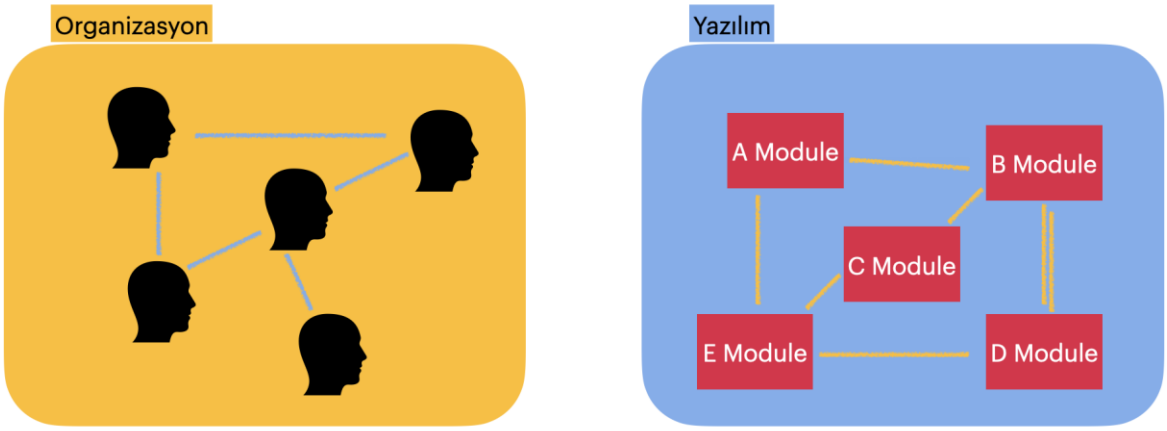
1- Conway’s Law

Bu yasa **temelinde** yazılım modülleri arasındaki ilişkiye benzer bir yapının, organizasyon yapısında olduğu ve buna parallellik gösterdiği üzerinedir. Sistem yapısındaki yazılım arayüzleri, organizasyonlar da bulunan sosyal sınırlar ile benzer şekilde çalışır ve iletişim yöntemleri içerir. Conway sosyolojik gözlemle bu yasayı ortaya koymuş. 1968 National Symposium bu Conway Yasası olarak adlandırılmaya başlanmıştır.




2- SaaS, PaaS, IaaS, On-Premises

*SaaS: Software as a service — hizmet olarak yazılım.*

Kullandığınız herhangi bir yazılımı satın alarak, çalışması için ihtiyacı olan tüm kaynakları sizin sağladığınız bir altyapı yerine, yazılım üreticisinin “hizmetine” dönemsel abone olarak kullanım sağladığınız bir dağıtım modelidir.

PaaS: Platform olarak bir Hizmet Olarak Platform (PaaS) olarak da bilinen bulut platformu hizmetleri, temel olarak uygulamalar için kullanılırken, belirli yazılımlara bulut bileşenleri sağlar. PaaS, geliştiriciler için özelleştirilmiş uygulamalar oluşturmak için kullanabilecekleri ve kullanabilecekleri bir çerçeve sunar. Tüm sunucular, depolama ve ağ iletişimi kurum veya üçüncü taraf bir sağlayıcı tarafından yönetilirken, geliştiriciler uygulamaların yönetimini koruyabilir.
Şirketinizin büyüklüğü ne olursa olsun, PaaS kullanmak aşağıdakiler dahil birçok avantaj sunar:

- Basit, uygun maliyetli geliştirme ve uygulamaların dağıtımı
- Ölçeklenebilir
- Son derece kullanılabilir
- Geliştiriciler, yazılımı sürdürmenin baş ağrısı olmadan uygulamaları özelleştirebilir
- Gereken kodlama miktarında önemli azalma
- İşletme politikasının otomasyonu
- Model Hibrit modele kolay geçiş

IaaS: Hizmet Olarak Altyapı

- Hizmet Olarak Altyapı (IaaS) olarak bilinen bulut altyapısı hizmetleri, ölçeklenebilir ve otomatik bilgi işlem kaynaklarından yapılır. IaaS, bilgisayarlara, ağlara, depolama alanlarına ve diğer servislere erişmek ve bunları izlemek için tamamen self servistir. IaaS işletmelerin donanım satın almak yerine talep üzerine ve ihtiyaç duydukları kaynakları satın almalarını sağlar.

