını mümkün kılmalıdır.
5. **Güvenilirlik ve Kesintisizlik**
    - Sistem, kullanıcıların işlemlerini kesintisiz bir şekilde gerçekleştirebileceği bir altyapı sunmalıdır.
    - Veri tutarlılığı ve hizmet sürekliliği, kullanıcı deneyimini artırmak için sağlanmalıdır.
6. **Bildirimlerin Hızlı İletilmesi**
    - Kullanıcılar, sistemden gelen bildirimlerin hızlı ve doğru bir şekilde iletilmesini beklemektedir.
    - Fatura bildirimleri, paket yenileme uyarıları ve bakiye yükleme onayları gecikmesiz olarak ulaşmalıdır.
7. **Esnek Paket Seçenekleri**
    - Kullanıcılar, kendi ihtiyaçlarına uygun esnek ve özelleştirilebilir paket seçeneklerini inceleyebilmeli ve satın alabilmelidir.
    - Sistem, farklı hizmet türlerine yönelik kombinasyonlar sunmalıdır.
8. **Veri Güvenliği**
    - Kullanıcıların hesap bilgileri ve hizmet kullanım detayları, yüksek güvenlik standartlarına uygun şekilde korunmalıdır.
    - Kullanıcılar, verilerinin yetkisiz erişimlere karşı korunduğundan emin olmalıdır.

9. **Kullanıcı Dostu Arayüz**

    - Uygulama, kullanıcı dostu bir arayüze sahip olmalıdır.
    - Tüm işlemler basit ve anlaşılır bir şekilde yapılabilmelidir.

Sonuç olarak, sistemin tasarımında ve geliştirilmesinde kullanıcı odaklı bir yaklaşım benimsenmiş ve yukarıdaki beklentilerin karşılanması hedeflenmiştir. Bu, hem müşteri memnuniyetini artıracak hem de operatörlerin operasyonel verimliliğini destekleyecektir.

2.2.4. Riskler ve Kısıtlamalar

Bu projede karşılaşılan kısıtlar ve olası çözüm önerileri aşağıda detaylı olarak açıklanmıştır.

1. Bağımlılıklar

- **Veritabanı Bağımlılığı:**
    - **Kısıt:** Mikroservis mimarisinin ideal uygulamalarında her modül kendi veritabanına sahip olmalıdır. Ancak bu projede tüm modüller Oracle veritabanına bağımlıdır. Bu durum, bir modülün başarısız olması durumunda diğer modüllerin etkilenmesine yol açabilir.
    - **Açıklama:** ABMF modülü, Oracle veritabanı ile çalışarak hesap bakiyelerini yönetir. Ancak Oracle veritabanı, AOM modülü tarafından doldurulmaktadır. Bu durum, ABMF'in bağımsız çalışmasını engelleyebilir ve AOM'un hesap yönetimi dışında ek sorumluluklar almasına neden olabilir. Modüller arası bu tür bir bağımlılık, mikroservislerin esnekliğini ve bağımsızlığını sınırlar.
    - **Çözüm Önerisi:** Her modül için ayrı veritabanları oluşturulabilir. Ancak bu durumda veritabanı tutarsızlıklarını önlemek ve normalizasyonu sağlamak için güçlü senkronizasyon mekanizmaları geliştirilmesi gereklidir.

- **Mesajlaşma Sistemi Bağımlılığı:**
    - **Kısıt:** Modüllerin Kafka mesaj kuyruğuna olan bağımlılığı, mikroservis mimarisinde ideal olmayan bir durum oluşturabilir. Çünkü mikroservislerin birbirine ve merkezi bir mesaj kuyruğuna olan bağımlılıkları, tasarım esnekliğini azaltabilir.
    - **Açıklama:** Örneğin, bir modül Kafka'nın çalışmaması durumunda işlevselliğini sürdüremeyebilir. Bu durum, tüm sistemi olumsuz etkileyebilir.
    - **Çözüm Önerisi:** Modüllerin doğrudan iletişim yerine olay odaklı bir mimariyle Kafka'ya bağımlılıkları azaltılabilir. Alternatif olarak, farklı mesaj kuyruğu sistemlerine geçiş yapılabilecek şekilde bir soyutlama katmanı eklenebilir.

2. Mikroservis Mimarisine Aykırılıklar

- **Fonksiyonel Çakışmalar:**
    - **Kısıt:** ABMF'in işlevinin AOM'a kaydırılması, AOM'un yalnızca "Account Order Management" işlevini yerine getiren bir modül olma rolünü karmaşıklaştırabilir. Bu durum, modülün görevlerinin birbirine karışmasına yol açabilir.

    - **Açıklama:** Mikroservislerin bağımsızlığı ve net işlev sınırları, modülerlik ve kolay bakım için önemlidir. Ancak bu tür görev karmaşası, kod karmaşıklığını artırabilir ve bakım maliyetlerini yükseltebilir.
    - **Çözüm Önerisi:** Modüllerin görevlerini net şekilde ayrıştırmak için detaylı bir görev analizi yapılmalı ve gerektiğinde mikroservis sınırları yeniden tanımlanmalıdır.

3. Teknik Altyapı Kısıtlamaları

- **Kafka Performansı:**
    - **Kısıt:** Kafka, yüksek trafikte darboğaz oluşturabilir. Modüllerin Kafka'ya yüksek bağımlılığı, Kafka'nın performansı ile sınırlı bir sistem yapısı oluşturabilir.
    - **Çözüm Önerisi:** Kafka'nın performansını artırmak için paralel işlem hattı yapılandırmaları uygulanabilir ve yedekli bir Kafka kümesi oluşturulabilir.
- **Oracle Veritabanı Performansı:**
    - **Kısıt:** Merkezi bir Oracle veritabanı kullanımı, artan yükle birlikte performans sorunlarına yol açabilir.
    - **Çözüm Önerisi:** Oracle için okuma/yazma yüklerini optimize edecek indeksleme stratejileri ve yedekli yapılandırmalar uygulanabilir.

4. Geliştirme ve Dağıtım Kısıtlamaları

- **CI/CD Süreçleri:**
    - **Kısıt:** Merkezi bileşenler üzerindeki bağımlılık nedeniyle CI/CD süreçlerinin düzgün çalışmaması riski bulunmaktadır.
    - **Çözüm Önerisi:** CI/CD süreçleri, bağımsız modüllerin ayrı dağıtımını destekleyecek şekilde tasarlanmalıdır. Ayrıca, hata izleme ve otomatik iyileştirme mekanizmaları ile desteklenmelidir.

5. Genel Riskler

- **Ekip Bilgi Seviyesi:**
    - Projede kullanılan teknolojilere (Kafka, Ignite, Hazelcast) ekip üyelerinin aşinalık düzeyi sınırlı olabilir.
    - Çözüm: Gerekli eğitimler düzenlenerek ekip üyelerinin bilgi seviyeleri artırılmalıdır.
- **Yük Dengelemesi ve Trafik Yönetimi:**
    - **Kısıt:** Dağıtık yapıdaki modüller arasında yük dengesi doğru sağlanmazsa, bazı modüllerde darboğazlar oluşabilir.
    - **Çözüm Önerisi:** Trafik yönetimini sağlamak için yük dengeleme (load balancer) mekanizmaları uygulanabilir.
- **Veri Senkronizasyonu:**
    - **Kısıt:** Farklı önbellekler ve veritabanları arasında veri senkronizasyonunun sağlanması zor olabilir.
    - **Çözüm Önerisi:** Veri tutarlılığını sağlayacak düzenli arka plan senkronizasyon işlemleri uygulanmalıdır.

6. Proje Bağımlılıkları

- **Üçüncü Parti Araçlara Bağımlılık:**

- **Kısıt:** Kafka, Ignite, Hazelcast ve Oracle gibi dış kütüphaneler veya hizmetlerin herhangi birinde oluşacak kesinti tüm sistemi etkileyebilir.
- **Çözüm Önerisi:** Üçüncü parti araçlara yedeklilik eklenmeli ve alternatif çözümler bulundurulmalıdır.

2.2.5. Sonuç

Yapılan ihtiyaç analizi, **Online Charging System (OCS)** projesinin modern telekomünikasyon sektörünün ihtiyaçlarını karşılayacak şekilde tasarlanmasını sağlamıştır. Sistem, kullanıcıların anlık bakiye sorgulama, limit aşımı bildirimleri ve hizmet yönetimi gibi temel beklentilerini karşılamayı hedeflemektedir. Ayrıca, altyapının modüler yapısı sayesinde esneklik ve ölçeklenebilirlik sağlanacaktır.

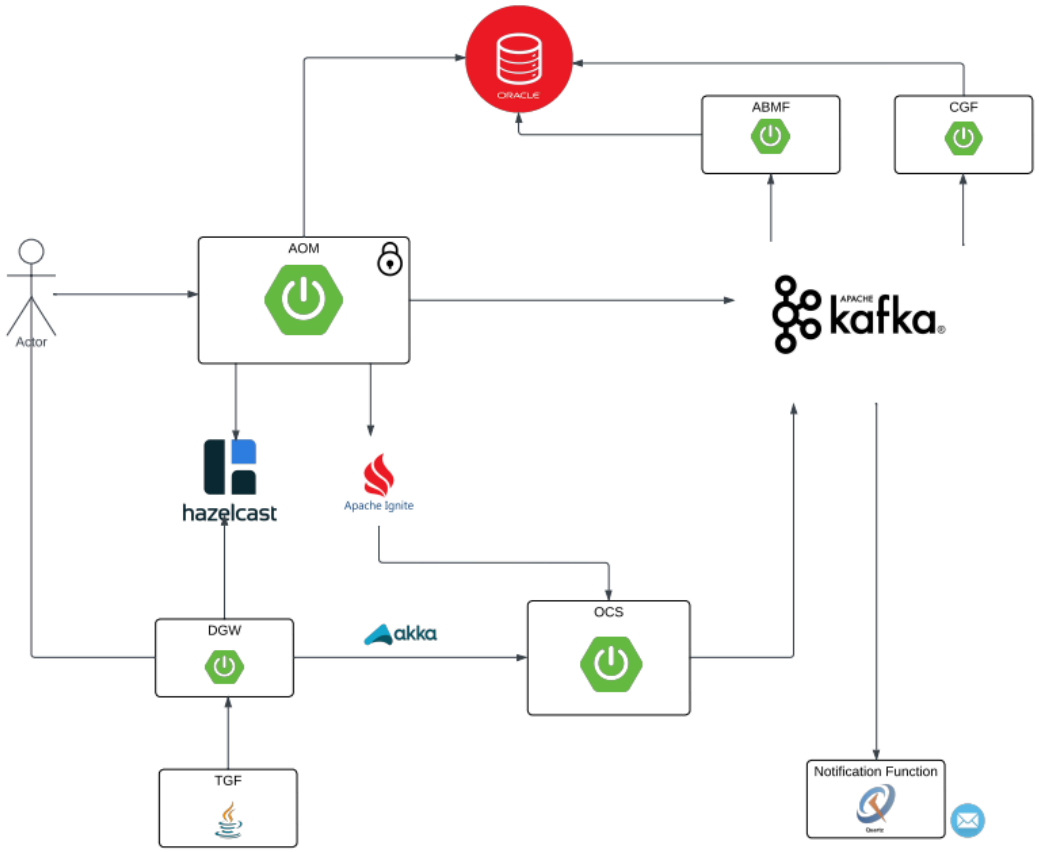
Performans gereksinimlerinin belirlenmesiyle, mesaj kuyruğu, ön bellekleme ve dağıtık sistem yönetimi gibi modern teknolojilerden faydalanarak, yüksek hızlı ve güvenilir bir sistem oluşturulması hedeflenmiştir. Teknik gereksinimler kapsamında AWS altyapısı, Apache Kafka, Hazelcast ve Oracle gibi bileşenler, sistemin performansını artırmak ve sürdürülebilir bir yapı sağlamak için seçilmiştir.