IaaS aşağıdakiler dahil birçok avantaj sunar:

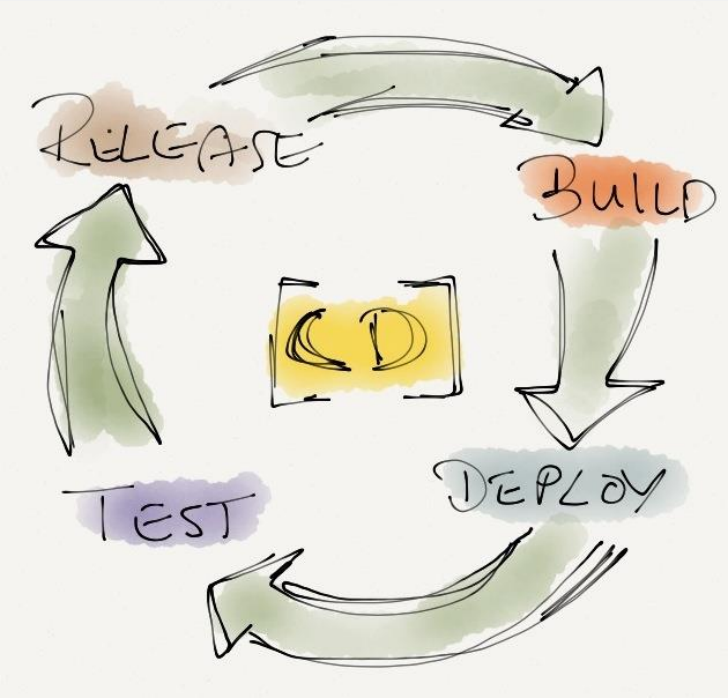
- En esnek bulut bilişim modeli
- Depolama, ağ, sunucu ve işlem gücü dağıtımını otomatikleştirmek kolaydır
- Depolama, ağ, sunucu ve işlem gücü dağıtımını otomatikleştirmek kolaydır
- Müşteriler, altyapılarının tam kontrolünü elinde tutuyor
- Müşteriler, altyapılarının tam kontrolünü elinde tutuyor
- Son derece ölçeklenebilir

**Continuous Integration, Continuous Delivery ve Continuous Deployment kavramlarını açıklayınız.**

**Continuous Integration:**Geliştirilmesi yapılan kodu ilgili repodan alıp,testlerden(unit,integration vs.) geçirip,paketleyip, test edilecek ortama kurulumunu otomatize etme sürecidir.

**Continuous Delivery:** Başarılı ile build alınmış yapıyı bir ortama atmanın otomatik bir yoludur.**Continuous Delivery işlemi manuel olarak yapılır.** Bu süreç düzgün bir şekilde uygulandığında, standartlaştırılmış bir test sürecinden geçmiş hazır yapıya sahip olunacaktır.

| Avantajları | Dezavantajları |
| --- | --- |
| ▪ Clienttın ihtiyacı doğrultusunda birden fazla servis tarafından üretilecek olan datayı tek bir request – response ile üretilmesini sağlayarak daha az maliyetli bir kullanıcı deneyimi ortaya koyabilir.<br>▪ Authentication, authorization, logging, security, routing vs. gibi cross cutting concern kavramlarının tek elden yönetilmesini sağlayabilir.<br>▪ Ve en önemlisi clientları, uygulamanın microservislere nasıl bölündüğü hususunda düşünmekten izole eder. | ▪ API gateway, ekstradan bir katman oluşturacağı için istek neticesinde işlevsel açıdan gözardı edilebilir bir farkla sürenin artmasına sebep olabilir.<br>▪ API gateway; geliştirici, dağıtım ve bakım gerektiren şahsına münhasır bir katmandır.<br>▪ Tüm servislere erişim api gateway üzerinden olduğu için herhangi bir çöküntü yahut kesinti durumunda tüm sistem aksaklığa uğrayabilir. |




**Continuous Deployment:**Sürecin son adımı olarak düşünebiliriz.CD ise CI'dan başarılı bir şekilde geçen sistemin ilgili yerlere dağıtım süreciyle ilgilenir.

**API Gateway pattern'ı açıklayınız.**

API Gateway, client ile sıkı bağlı(tightly coupled) bir bağ yarattığı için eleştirilir ancak avantajları bu dezavantajı ciddi manada gölgelemekte olduğundan bu durum göz ardı edilebilmektedir.