Proje sürecinde karşılaşılan bağımlılıklar ve teknik sınırlamalar, dikkatli bir planlama ve kapsamlı test süreçleriyle ele alınacaktır. Bu bağlamda, mikroservis mimarisinin esnekliğinden faydalanılarak, veri senkronizasyonu ve CI/CD süreçlerinin iyileştirilmesi sağlanacaktır.

Sonuç olarak, ihtiyaç analizi, sistemin performans, güvenilirlik ve kullanıcı memnuniyeti hedeflerine ulaşması için gerekli altyapıyı belirlemiş ve projenin sağlam bir temele oturmasını sağlamıştır. Bu sistemin, modern telekomünikasyon hizmetleri için güçlü ve yenilikçi bir çözüm sunması beklenmektedir.

2.3. Tasarım

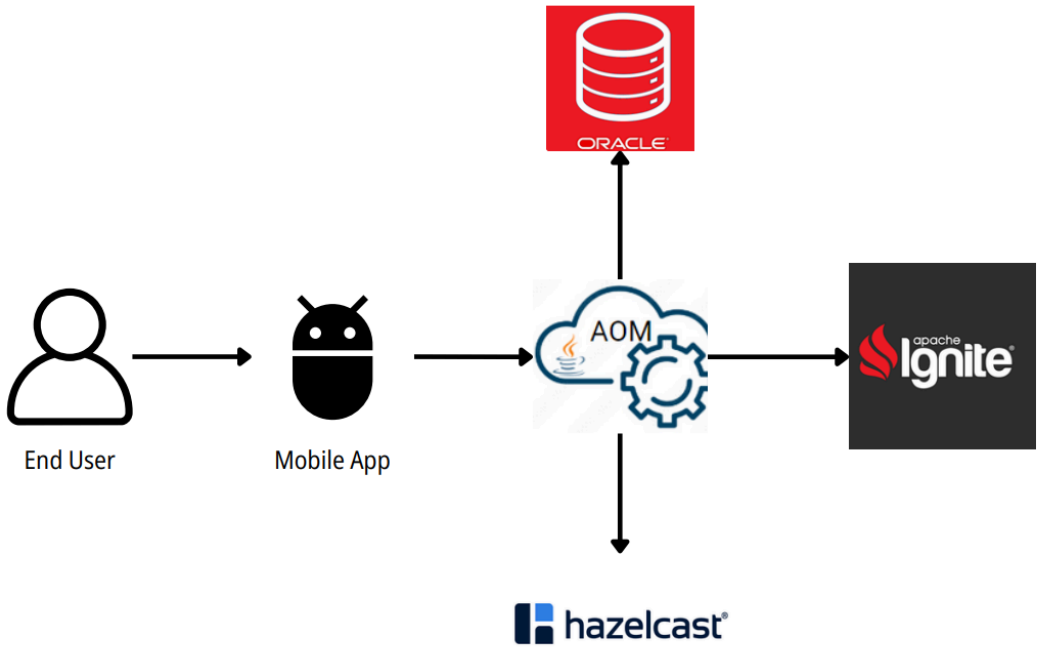
2.3.1. Mikroservis Mimarisinin Detayları



AOM (Account Order Management)

AOM Modülünün Tanımı:

AOM (Account Order Management), telekomünikasyon sistemlerinde kullanıcının paket ve hesap bilgilerini yönetmek için tasarlanmış temel bir modüldür. Bu modül, kullanıcıların kayıt işlemlerini, yetkilendirme süreçlerini ve abone oldukları tarifelerin takibini sağlar. Kullanıcıların paket ve bakiye bilgilerine erişimini mümkün kılar ve bu bilgilerin güvenli bir şekilde saklanmasını temin eder.



AOM Modülünün İşlevleri:

1. **Kullanıcı Kayıt İşlemleri:**
   - Kullanıcının sisteme ilk defa kayıt olması sırasında, gerekli bilgileri (örneğin, kullanıcı adı, telefon numarası, e-posta) toplar.
   - Kullanıcının seçtiği tarifeyi (örneğin, SMS, ses veya veri paketleri) sisteme kaydeder.

2. **Yetkilendirme İşlemleri:**
   - Sistemde oturum açmak isteyen kullanıcıların kimlik doğrulama ve yetkilendirme işlemlerini gerçekleştirir.
   - Kullanıcının aktif bir tarifeye sahip olup olmadığını kontrol eder.

3. **Veri Yönetimi:**
   - Kullanıcı bilgileri ve paket detayları, **kalıcı bir veritabanında** saklanır.
   - Kullanıcı verileri, performansı artırmak ve hızlı erişim sağlamak için:
     - **Bellek tabanlı veritabanına (in-memory database - Apache Ignite)** yüklenir.
     - **Hazelcast** gibi önbellek tabanlı bir veri tabanına eklenir.
   - Veritabanı ve önbellek senkronizasyonu sağlanarak veri bütünlüğü korunur.
4. **Bilgi Döndürme:**
   - Kullanıcının talep etmesi durumunda, paket bilgilerini (örneğin, mevcut tarife, kalan dakika, SMS, veri) geri döndürür.
   - Güncellenmiş bakiye ve kullanım detaylarını hızlı bir şekilde iletir.

AOM Modülünün Avantajları:

- **Hızlı Erişim:** Hazelcast kullanımı sayesinde, kullanıcı bilgilerine düşük gecikme süresiyle erişim sağlanır.
- **Veri Güvenliği:** Kalıcı ve bellek tabanlı veritabanlarının birlikte kullanımı, verilerin hem güvenli hem de hızlı bir şekilde işlenmesine olanak tanır.
- **Esneklik:** Yeni tarife ve kullanıcı bilgileri kolaylıkla sisteme entegre edilebilir.
- **Ölçeklenebilirlik:** Artan kullanıcı sayısına rağmen sistem performansı düşmeden hizmet verebilir.

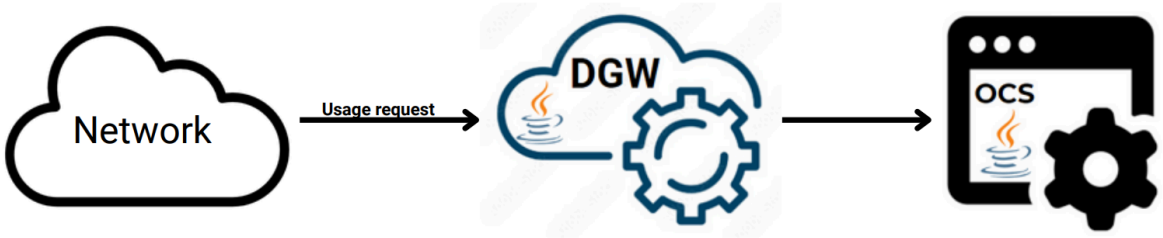
AOM ile İlgili Veri Akışı Örneği:

1. Kullanıcı, yeni bir tarife seçimiyle kayıt işlemi başlatır.
2. AOM, kullanıcıdan aldığı bilgileri:
    - Kalıcı veritabanına kaydeder.
    - Bellek tabanlı veritabanına yükler.
    - Hazelcast önbelleğine ekler.
3. Kullanıcı, sistemden kalan bakiye ve paket detaylarını talep ettiğinde, bu bilgiler hızlı bir şekilde önbellekten döndürülür.
4. Yetkilendirme süreçlerinde, kullanıcı oturumu açarken AOM tarafından doğrulama yapılır.

DGW (Diameter Gateway)

Tanım:

DGW (Diameter Gateway), telekomünikasyon sistemlerinde Diameter protokolünü simüle eden bir ağ geçidi modülüdür. Gerçek Diameter protokolü yerine simüle edilmiş bir ortamda çalışarak kullanım verilerinin akışını yönetir ve bu verileri çevrimiçi ücretlendirme sistemi (OCS) modülüne iletir. DGW, kullanım verilerini doğrular ve ilgili bileşenlere iletilmesini sağlar.



İşlevleri:

1. **Kullanım Verilerinin Alınması:**
    - **TGF (Traffic Generator Function):** Simüle edilmiş bir ortamda, DGW'ye kullanım verileri TGF tarafından iletilir.
    - Kullanım verileri, çağrı, veri veya mesajlaşma gibi hizmetlerin tüketim bilgilerini içerir.
2. **Kullanıcı Durumunun Doğrulanması:**
    - DGW, önbellekteki (örneğin, Hazelcast) kullanıcı verilerini sorgular.
    - Kullanıcının sistemde mevcut olup olmadığı kontrol edilir.
    - Eğer kullanıcı sistemde kayıtlıysa veri OCS'ye iletilir. Kayıtlı değilse işlem durdurulur.

3. **Veri Yönlendirme:**
    - Kullanım verilerini OCS modülüne gönderir.
    - Diameter protokolünün işlevlerini simüle ederek veri aktarımını sağlar.
4. **Protokol Simülasyonu:**
    - Gerçek Diameter protokolü yerine, simüle edilmiş bir versiyon kullanılarak sistem testleri ve geliştirme süreçleri yürütülür.

Özellikleri:

- **Simüle Edilmiş Ortam:** DGW, gerçek Diameter protokolü yerine TGF tarafından üretilen simüle edilmiş kullanım verilerini işler. Bu, geliştirme ve test süreçlerini kolaylaştırır.
- **Önbellek Entegrasyonu:** Kullanıcı doğrulama işlemleri için Hazelcast gibi bir önbellek veri tabanına bağlanır.
- **Hız ve Performans:** Verilerin anlık olarak işlenmesini ve doğru bileşene yönlendirilmesini sağlar.
- **Test Amaçlı Kullanım:** Diameter protokolünün simüle edilmesi, sistem performansını gerçek ortamdan bağımsız olarak test etmek için avantaj sağlar.

Sistem İşleyişinde Rolü:

1. **TGF'den Gelen Veriler:**
    - TGF modülü, çağrı, veri ve SMS gibi tüketim verilerini DGW'ye iletir.
2. **Önbellek Kontrolü:**
    - Kullanıcı verisi önbellekte sorgulanır. Eğer kullanıcı mevcutsa işlem devam eder.
3. **OCS'ye İletim:**
    - Doğrulanan kullanım verisi OCS modülüne gönderilir.
    - OCS, bu veriyi işleyerek kullanıcıyı ücretlendirir.

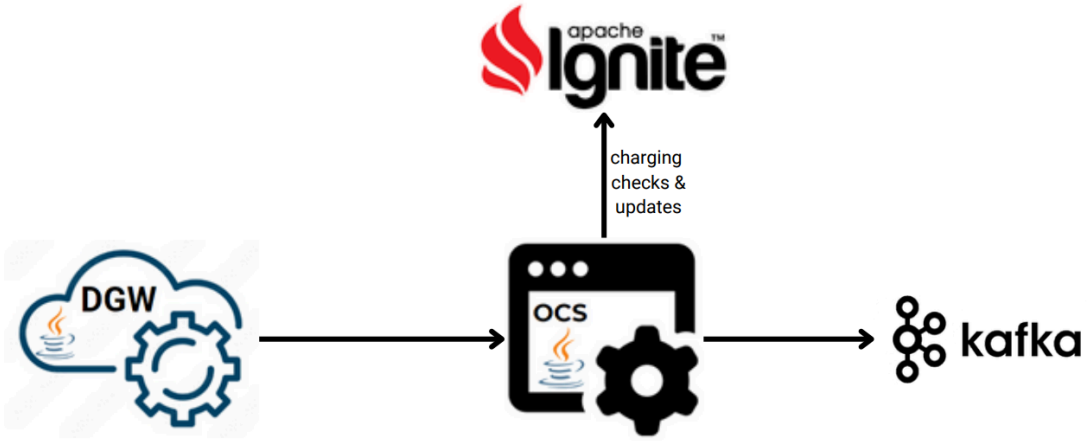
Avantajları:

- **Doğruluk:** Kullanıcı verilerini önbellekte sorgulayarak işlemleri doğru ve güvenli bir şekilde yürütür.
- **Hızlı Veri İşleme:** Anlık veri işleme ve yönlendirme sayesinde sistem performansını artırır.
- **Esneklik:** Gerçek Diameter protokolü yerine simülasyon kullanılarak geliştirme süreçleri hızlandırılır.
- **Geliştirme ve Test Kolaylığı:** Gerçek sistem yükü olmaksızın, protokolün simüle edilmesiyle sistem testleri daha hızlı ve düşük maliyetle yapılır.

OCS (Online Charging System)

Tanım:

OCS (Online Charging System), gerçek zamanlı ücretlendirme işlemlerini gerçekleştiren ve sistemin en kritik bileşenlerinden biridir. Kullanıcının veri tüketimi, ses, SMS gibi servislerini takip eder, anlık olarak hesaplanan maliyetleri kullanıcı hesabından düşer ve kullanıcının hizmet alımını yönlendirir. Abonelerin kullanım durumuna göre yetkilendirme, ücretlendirme ve bakiye kontrol süreçlerini yönetir.



İşlevleri:

1. **Ücretlendirme:**
   - Ücretlendirme işlemi, hizmet türüne (ses, SMS, veri) ve abone tarifesine göre gerçekleştirilir.
2. **Bakiye Kontrolü:**
   - Kullanıcının hesabındaki mevcut bakiyeyi kontrol eder.
   - Yetersiz bakiye durumunda hizmeti durdurabilir ya da kullanıcıya bilgilendirme yapabilir.
3. **Hizmet Yetkilendirme:**
   - Kullanıcının hizmet alımını başlatmadan önce bakiye durumunu kontrol eder ve yetkilendirme sağlar.
   - Örneğin, kullanıcı bir sesli arama başlatmak istediğinde, CHF bu işlemi yetkilendirir ve maliyeti anlık olarak düşer.
4. **Veri İletimi ve Entegrasyon:**
   - Kullanım verilerini *Kafka* gibi mesajlaşma sistemleri üzerinden diğer modüllere (ABMF ve CGF gibi) iletir.
   - Bu sayede sistemdeki diğer fonksiyonlarla entegrasyonu sağlar.
5. **DGW (Diameter Gateway) ile Etkileşim:**
   - DGW tarafından gönderilen kullanım verilerini alır.
   - Verileri işler, maliyetini hesaplar ve ilgili bileşenlere yönlendirir.
6. **Esnek ve Modüler Yapı:**
   - Farklı medya türlerini (ses, veri, SMS gibi) ve hizmet modellerini destekleyecek şekilde tasarlanmıştır.

Özellikleri:

- **Hizmet Bazlı Ücretlendirme:** Kullanıcının hizmet türüne göre özelleştirilmiş ücretlendirme seçenekleri sunar.
- **Modüler Entegrasyon:** ABMF, CGF gibi diğer OCS modülleriyle yüksek uyumluluk sağlar.
- **Veri Kaybını Önleme:** Kafka gibi mesajlaşma sistemleri ile veri aktarımı sırasında oluşabilecek kayıpları minimize eder.

Sistem İşleyişinde Rolü:

1. **DGW'den Gelen Veriler:**
    - Kullanıcıların tüketim verileri DGW üzerinden OCS'ye iletilir.
2. **Ücretlendirme ve Bakiye Kontrolü:**
    - Kullanıcının tüketim bilgileri işlenir ve maliyeti hesaplanır.
    - Hesaptan düşülen maliyet bilgisi, ABMF'ye iletilir.
3. **Hizmet Yetkilendirme:**
    - Yetersiz bakiye durumunda kullanıcıya bilgilendirme yapılır veya hizmet durdurulur.
    - Bakiye yeterliyse hizmet yetkilendirilir ve süreç devam eder.
4. **Veri Aktarımı:**
    - İşlenen veriler, ABMF (Account Balance Management Function) ve CGF (Charging Gateway Function) gibi diğer modüllere iletilir.

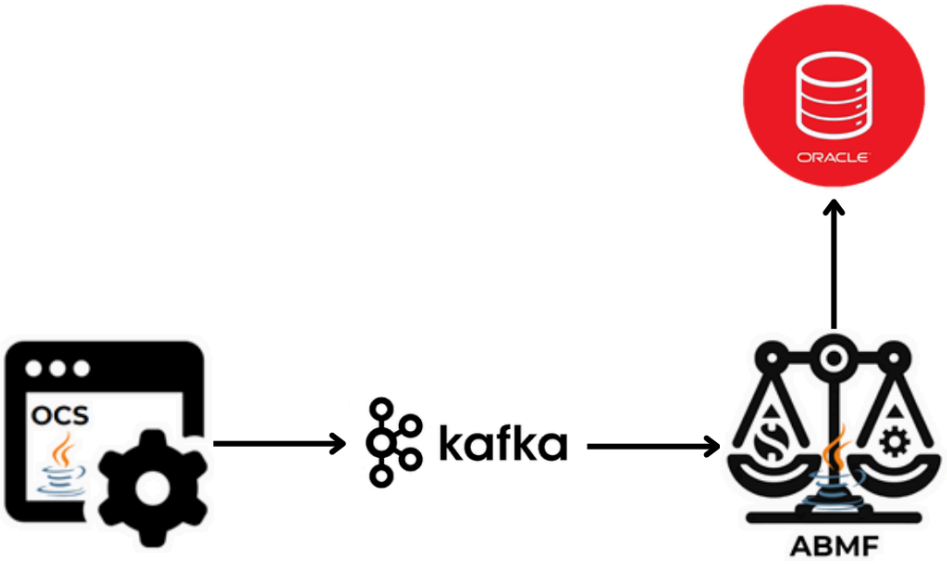
Avantajları:

- **Anlık ve Doğru Ücretlendirme:** Kullanıcı deneyimini iyileştiren hızlı ve hatasız işlem sağlar.
- **Esneklik ve Ölçeklenebilirlik:** Yeni hizmet türlerinin kolayca entegre edilmesine olanak tanır.
- **Güvenilir Veri İşleme:** Kafka ile mesaj aktarımı yaparak sistem performansını artırır.
- **Uyumluluk:** Modern telekomünikasyon ihtiyaçlarına uygun, esnek bir yapı sunar.

ABMF (Account Balance Management Function)

Tanım:

ABMF (Account Balance Management Function), kullanıcı hesaplarının bakiyesini yönetmek ve gerçek zamanlı olarak güncellemek üzere tasarlanmış bir modüldür. OCS (Online Charging System) tarafından yapılan geçici bakiye işlemlerini kalıcı hale getirerek sistemin veri tutarlılığını ve güvenilirliğini sağlamaktadır. ABMF, aynı zamanda sistemin diğer modülleri ile entegre çalışarak kullanıcı hesap bilgilerinin doğru ve güncel olmasını garanti eder.



İşlevleri:

1. **Gerçek Zamanlı Bakiye Yönetimi:**
    - OCS tarafından geçici olarak güncellenen bakiye verilerini alır ve Oracle gibi kalıcı bir veritabanında saklar.
    - Kullanıcıların tüketim bilgilerine dayalı olarak bakiye güncellemelerini anlık şekilde senkronize eder.
2. **Hesap Yetkilendirme:**
    - Kullanıcının bakiyesini kontrol ederek hizmet yetkilendirme işlemlerine destek olur.
3. **Tüketim ve Kullanım Verilerinin Yönetimi:**
    - Kullanıcının tüketim bilgilerini (örneğin, kullanılan dakika, SMS veya veri miktarı) kaydeder ve bu bilgileri CGF modülüne aktarır.
    - Bu veriler, faturalandırma ve kullanıcı bilgilendirme süreçlerinde kullanılır.
4. **Veri Entegrasyonu ve Senkronizasyon:**
    - OCS'nin Apache Ignite üzerinden yaptığı geçici değişiklikleri Oracle gibi kalıcı bir veritabanına entegre eder.

Özellikleri:

- **Tutarlılık ve Kesinlik:** Kullanıcı hesap bilgilerini doğru bir şekilde yönetir ve veri tutarsızlıklarını önler.
- **Esneklik:** Farklı medya türleri ve paket yapılarıyla kolayca uyum sağlar.
- **Gerçek Zamanlı İşleme:** Tüm işlemleri anlık olarak gerçekleştirir ve sistemin diğer modülleri ile senkronize bir şekilde çalışır.
- **Güvenilir Veri Yönetimi:** Oracle gibi kalıcı veritabanları ile entegrasyon sağlayarak veri kayıplarını önler.

Sistem İşleyişinde Rolü:

1. **OCS ile Etkileşim:**
    - OCS tarafından Apache Ignite üzerinde yapılan bakiye güncellemelerini Oracle veritabanına aktarır.
    - Kullanıcı hesap bilgilerini güncel tutarak sistemin diğer modülleri için güvenilir bir veri kaynağı sağlar.
2. **Oracle Veritabanı ile İşbirliği:**
    - Kullanıcı hesap bilgilerini kalıcı olarak saklar ve sistem yeniden başlatıldığında bu verilerin güvenilir şekilde yüklenmesini sağlar.

Avantajları:

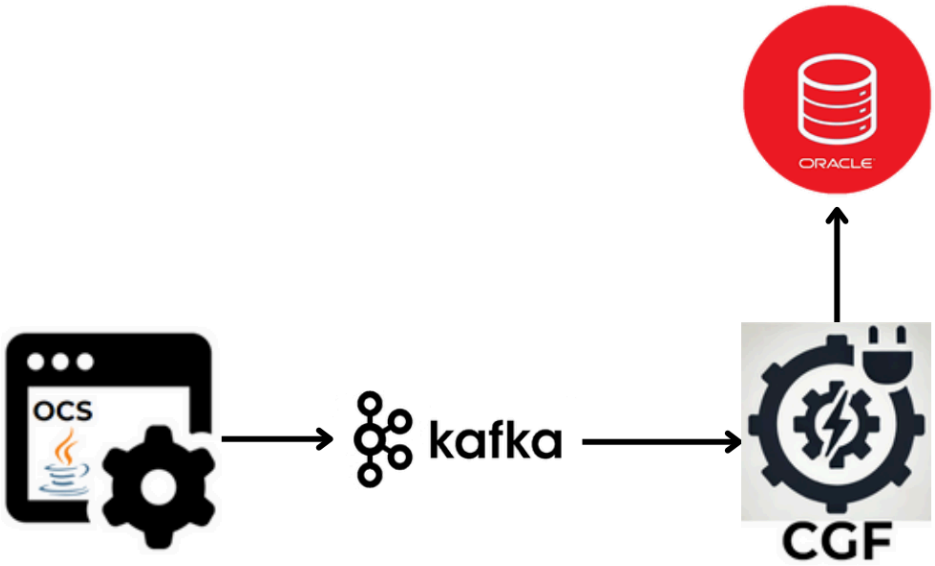
- **Veri Güvenliği ve Yönetimi:** Kullanıcı hesap verilerinin güvenilir şekilde yönetilmesi ile müşteri memnuniyeti artırılır.
- **Gerçek Zamanlı Güncellemeler:** Anlık güncellemelerle kullanıcılar ve operatörler için doğru ve güncel bilgi sağlanır.

- **Mikroservis Mimarisiyle Uyum:** Diğer mikroservis modülleriyle performans kaybı olmaksızın entegre olabilir.
- **Ölçeklenebilirlik:** Kullanıcı sayısındaki ve tüketim hacmindeki artışlara kolaylıkla uyum sağlar.

CGF (Charging Gateway Function)

Tanım:

CGF (Charging Gateway Function), telekomünikasyon sistemlerinde faturalandırma ve veri yönetim süreçlerinde kritik bir rol oynayan modüldür. CGF, kullanıcı tüketim verilerini toplar, işler ve kalıcı veritabanında saklar. Aynı zamanda bu verileri, faturalandırma süreçlerine veya raporlama ihtiyaçlarına uygun hale getirir. CGF, OCS tarafından yapılan ücretlendirme işlemlerinin detaylarını kaydeder.



İşlevleri:

1. **Kullanıcı Tüketim Detaylarının Toplanması ve İşlenmesi:**
   - OCS (Online Charging System) tarafından gönderilen kullanıcı tüketim verilerini işler.
   - Kullanıcının gerçekleştirdiği işlemler (örneğin, sesli arama süresi, SMS gönderimi, veri kullanımı) gibi bilgiler detaylı şekilde kaydedilir.
2. **Veri Saklama ve Yönetim:**
   - Toplanan tüketim detaylarını Oracle gibi kalıcı bir veritabanında saklar.
   - Bu veriler, müşteri hizmetleri, raporlama veya denetim gibi işlemler için daha sonra kullanılabilir.

Özellikleri:

- **Kesinlik ve Güvenilirlik:** Tüm kullanıcı tüketim bilgileri doğru ve eksiksiz şekilde kaydedilir.
- **Esneklik:** Farklı hizmet türleri ve operatör gereksinimlerine kolayca uyarlanabilir.

- **Raporlama ve Analitik:** Faturalandırma süreçlerinin yanı sıra operatörlerin raporlama ve analiz ihtiyaçlarını da karşılar.

Sistem İşleyişinde Rolü:

1. **OCS ile Entegrasyon:**
    - OCS tarafından gönderilen ücretlendirme detaylarını işler ve kaydeder.
    - Bu detaylar faturalandırma ve müşteri bilgilendirme süreçleri için kullanılır..
2. **Oracle Veritabanı ile Veri Yönetimi:**
    - Kullanıcı verilerini uzun vadeli saklama ve raporlama ihtiyaçlarına uygun olarak kalıcı hale getirir.

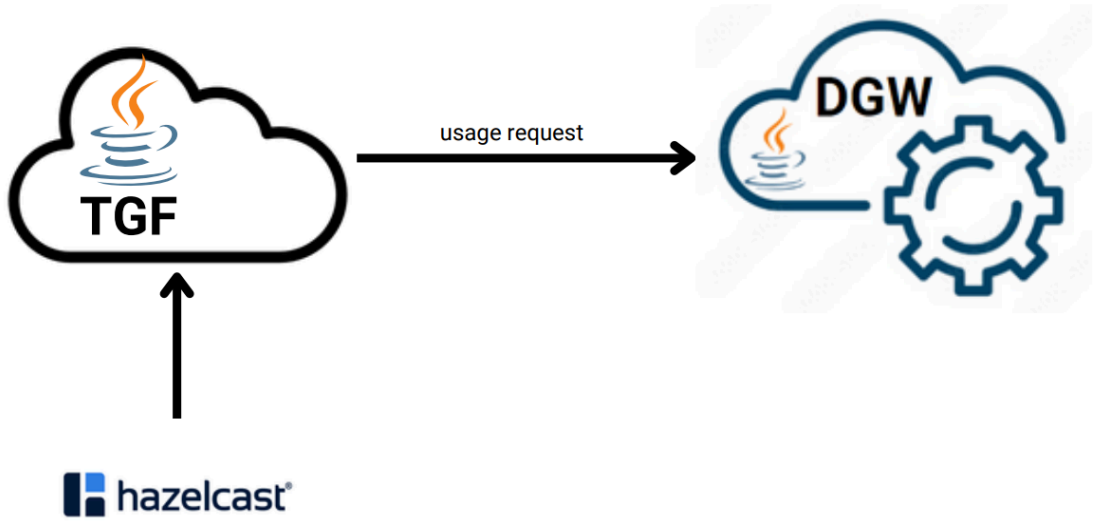
Avantajları:

- **Faturalandırma Süreçlerini Hızlandırma:** Tüketim verilerini düzenli bir şekilde sunarak fatura oluşturma sürecini optimize eder.
- **Yüksek Performans:** Büyük miktarda veriyi işleyebilir ve saklayabilir.
- **Mikroservis Mimarisiyle Uyum:** Diğer modüllerle sorunsuz entegre çalışabilir.
- **Ölçeklenebilirlik:** Artan kullanıcı sayısına ve veri hacmine uyum sağlar.

TGF (Traffic Generator Function)

Tanım:

TGF (Traffic Generator Function), telekomünikasyon sistemlerinde kullanıcı davranışlarını ve hizmet tüketimini simüle etmek amacıyla tasarlanmış bir modüldür. Bu modül, gerçek kullanıcı verilerini taklit ederek sistemin test edilmesi, performansının değerlendirilmesi ve farklı kullanım senaryolarında davranışlarının analiz edilmesi için kritik bir rol oynar. Özellikle DGW (Diameter Gateway) modülü ile entegre çalışarak sistemin kullanım verilerini işler.



İşlevleri:

1. **Trafik Üretimi ve Simülasyonu:**

    - Gerçek kullanıcı davranışlarını taklit etmek için rastgele veya senaryoya dayalı trafik üretir.
    - Ses, veri, SMS gibi çeşitli hizmet türlerine yönelik farklı tüketim senaryoları oluşturur.
2. **Sistem Performans Testi:**
    - Üretilen trafiği DGW modülüne göndererek sistemin yük altında nasıl çalıştığını analiz etmeyi sağlar.
    - Sistem performansını değerlendirmek için stres testi ve ölçeklenebilirlik testlerine katkı sunar.
3. **Veri Akışının Sağlanması:**
    - DGW'ye iletilen kullanım verilerinin doğruluğunu kontrol eder.
    - Kullanıcı bazlı tüketim verilerini OCS (Online Charging System) modülüne yönlendirir.
4. **Test Ortamı Desteği:**
    - Simüle edilen verilerle sistemin farklı senaryolara nasıl tepki verdiğini test etmeyi mümkün kılar.
    - Sistem geliştirme aşamasında olası hataların erken tespit edilmesini sağlar.

Özellikleri:

- **Esneklik:** Farklı trafik senaryoları oluşturabilir ve değişken parametrelere göre test gerçekleştirebilir.
- **Gerçekçiliği Artırılmış Simülasyon:** Kullanıcı davranışlarını doğru şekilde taklit ederek sistemi gerçek dünya koşullarında test etmeye olanak tanır.
- **Senkronizasyon:** Üretilen veriler, sistemin diğer modülleriyle (DGW, OCS) uyumlu bir şekilde çalışır.
- **Performans Değerlendirme:** Sistem üzerindeki trafik yükünün etkilerini analiz ederek, darboğazların ve performans sorunlarının tespitine yardımcı olur.

Sistem İşleyişinde Rolü:

1. **DGW ile Entegrasyon:**
    - TGF tarafından üretilen kullanım verileri DGW'ye iletilir ve DGW üzerinden OCS'ye yönlendirilir.
    - Simüle edilen veri akışı, sistemin tüm modüllerini test etmeyi sağlar.
2. **OCS ile Veri Akışı:**
    - TGF'den gelen veri, OCS tarafından ücretlendirilir ve hesap yönetimi yapılır.
    - Farklı hizmet türlerine yönelik tarifelerin test edilmesine olanak sağlar.
3. **Test ve Analiz:**
    - Sistem geliştirme sırasında senaryo bazlı testlerin yapılmasını sağlar.
    - Operatörlerin gelecekteki trafik artışlarına hazır olup olmadığını analiz etmek için kullanılır.

Avantajları:

- **Sistem Güvenilirliği:** Sistem modüllerinin tümü, farklı trafik senaryoları ile test edilerek olası hatalar önceden tespit edilir.
- **Ölçeklenebilirlik Testleri:** Sistem büyüdükçe performansın nasıl değiştiğini analiz etmek için trafik yükü simüle edilir.
- **Hızlı Geliştirme:** Geliştirme aşamasında test verilerinin manuel olarak oluşturulmasına gerek kalmadan sistem daha hızlı test edilir.

- **Gerçekçi Senaryolar:** Çeşitli kullanıcı davranışlarını yansıtan trafik simülasyonlarıyla sistem gerçek dünyaya uygun şekilde hazırlanır.

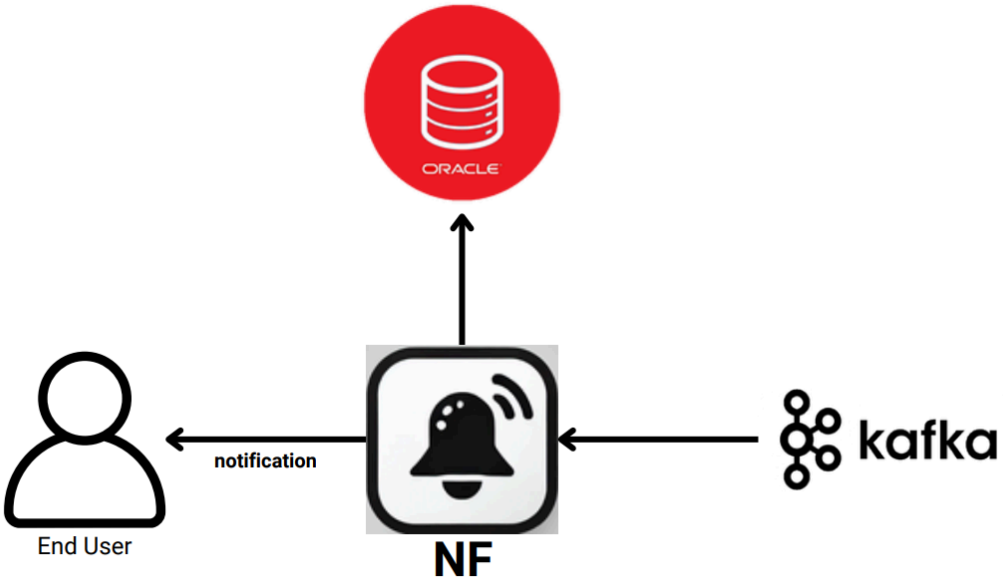
Sistem İçindeki Örnek Kullanım:

- **Senaryo 1:** Bir kullanıcının veri tüketimini simüle ederek ABMF ve CGF modüllerinin doğru şekilde çalışıp çalışmadığını test etmek.
- **Senaryo 2:** Çok sayıda eşzamanlı arama simüle ederek sistemin yük altında nasıl performans gösterdiğini analiz etmek.

NF (Notification Function)

Tanım:

Notification Function (NF), kullanıcılarla iletişim kurarak sistem tarafından oluşturulan bildirimlerin zamanında ve doğru bir şekilde iletilmesini sağlayan modüldür. NF, hem kullanıcıların aylık ücretlendirme işlemleriyle ilgili bildirimlerini oluşturmak hem de kullanım bazlı özel durumlar (örneğin, %80 kota kullanımı) için anlık bilgilendirmeler yapmak üzere tasarlanmıştır. Kafka üzerinden aldığı mesajları işleyerek kullanıcılara uygun kanallardan (SMS, e-posta vb.) iletir.



İşlevleri:

1. **Aylık Ücretlendirme Bildirimi:**
    - **AOM Entegrasyonu:** Kullanıcı ilk oluşturulduğunda AOM tarafından Kafka'ya yazılan kayıt bilgisini alır.
    - **Job Oluşturma:** AOM ile entegre çalışan NF, her kullanıcı için bir "ay sonu ücretlendirme bildirim" işini (job) planlar.
    - **Bildirim Gönderimi:** Job tamamlandığında kullanıcılara fatura bilgilerini içeren bildirim gönderir.
2. **Kullanım Bazlı Bildirimler:**

    - **Kafka Entegrasyonu:** OCS tarafından Kafka'ya yazılan kullanım verilerini dinler.
    - **%80 Kota Kullanımı:** Kullanıcının belirli bir hizmet kotasının (%80, %100 vb.) aşılması durumunda, sistemin belirlediği kurallar doğrultusunda anlık bildirim gönderir.
    - **Ekstra Bildirimler:** Olası kota aşımı veya bakiye tükenmesi gibi durumlarda da kullanıcıyı bilgilendirir.
3. **Çok Kanallı Bildirim Gönderimi:**
    - Kullanıcının tercihine bağlı olarak bildirimleri e-posta veya sms olarak iletir.
    - Bildirimlerin iletildiğine dair log kaydı tutar.

Özellikleri:

- **Gerçek Zamanlı Bildirimler:** Kafka'dan aldığı mesajları anlık olarak işler ve ilgili kullanıcıya hızlıca iletir.
- **Job Yönetimi:** Kullanıcıların aylık ücretlendirme bildirimlerini düzenli olarak göndermek için otomatik işler (jobs) planlar.
- **Loglama:** Gönderilen tüm bildirimleri kaydederek sistemde izlenebilirliği artırır.

Sistem İşleyişinde Rolü:

1. **AOM ile Entegrasyon:**
    - Kullanıcı oluşturulduğunda, AOM'un Kafka'ya yazdığı mesajları okuyarak bu kullanıcı için bildirim süreçlerini başlatır.
    - Kullanıcıların aylık ücretlendirme bildirimleri için job planlar.
2. **OCS ile Entegrasyon:**
    - OCS'nin Kafka'ya yazdığı kullanım verilerini dinleyerek, kullanıcıların belirli eşiklere ulaştığında bilgilendirilmesini sağlar (%80, %100 kota kullanımı gibi).
    - OCS'nin sağladığı tüketim bilgileri doğrultusunda anlık bildirimler oluşturur.
3. **Kullanıcıya Bilgi Gönderimi:**
    - Kullanıcının belirli bir hizmet türünde kotasının dolduğunu veya yetersiz bakiyesi olduğunu bildirir.
    - Fatura kesim tarihini ve ödenecek tutarı iletir.

2.3.1.2. Mikroservis Mimarisinde Kullanılan Teknolojiler

1. Kafka

Tanım:

Kafka, yüksek performanslı, ölçeklenebilir ve dağıtık bir mesajlaşma platformudur. Veri akışını yöneten ve modüller arası iletişimi sağlayan bir mesaj kuyruğu sistemi olarak görev yapar. OCS gibi mikroservis tabanlı sistemlerde, modüller arasında güvenilir ve asenkron iletişim kurmak için kullanılır.

Kullanımı:

- **Mesajlaşma Sistemi:** Tüm modüller arasında veri akışını yönetir. Örneğin, OCS, kullanıcıların tüketim verilerini Kafka üzerinden diğer modüllere iletir.

- **Gerçek Zamanlı İşleme:** Kullanıcıların anlık kota kullanımları, bakiye bilgileri ve bildirim ihtiyaçları Kafka aracılığıyla iletilir.

Avantajları:

- Modüller arasında bağımsız iletişim.
- Yüksek veri işleme kapasitesi.
- Gelişmiş hata yönetimi ve dayanıklılık.

2. Oracle

Tanım:

Oracle, güvenilir ve performanslı bir veritabanı yönetim sistemidir. Verilerin kalıcı olarak saklanması ve düzenlenmesi için kullanılır. Telekomünikasyon sistemlerinde hassas kullanıcı bilgileri ve tüketim verilerinin güvenilir bir şekilde saklanmasını sağlar.

Kullanımı:

- **Kalıcı Veri Depolama:**
  - Kullanıcı hesap bilgileri (AOM tarafından eklenir).
  - Kullanıcıların aylık fatura bilgileri ve kullanım detayları.
- **ABMF Entegrasyonu:** ABMF, bakiyeleri güncellemek ve senkronizasyon sağlamak için Oracle veritabanını kullanır.
- **Faturalama ve Raporlama:** NF modülü, faturalama için gerekli bilgileri Oracle'dan alır.

Avantajları:

- Yüksek güvenilirlik ve veri bütünlüğü.
- Karmaşık sorgulama yetenekleri.
- Büyük veri kümeleriyle ölçeklenebilir çalışma kapasitesi.

3. Hazelcast

Tanım:

Hazelcast, dağıtık bir önbellekleme ve veri işleme platformudur. Kullanıcı bilgilerine hızlı erişim ve sık kullanılan verilerin depolanması için kullanılır. Bellek tabanlı bir yapı sunduğu için, performansı artırır ve veriye erişim süresini azaltır.

Kullanımı:

- **AOM ile Entegrasyon:**
  - Kullanıcı bilgileri (Telefon numarası, id) Hazelcast’te saklanır.
- **Sistem Performansını Artırma:** Oracle gibi kalıcı veri tabanına erişimi minimize ederek, yüksek performans sağlar.

Avantajları:

- Yüksek hızlı veri erişimi.
- Dağıtık yapı ile ölçeklenebilirlik.

4. Apache Ignite

Tanım:

Apache Ignite, bellek içi bir veri platformudur. Verilerin hızla işlenmesi ve modüller arası paylaşılan verilerin yönetimi için kullanılır. Mikroservis tabanlı sistemlerde anlık veri yönetimi sağlar.

Kullanımı:

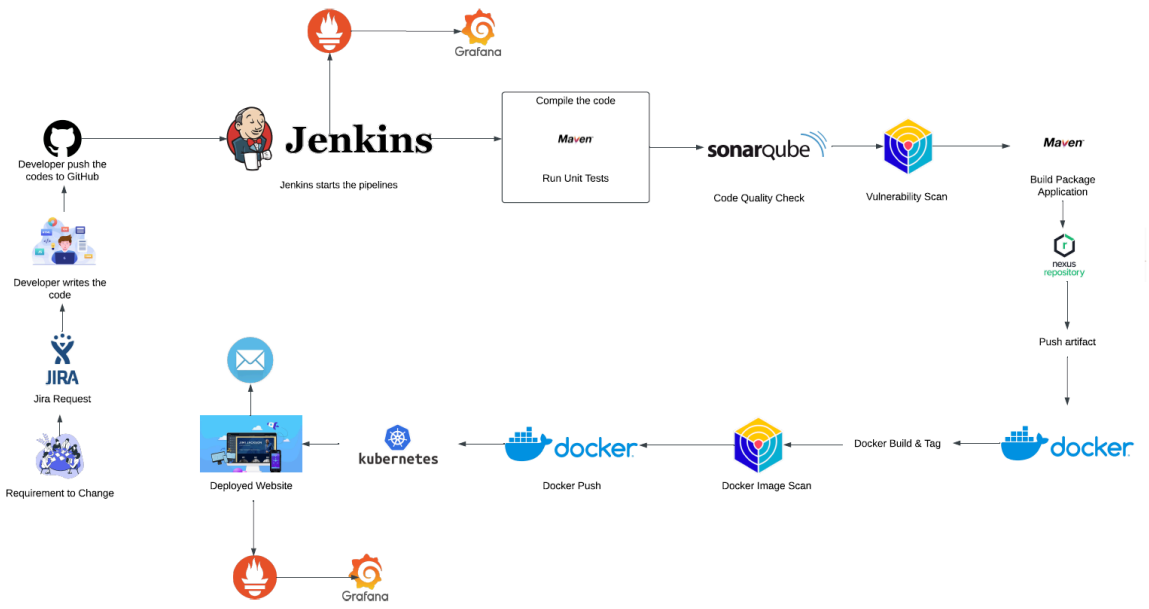
- **OCS ile Entegrasyon:**
  - OCS, kullanıcının anlık ücretlendirme verilerini Ignite üzerinde yönetir.
  - Oracle veritabanına yazılmadan önce geçici olarak Ignite'ta saklanır.
- **ABMF ile Entegrasyon:**
  - ABMF, OCS'nin Ignite üzerinde güncellediği verileri alarak Oracle'a senkronize eder.

Avantajları:

- Gerçek zamanlı veri işleme.
- Yüksek performanslı sorgulama.
- Dağıtık yapı ile verinin güvenliğini artırma.

2.3.2. CI-CD

2.3.2.1. Jenkins ile CI-CD Pipeline



1-Değişiklik Gereksinimi: Gereksinimlerde bir değişiklik ihtiyacı belirlenir. Bu genellikle bir iş talebi veya müşteri gereksinimiyle başlar.

2-Jira Talebi: Değişiklik talebi bir iş takip sistemi (örneğin Jira) aracılığıyla kaydedilir.

3-Geliştirici Kod Yazar: Geliştiriciler, talebe uygun kodları yazarlar.

4-Kodların GitHub'a Gönderilmesi: Yazılan kodlar, bir versiyon kontrol sistemi olan GitHub'a gönderilir.

5-Jenkins Pipeline'larını Çalıştırır: Jenkins, kodların entegre edilmesi ve dağıtımı için önceden tanımlanmış CI/CD boru hatlarını çalıştırır.

6-Birim Testlerini Çalıştırma: Pipeline, yazılan kodlar üzerinde birim testleri çalıştırır.

7-Kod Kalitesi Kontrolü ve Güvenlik Taraması: Kod kalitesi analiz edilir ve güvenlik açıkları taranır.

8-Uygulama Paketi Oluşturma: Kod, çalışabilir bir uygulama paketi haline getirilir.

9-Artefaktın Depoya Gönderilmesi: Oluşturulan paket, bir artefakt deposuna (örneğin, Nexus, Artifactory) gönderilir.

10-Docker İmajı Oluşturma ve Etiketleme: Docker imajı oluşturulur ve etiketlenir.

11-Docker İmajı Güvenlik Taraması: Docker imajları güvenlik açıklarına karşı taranır.

12-Docker İmajını Depoya Gönderme: İmaj, Docker Hub veya özel bir konteyner deposuna gönderilir.

13-Web Sitesinin Dağıtımı: Son olarak, oluşturulan uygulama bir ortama (örneğin, prodüksiyon veya staging) dağıtılır ve kullanıma hazır hale getirilir.

14-Pipeline Durumunun Yöneticilere Bildirilmesi: Pipeline süreci tamamlandığında Jenkins veya benzeri bir otomasyon aracıyla yöneticilere e-posta gönderilir. Bu e-posta genellikle pipeline başarısı, hata durumları veya test raporlarını içerir.

15-Prometheus ve Grafana ile İzleme:

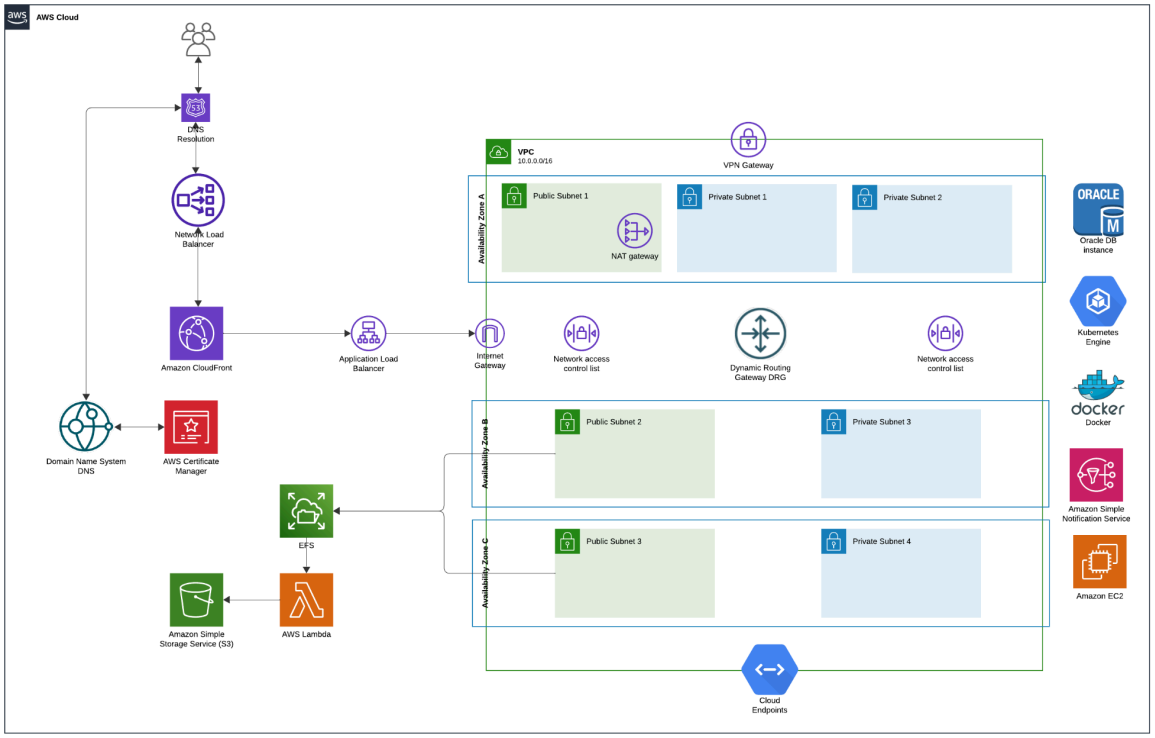
> **Prometheus**: Sistem metriklerini ve pipeline süreçlerini gerçek zamanlı olarak toplar. Pipeline süreçlerinde yaşanan hata veya gecikmeleri tespit etmek için veriler toplanır.
>
> **Grafana**: Prometheus'tan alınan verilerle kullanıcı dostu bir şekilde izleme panelleri oluşturulur.

Pipeline Yönetiminde Kullanılan Teknolojiler

1. **Jira**
    - *Amaç:* Yazılım geliştirme ekibinin görevlerini ve isteklerini yönetmek için kullanılan bir proje yönetim aracı.
    - *Kullanımı:* Gereksinim değişiklikleri burada açılan taleplerle başlatılır.

2. **GitHub**
    - *Amaç:* Kaynak kodlarının versiyon kontrolü ve paylaşımı için kullanılan platform.
    - *Kullanımı:* Geliştiriciler kodu yazdıktan sonra GitHub'a push ederler.
3. **Jenkins**
    - *Amaç:* CI/CD süreçlerini otomatikleştiren bir açık kaynaklı araç.
    - *Kullanımı:* Jenkins pipeline'ları başlatır, kodu derler, test eder ve sonraki adımları tetikler.
4. **Maven**
    - *Amaç:* Java projeleri için derleme, bağımlılık yönetimi ve paketleme aracı.
    - *Kullanımı:* Kodun derlenmesi, unit testlerin çalıştırılması ve uygulamanın paketlenmesi için kullanılır.
5. **SonarQube**
    - *Amaç:* Kod kalitesi analizini ve hatalarını tespit eden bir araç.
    - *Kullanımı:* Jenkins aracılığıyla entegre edilir ve kodun kalite kontrolünü sağlar.
6. **Nexus Repository**
    - *Amaç:* Derlenmiş artefaktların saklanması ve yönetilmesi için kullanılan bir depo.
    - *Kullanımı:* Maven tarafından oluşturulan paketler burada saklanır.
7. **Docker**
    - *Amaç:* Uygulamaları konteynerleştirme aracı, taşınabilir çalışma ortamları sağlar.
    - *Kullanımı:*
        - Uygulamanın Docker imajı oluşturulur, etiketlenir ve Nexus'tan ya da doğrudan bir Docker registries'e push edilir.
        - Docker Push adımıyla imaj, Kubernetes tarafından kullanılır.
8. **Kubernetes**
    - *Amaç:* Konteynerlerin otomatik olarak orkestrasyonu ve yönetilmesi.
    - *Kullanımı:* Docker imajı Kubernetes cluster'ında deploy edilir.
9. **Grafana**
    - *Amaç:* Sistem performansını izleme ve görselleştirme aracı.
    - *Kullanımı:* Jenkins ve Kubernetes metriklerini izlemek için kullanılır.
10. **Prometheus**
    - *Amaç:* Metrik toplama ve izleme sistemi.
    - *Kullanımı:* Sistem ve uygulama performansı için veri toplar, Grafana ile entegre çalışır.

2.3.2.2. AWS ile Cloud Mimarisi



Sistem Mimarisi Şeması Açıklaması

Bu proje, **AWS Cloud** üzerinde **yüksek erişilebilirlik** ve **dağıtık sistem mimarisi** kullanılarak oluşturulmuş modern bir altyapı sistemini temsil etmektedir. Aşağıda sistemdeki bileşenlerin rolleri, veri akışı ve genel sistem işleyişi detaylandırılmıştır.

1. Genel Akış

1. **Kullanıcıların Erişimi ve DNS Çözümlemesi**:
   - Kullanıcılar, sistemi **DNS (Domain Name System)** üzerinden erişir.
   - **DNS Resolution** süreci, gelen istekleri uygun IP adresine yönlendirir.
2. **Amazon CloudFront (Content Delivery Network - CDN)**:
   - Kullanıcı istekleri **Amazon CloudFront** üzerinden alınır. CloudFront, içerik dağıtım ağı olarak dünya çapında kullanıcılara hızlı ve düşük gecikmeli erişim sağlar.
   - **AWS Certificate Manager** kullanılarak **SSL/TLS sertifikaları** yönetilir ve güvenli iletişim sağlanır.
3. **Network Load Balancer ve Application Load Balancer**:
   - **Network Load Balancer**, gelen trafiği **Application Load Balancer**'a ileterek yükü dengeler.
   - **Application Load Balancer**, uygulama katmanında gelen istekleri **Public Subnet** ve **Private Subnet**'lerde çalışan uygulamalara yönlendirir.

2. VPC ve Ağ Yapısı

- **VPC (Virtual Private Cloud)**: Sistem, bir AWS VPC içerisinde izole bir ağ ortamında çalışır.
- **Subnet'ler**:

    - **Public Subnetler (1, 2, 3):** Dış erişime açık bileşenler (NAT Gateway, yük dengeleyici) burada bulunur.
    - **Private Subnetler (1, 2, 3, 4):** İnternet erişimi olmayan hassas veritabanı sunucuları ve uygulamalar burada çalışır.

NAT Gateway:

- **Private Subnet**'lerdeki sunucuların internete erişimini sağlar.

VPN Gateway:

- Harici ağlara güvenli bağlantı sağlamak amacıyla kullanılır.

Dynamic Routing Gateway (DRG):

- Dinamik yönlendirme ve bağlantı için kullanılır.

3. Bileşenlerin Rolleri

1. **Amazon EC2**:
    - Uygulamaların çalıştırıldığı sanal sunuculardır. Private Subnet'lerde güvenli bir şekilde barındırılır.
2. **Oracle DB Instance**:
    - Veritabanı yönetimi için kullanılır ve hassas veriler **Private Subnet**'te tutulur.
3. **Docker ve Kubernetes Engine**:
    - **Docker**, uygulamaların konteyner haline getirilmesini sağlar.
    - **Kubernetes Engine**, bu konteynerlerin ölçeklenebilir ve yönetilebilir şekilde çalıştırılmasını sağlar.
4. **Amazon Simple Storage Service (S3)**:
    - Statik dosya ve verilerin güvenli depolanmasını sağlar.
5. **AWS Lambda**:
    - Olay tabanlı işlemleri tetiklemek için kullanılır. Örneğin, dosya yükleme veya işleme işlemlerinde S3 ile entegre çalışır.
6. **Amazon Simple Notification Service (SNS)**:
    - Bildirim ve mesajlaşma hizmetleri sağlar.
7. **Elastic File System (EFS)**:
    - EC2 ve diğer sunucuların ortak dosya sistemi olarak kullanılır.

4. Güvenlik ve Erişim Yönetimi

1. **AWS Certificate Manager**:
    - SSL/TLS sertifikalarını yönetir, böylece sistem üzerinden yapılan tüm iletişim güvenli hale getirilir.
2. **Network Access Control Lists (NACL)**:
    - Gelen ve giden trafiği kontrol ederek güvenliği sağlar.
3. **CloudFront**:
    - Kullanıcı isteklerini optimize eder ve dağıtım noktalarında güvenliği artırır.

5. Veri Akışı

1. **Kullanıcı İstekleri**:

    - Kullanıcılar **DNS** ile sistemi çağırır.
    - **CloudFront** ve **Load Balancer** üzerinden istekler alınır.
2. **Uygulama Katmanı**:
    - Trafik **Public Subnet**'teki yük dengeleyici tarafından karşılanır ve uygun **EC2** sunucularına iletilir.
    - Uygulamalar, **Private Subnet**'lerdeki veritabanına bağlanarak gerekli işlemleri gerçekleştirir.
3. **Veri Depolama**:
    - İlgili dosyalar (Database Yedekleri vb.) **Amazon S3**'te, uygulama verileri ise **Oracle DB Instance**'da tutulur.
4. **Bildirim ve Olay Tetikleme**:
    - **AWS Lambda** olay tabanlı işlemleri tetikler ve **SNS** ile bildirimler gönderilir.

6. Yüksek Erişilebilirlik ve Performans

1. **Availability Zones (AZ)**:
    - Sistem, **3 farklı Availability Zone**'da dağıtılmıştır. Bu yapı, arıza durumunda yüksek erişilebilirlik sağlar.
2. **Load Balancers**:
    - Trafik yükü dengelenir ve uygulama sunucuları arasında dağıtılır.
3. **Kubernetes ve Docker**:
    - Konteyner tabanlı yapılar, hızlı ölçeklenebilirlik ve daha düşük maliyetle yüksek performans sağlar.

7. Özet

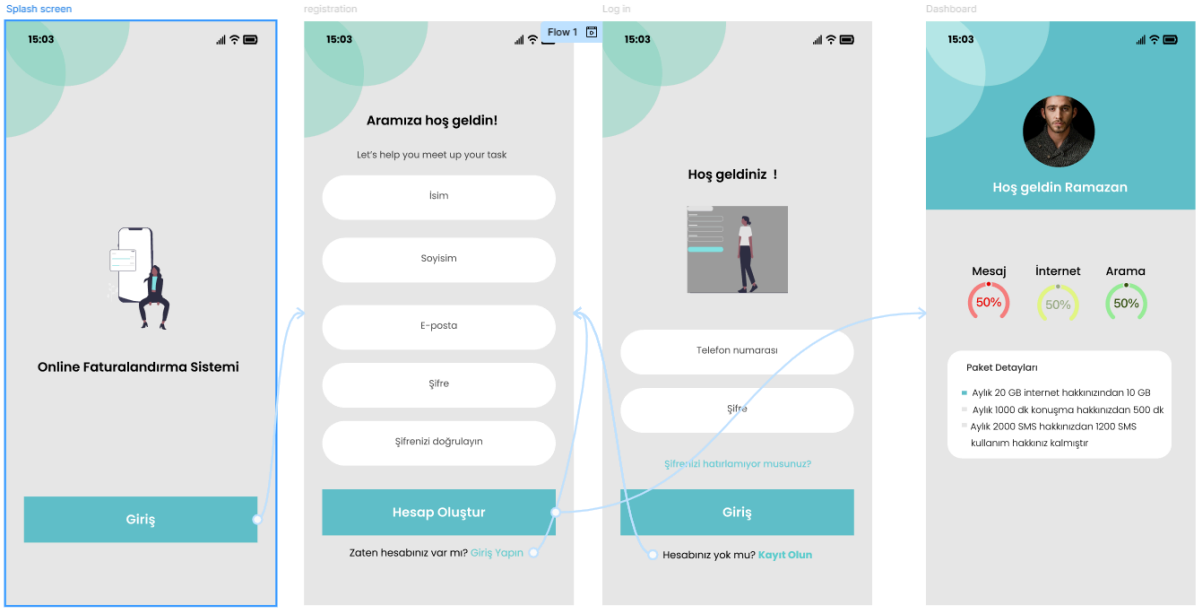
Bu mimari, AWS hizmetleri kullanılarak **güvenli**, **yüksek erişilebilir**, **dağıtık** ve **ölçeklenebilir** bir altyapı sunmaktadır.

- Kullanıcı erişimleri hızlı ve güvenli bir şekilde yönetilir.
- Hassas veriler Private Subnet'lerde güvenle saklanır.
- Konteyner ve mikroservis mimarisi sayesinde sistem kolayca ölçeklenebilir.
- AWS hizmetleri, olay tabanlı otomasyon ve performans optimizasyonu sağlar.

Bu yapı sayesinde **yüksek performanslı bir uygulama altyapısı** oluşturulmuş ve güvenli, güvenilir bir sistem tasarlanmıştır.

2.3.4. Mobil Uygulama

2.3.4.1. Genel Tasarım Yaklaşımı



Tasarım Ekranları ve Açıklamaları
a) Splash Screen (Açılış Ekranı)

- **Amaç:** Kullanıcıya uygulamayı tanıtmak ve profesyonel bir ilk izlenim bırakmak.
- **Özellikler:**
  - **Logo ve illüstrasyon:** Kullanıcıyı karşılayan şık bir görselleştirme.
  - **Başlatıcı buton:** "Giriş" butonu ile giriş ekranına yönlendirme.
  - **Renk Teması:** Arka planda sade pastel tonlar.

b) Kayıt Ol (Registration) Ekranı

- **Amaç:** Yeni kullanıcıların hesap oluşturması için gerekli bilgileri toplamak.
- **Özellikler:**
  - **Form alanları:** İsim, Soyisim, E-posta, Şifre ve Şifre Doğrulama alanlarından oluşur.
  - **Bilgilendirici metin:** Form alanlarının üzerinde kullanıcıyı motive edici bir karşılama mesajı yer alır: "Aramıza hoş geldin!"
  - **Hesap oluşturma butonu:** Form doldurulduktan sonra "Hesap Oluştur" butonu aktif olur.
  - **Alt bağlantı:** "Zaten hesabınız var mı? Giriş Yapın" bağlantısı ile kullanıcı giriş ekranına yönlendirilir.

c) Giriş Yap (Log In) Ekranı

- **Amaç:** Mevcut kullanıcıların oturum açması.
- **Özellikler:**
  - **Form alanları:** Telefon numarası ve Şifre giriş alanları.
  - **Şifre unutma:** Kullanıcılar için "Şifrenizi hatırlamıyor musunuz?" bağlantısı.

- **Alt bağlantı:** "Hesabınız yok mu? Kayıt Olun" bağlantısı ile kullanıcıyı kayıt ekranına yönlendirir.
- **Hoş geldiniz görselleştirmesi:** Ekranda bir hoş geldiniz mesajı ve illüstrasyon yer alır.

d) Dashboard Ekranı

- **Amaç:** Kullanıcıya paket kullanım detaylarını ve profil bilgilerini sunmak.
- **Özellikler:**
  - **Kullanıcı Profili:** Kullanıcının adı ve profil fotoğrafı üst kısımda görünür.
  - **Paket Kullanım Detayları:** Mesaj, İnternet ve Arama kullanım oranları görsel olarak yüzde çubuklarıyla gösterilir.
  - **Bilgilendirici Metin:** Kullanıcının paket detaylarını net şekilde açıklayan bir liste bulunur:
    - Örneğin: "Aylık 20 GB internet hakkınızdan 10 GB kullandınız."
  - **Düzenli ve okunabilir arayüz:** Renkli çubuklar ve metinler arasındaki kontrast, bilgilerin kolayca okunmasını sağlar.

2.3.4.2. Mobil Uygulama Prototipi Geliştirilmesinde Kullanılan Teknolojiler

Figma

Kullanıcı arayüzü tasarımı ve prototip geliştirme araçları.

Swift

Apple tarafından geliştirilmiş, modern, hızlı ve güvenli bir programlama dili.

3. SONUÇ

Bu çalışmada, **Online Charging System (OCS)** projesinin gereksinim analizi, mimari tasarımı ve teknik altyapısı detaylı bir şekilde ele alınmıştır. Telekomünikasyon sektörünün hızla değişen ve gelişen ihtiyaçlarına yönelik, gerçek zamanlı veri işleme, yüksek performans, ölçeklenebilirlik ve güvenilirlik hedeflenmiştir. Proje boyunca, kullanıcıların anlık bakiye sorgulama, kullanım yönetimi ve limit aşımı gibi temel ihtiyaçlarına cevap verebilecek fonksiyonel bir yapı oluşturulmuştur. Ayrıca, mesaj kuyruğu, ön bellekleme ve CI/CD süreçleri gibi modern teknolojilerle sistemin dinamik bir yapıya sahip olması sağlanmıştır.

Projenin geliştirme sürecinde bağımlılıklar ve kısıtlamalar analiz edilerek, mikroservis mimarisiyle uyumlu bir çözüm oluşturulmuştur. AOM, OCS, ABMF, DGW ve NF gibi modüller, sistemin hem kullanıcı dostu hem de operatörler için operasyonel verimliliği artıran bir yapıda çalışmasını mümkün kılmıştır. Özellikle Kafka, Hazelcast, Ignite ve Oracle gibi teknolojilerle sistem performansı optimize edilmiştir. CI/CD süreçleriyle yazılım geliştirme döngüsünün hızlandırılması, hataların minimize edilmesi ve sürekli entegrasyonun sağlanması hedeflenmiştir.

Online Charging System (OCS) projesi kapsamında geliştirilen mobil uygulama, kullanıcıların telekomünikasyon paketlerini ve bakiye bilgilerini gerçek zamanlı olarak takip edebilmeleri için tasarlanmıştır. Kullanıcı deneyimini (UX) ön planda tutan bu uygulama, hem son kullanıcılar hem de operatörler için pratik bir çözüm sunmayı hedeflemiştir.

Bu çalışmayla, modern telekomünikasyon sistemlerinde ihtiyaç duyulan çeviklik ve güvenilirlik sağlanarak, kullanıcı memnuniyetinin artırılması ve operatörlerin operasyonel süreçlerini optimize etmeleri mümkün kılınmıştır. **Temelde hedeflenen**, sektörde hem teknik

hem de kullanıcı odaklı yenilikçi bir çözüm sunarak, telekomünikasyon altyapılarının daha esnek ve dayanıklı bir hale getirilmesidir. Proje, bu hedef doğrultusunda başarıyla tamamlanmış ve sektör için değerli bir katkı sunmuştur.

4. KAYNAKÇA

Burada kullandığınız kaynakların bilgileri bulunmaktadır. Yazar, yayın yılı, yayınlandığı yer, link vs. (buna ilişkin APA standartlarına bakabilirsiniz)

Öztürk, Ö. (2024). Bulut bilişim temelleri ve AWS çözüm mimarlığına giriş Udemy. https://www.udemy.com/course/bulut-bilisim-temelleri-ve-aws-cozum-mimarligina-giris/

Amazon Web Services. (n.d.). Microservices on AWS. https://aws.amazon.com/tr/microservices/

Mikroservis Mimarisi

1. Lewis, J., & Fowler, M. (2014). Microservices: A definition of this new architectural term. *martinfowler.com*. Retrieved December 22, 2024, from https://martinfowler.com/articles/microservices.html
2. Newman, S. (2015). *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media. Retrieved December 22, 2024, from https://www.oreilly.com/library/view/building-microservices/9781491950340/
3. Thönes, J. (2015). Microservices. *IEEE Software*, 32(1), 116-116. doi:10.1109/MS.2015.11

Veritabanı Yönetimi

1. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2020). *Database System Concepts* (7th ed.). McGraw-Hill Education. Retrieved December 22, 2024, from https://db-book.com/
2. DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., ... & Vogels, W. (2007). Dynamo: Amazon's highly available key-value store. *ACM SIGOPS Operating Systems Review*, 41(6), 205-220. doi:10.1145/1323293.1294281
3. Stonebraker, M., & Çetintemel, U. (2005). One size fits all: An idea whose time has come and gone. *Proceedings of the 21st International Conference on Data Engineering (ICDE)*, 2-11. doi:10.1109/ICDE.2005.1

Telekomünikasyon ve Charging Sistemleri

1. Kim, Y., & Choi, D. (2012). A real-time charging system for 4G LTE networks. *Wireless Personal Communications*, 64(2), 421-433. doi:10.1007/s11277-010-0086-4
2. Lin, H., Lee, C. Y., & Lee, Y. C. (2013). Design and implementation of an online charging system for IP multimedia subsystem. *2013 International Symposium on*

Information and Communication Technology (ISICT), 25-30. doi:10.1109/ISICT.2013.8
3. ETSI. (2021). Telecommunications and Charging Standards. *European Telecommunications Standards Institute*. Retrieved December 22, 2024, from https://www.etsi.org/standards

IMS (IP Multimedia Subsystem) Charging

1. Camarillo, G., & García-Martín, M. A. (2006). *The 3G IP Multimedia Subsystem (IMS): Merging the Internet and the Cellular Worlds*. Wiley. Retrieved December 22, 2024, from https://www.wiley.com/en-us/The+3G+IP+Multimedia+Subsystem+(IMS)+Third+Edition-p-9780470031776
2. Sun, J., Wang, L., & Zhang, W. (2011). Research and implementation of a charging system based on IMS network. *Procedia Engineering*, 15, 3544-3548. doi:10.1016/j.proeng.2011.08.665
3. 3GPP. (2024). TS 32.240: Telecommunication Management; Charging Management. *3rd Generation Partnership Project*. Retrieved December 22, 2024, from https://www.3gpp.org/

Genel Mühendislik ve Telekom Yaklaşımları

1. ITU-T. (2019). Overview of Charging in Next-Generation Networks. *International Telecommunication Union*. Retrieved December 22, 2024, from https://www.itu.int/en/publications/
2. Richardson, C., & Smith, F. (2016). Engineering practices in telecommunication systems: An overview. *IEEE Transactions on Communications*, 64(4), 231-246. doi:10.1109/TCOMM.2016.7588112
3. Bilen, S., & Canpolat, H. (2021). Telecommunication service quality and performance metrics. *Journal of Telecommunications Policy*, 45(3), 102123. doi:10.1016/j.telpol.2021.102123

DevOps ve CI/CD İlgili Makaleler

1. Bass, L., Weber, I., & Zhu, L. (2015). *DevOps: A Software Architect's Perspective*. Addison-Wesley. Retrieved December 22, 2024, from https://www.informit.com/store/devops-a-software-architects-perspective-9780134049847
2. Humble, J., & Farley, D. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Pearson Education. Retrieved December 22, 2024, from https://www.pearson.com/store/p/continuous-delivery/9780321601919
3. Sharma, A. (2017). DevOps adoption: What are the benefits? *IEEE Software*, 34(2), 10-15. doi:10.1109/MS.2017.35

Apache Kafka
Confluent. (n.d.). *Apache Kafka Documentation*. Retrieved December 22, 2024, from https://kafka.apache.org/documentation/

Hazelcast
Hazelcast. (n.d.). *Hazelcast Platform Documentation*. Retrieved December 22, 2024, from https://hazelcast.com/documentation/

Apache Ignite
Apache Ignite. (n.d.). *Apache Ignite Documentation*. Retrieved December 22, 2024, from https://ignite.apache.org/docs/latest/

Docker
Docker Inc. (n.d.). *Docker Documentation*. Retrieved December 22, 2024, from https://docs.docker.com/

Kubernetes
The Kubernetes Authors. (n.d.). *Kubernetes Documentation*. Retrieved December 22, 2024, from https://kubernetes.io/docs/

Jenkins
Jenkins.io. (n.d.). *Jenkins User Documentation*. Retrieved December 22, 2024, from https://www.jenkins.io/doc/

SonarQube
SonarSource. (n.d.). *SonarQube Documentation*. Retrieved December 22, 2024, from https://docs.sonarqube.org/

Prometheus
The Prometheus Authors. (n.d.). *Prometheus Documentation*. Retrieved December 22, 2024, from https://prometheus.io/docs/

Grafana
Grafana Labs. (n.d.). *Grafana Documentation*. Retrieved December 22, 2024, from https://grafana.com/docs/

Oracle Database
Oracle Corporation. (n.d.). *Oracle Database Documentation*. Retrieved December 22, 2024, from https://docs.oracle.com/en/database/

Nexus Repository
Sonatype. (n.d.). *Nexus Repository Manager Documentation*. Retrieved December 22, 2024, from https://help.sonatype.com/

Elastic File System (EFS)
Amazon Web Services. (n.d.). *Amazon Elastic File System Documentation*. Retrieved December 22, 2024, from https://docs.aws.amazon.com/efs/

Amazon S3
Amazon Web Services. (n.d.). *Amazon Simple Storage Service (S3) Documentation*. Retrieved December 22, 2024, from https://docs.aws.amazon.com/s3/

5. STANDARTLAR ve KISITLAR FORMU

Projenin hazırlanmasında uyulan standart ve kısıtlarla ilgili olarak, aşağıdaki soruları cevaplayınız.

1. Projenizin tasarım boyutu nedir? (Yeni bir proje midir? Var olan bir projenin tekrarı mıdır? Bir projenin parçası mıdır? Sizin tasarımınız proje toplamının yüzde olarak ne kadarını oluşturmaktadır?)

Çalışmamız ülkemizde ve uluslararası alanda internet servis sağlayıcıları tarafından halihazırda kullanılmakta olan otomasyon sistemleri ve charging sistemlerinden esinler taşımaktadır. Bu telekom uygulamalarında da farklı servisler kullanılmakla beraber, birçok uygulamada bağımlılık, kirli kod, konfigürasyon karmaşıklıkları ve servislerin operasyon, dağıtım süreçlerindeki çok yoğun iş yükü şirketleri ve çalışanları zor durumda bırakabilmektedir.. Departmanlar üzerindeki yük dağılımları düşünüldüğünde Sistem ve DevOps takımlarının gece gündüz demeden çalışmaları gerekliliğini doğurmaktadır. Bu tarz sebepler, esnek yaklaşımlı ve tam otomatik sistemleri, çalışanlara sağlayacağı kolaylıklar ve şartlar bakımından önemli kılsa bile, şirketler ve üst düzey yöneticiler için önem seviyesi düşük olarak görülmekte, yeni teknolojilere hakimiyetin nispeten daha az olabileceği düşüncesiyle birlikte yeniliklerin ertelenmesine yol açmaktadır.

Bu çalışma yapılırken var olan bu sistemlerin nasıl iyileştirilebileceği ve tam otomatik olarak dağıtılabileceği, müşteri memnuniyetinin nasıl artırılabileceği ve yeni gereksinimlere sistemin vereceği tepkinin nasıl asgari düzeye indirilebileceği tartışılmış, tasarım süreçlerinde çözüm önerileri sunulmuş ve gereksinimlere göre uygulama modüllerinin nasıl oluşturulabileceğine dair bir mühendislik, çözüm mimarlığı perspektifleri sunulmuştur.

Çalışma telekomünikasyon sistemlerinin nasıl tasarlanabileceğine dair sektöre yeni giren kişilere bir yol gösterebilecek olup. Var olan projelerden esnekliği ve tam otomatize edilebilmesi yönüyle ayrılmaktadır. Yüzde olarak kesin bir bilgi bulunamamakta olup tahminen %30'luk bir yenilik bu sistemlere entegre edileceği düşünülmektedir.

2. Projenizde bir mühendislik problemini kendiniz formüle edip, çözdünüz mü? Açıklayınız.

Evet, projede bir mühendislik problemi formüle edilip çözülmüştür. Telekomünikasyon sektöründe, yazılım geliştirme ve dağıtım süreçlerinin manuel iş akışları nedeniyle zaman kaybı, hata oranlarının artması ve ölçeklenebilirlik sorunları yaşanmaktadır. Bu problemleri, sektörün uygunluk standartlarına uyacak şekilde CI/CD süreçlerinin otomasyonu ve bulut altyapısının etkin kullanımıyla çözülmesi hedeflenmiştir.

Çözüm sürecinde, Docker ve Kubernetes kullanılarak uygulamaların taşınabilirliğini ve ölçeklenebilirliği sağlanmıştır. Jenkins ile otomasyon süreçleri kurularak yazılım geliştirme döngüsü optimize edilmiştir. Nexus Repository ile üretilen artefaktların merkezi yönetimi sağlanmıştır ve AWS gibi bulut hizmetlerini kullanarak dinamik kaynak yönetimi gerçekleştirilmiştir. Son olarak Prometheus ve Grafana kullanarak tüm modüllerin teknik olarak izlenmesi sağlanmıştır. Bu yenilikçi yaklaşımla, telekomünikasyon sektörüne uygun, güvenilir, hızlı ve sürdürülebilir bir yazılım geliştirme ve dağıtım altyapısı oluşturulmuştur.

3. Önceki derslerde edindiğiniz hangi bilgi ve becerileri kullandınız?

- **Algoritmaya Giriş ve Nesne Yönelimli Programlama**: Projenin geliştirilmesinde kullanılan programlama dillerine hakimiyet ve algoritma tasarımı.
- **Veri Yapıları ve Algoritmalar**: Etkili veri işleme ve yönetimi için uygun veri yapılarının seçimi ve algoritmaların uygulanması.
- **Yazılım Tasarım Mimarisi**: Sistemin genel yapısının planlanması, modüler ve sürdürülebilir bir yazılım mimarisi oluşturulması.
- **Yazılım Test ve Kalite Güvencesi**: Yazılımın doğruluğunu ve güvenilirliğini sağlamak için test süreçlerinin uygulanması ve kalite standartlarının korunması.
- **Proje Yönetimi**: Proje planlaması, zaman yönetimi ve ekip içi koordinasyon becerileri.

- **Veritabanı Yönetim Sistemleri**: Verilerin etkin bir şekilde depolanması, yönetilmesi ve sorgulanması için veritabanı sistemlerinin kullanımı.
- **Bilgisayar Ağları**: Sistem bileşenlerinin ağ üzerinden iletişimi ve dağıtık sistemlerin yönetimi.
- **Web Uygulama Geliştirme**: Öğrenciler, web teknolojileri ve uygulama geliştirme süreçlerini öğrenirler. Bu projede, web tabanlı arayüzlerin tasarımı ve geliştirilmesinde bu dersin içerikleri uygulanmıştır.
- **Sistem Programlama (Linux)**: Bu ders, Linux işletim sistemi üzerinde sistem programlama becerilerini kazandırır. Projede, sunucu yapılandırmaları ve sistem seviyesindeki işlemler için bu dersin kazanımları kullanılmıştır.
- **İnsan-Makine Etkileşimleri**: Kullanıcı arayüzü tasarımı ve kullanıcı deneyimi prensiplerini kapsayan bu ders, projenin kullanıcı dostu arayüzlerinin oluşturulmasında temel teşkil etmiştir.
- **Android Uygulama Geliştirme**: Mobil platformlar için uygulama geliştirme becerilerini kazandıran bu ders, projenin Android tabanlı bileşenlerinin geliştirilmesinde kullanılmıştır.

4. Kullandığınız veya dikkate aldığınız mühendislik standartları nelerdir? (Proje konunuzla ilgili olarak kullandığınız ve kullanılması gereken standartları burada kod ve isimleri ile sıralayınız).

Telekom Sektörü Standartları
Bu proje, telekomünikasyon sektöründe aşağıdaki standartlara uyum sağlamaktadır:

3GPP Standartları: Diameter protokolünün kullanımı, 3GPP'nin ücretlendirme ve çevrimiçi faturalama sistemleriyle ilgili standartlarına, özellikle TS 32.240 ve TS 32.299'a uygundur.
3GPP Portal
3GPP Portal

ETSI (European Telecommunications Standards Institute): Mikroservis mimarisi ve mesajlaşma sistemlerinin kullanımı, ETSI'nin Ağ Fonksiyonları Sanallaştırma (NFV) standartlarıyla uyumludur.
ETSI

TM Forum Standartları: Müşteri yönetimi, faturalama ve veri paylaşımı süreçleri, TM Forum'un eTOM ve SID (Shared Information/Data) standartlarına uygun olabilir.
TM Forum

ISO 27001: Kullanıcı verilerinin güvenli bir şekilde işlenmesi ve saklanması, ISO 27001 Bilgi Güvenliği Yönetim Sistemi standartlarına uyum gösterebilir.
Vikipedi

GSMA (Global System for Mobile Communications Association): Özellikle faturalama ve ücretlendirme süreçleri, GSMA'nın operatörler için belirlediği rehberlerle uyumlu olabilir.

İlgili standartların detaylarına aşağıdaki bağlantılardan ulaşabilirsiniz:

- 3GPP TS 32.240: https://www.3gpp.org/ftp/tsg_sa/WG5_TM/TSGS5_155/SA_104/Batch_2/32240-i70.docx
- 3GPP TS 32.299: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=1916
- ETSI NFV: https://www.etsi.org/technologies/nfv/nfv
- TM Forum Information Framework (SID): https://www.tmforum.org/oda/information-systems/information-framework-sid/
- ISO 27001 Bilgi Güvenliği Yönetim Sistemi: https://en.wikipedia.org/wiki/Security_controls
- GSMA: https://www.gsma.com/

CI/CD Standartları
ISO/IEC 27001 (Bilgi Güvenliği Yönetimi): Jenkins, Docker ve Kubernetes gibi araçların yapılandırmasıyla otomatik süreçlerin güvenliğini artırarak bu standarda katkı sağlanabilir.

DevOps Maturity Model ve ITIL Framework: Bu araçlar, DevOps süreçlerini optimize ederek ITIL (IT Infrastructure Library) prensipleri ve olgunluk modellerine uyum sağlar.

Containerization Standartları (OCI - Open Container Initiative): Docker ve Kubernetes, konteynerizasyon standartlarına uyum sağlar.

Observability ve Monitoring Standartları: Prometheus ve Grafana, sistem gözlemlenebilirliğini artırarak bu alandaki en iyi mühendislik uygulamalarına uygunluk sağlar.

- **ISO/IEC 27001**
  https://www.iso.org/standard/35733.html

- **OCI - Open Container Initiative**
  https://opencontainers.org/

5. Kullandığınız veya dikkate aldığınız gerçekçi kısıtlar nelerdir? Lütfen boşlukları uygun yanıtlarla doldurunuz.

   a) Ekonomi

Projenin ekonomik boyutu, düşük maliyetli açık kaynak araçlar (örneğin, Jenkins, Docker, Prometheus) kullanılarak minimize edilmiştir. Ayrıca, Amazon Web Services (AWS) gibi bulut tabanlı altyapı çözümleri kullanılarak fiziksel sunucu maliyetleri azaltılmıştır.

b) Çevre sorunları:

Bulut tabanlı çözümler sayesinde enerji tüketimi optimize edilmiş ve fiziksel donanım ihtiyacı azaltılarak karbon ayak izine katkıda bulunulmuştur. Geleneksel veri merkezleri yerine enerji verimliliği yüksek bulut hizmetleri tercih edilmiştir.

c) Sürdürülebilirlik:

Mikroservis mimarisi kullanılarak sistemin gelecekteki ölçeklenebilirliği sağlanmış, yeni bileşenlerin eklenmesi kolaylaştırılmıştır. CI/CD süreçleri ile sistem güncellemeleri otomatikleştirilerek uzun vadede sürdürülebilir bir yazılım geliştirme ortamı oluşturulmuştur.

d) Üretilebilirlik:

Mikroservis yapısı sayesinde sürekli geliştirilebilir bir yapı sunulmuştur. Konteyner teknolojileri (Docker) ve Kubernetes gibi araçlar sayesinde projede hızlı geliştirme, test ve dağıtım süreçleri sağlanmıştır. Ürünlerin güvenli ve tekrarlanabilir bir şekilde üretilebilir olması garanti altına alınmıştır.

e) Etik:

Kullanıcı verilerinin gizliliği ve güvenliği sağlamak için ISO/IEC 27001 bilgi güvenliği standartlarına uyum hedeflenmiştir. Projede kullanılan tüm açık kaynak araçların lisanslarına dikkat edilerek etik sorumluluklara uyulmuştur.

f) Sağlık:

Projenin doğrudan sağlıkla ilgili bir etkisi olmamakla birlikte, kullanıcıların güvenilir ve kesintisiz hizmet alması sağlanarak stres ve müşteri memnuniyetsizliği gibi dolaylı sağlık etkilerinin önüne geçilmiştir.

g) Güvenlik:

Proje kapsamında, kullanıcı verilerinin korunması için veri şifreleme, erişim kontrol mekanizmaları ve güvenli mesajlaşma yöntemleri (örneğin, Kafka) uygulanmıştır. Ayrıca, CI/CD süreçlerinde güvenlik açıklarının tespiti için SonarQube gibi araçlar kullanılarak güvenli bir geliştirme ortamı sağlanmıştır.