***API Gateway'in temel işlevi clienttan isteği alıp uygun servise iletmesidir.***

**Backend for frontend (BFF) pattern' ı açıklayınız.**

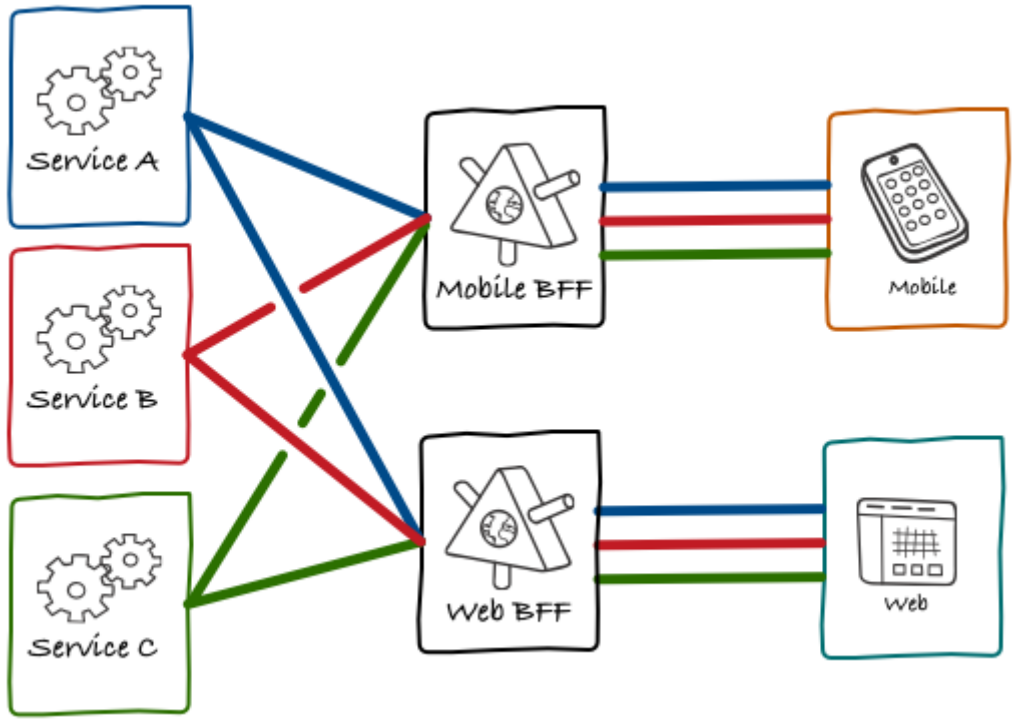
Her client için ayrı bir api-gateway tanımlanmasını konu alır. Yani yukarıdaki 3 durum için (web-ui,mobile-app,3rd app) 3 farklı BFF ile sorunlarımızı çözümlemeye çalışan pattern'e denir.

**BFF'in avantajları;**

- Her client için uygun api'yi sağlar.
- Olası problemlerin yönetimini kolaylaştırır.
- Her client için mikroservisleri bölümlere ayırır.
- Business logic'i mikroservislerden alarak tek bir nokta üzerine almamızı sağlar.

**BFF'in dezavantajları;**

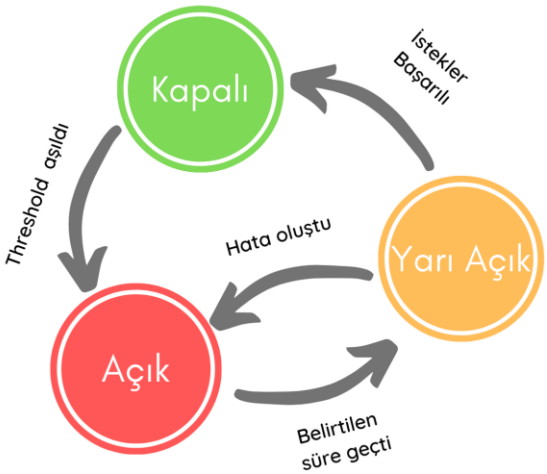
- Uygulama sayısının artmasıyla birlikte bakım ve operasyon maliyetlerinin artması
- Mimari karmaşıklığın artması
- Eklenecek ayrı bir katmanın ağ isteklerinde gecikme yaratma ihtimali
- BFF'lerde kod tekrarlarının yapılma ihtimali




**Circuit-breaker pattern' ı açıklayınız.**

**Circuit Breaker Pattern, bir yazılımda hataları tespit ederek hatanın tekrar etmemesini sağlar. Böylece sistemde hataların tekrar etmesi sonucu oluşacak bloklanma, hizmet kesintileri, aşırı kaynak kullanımı gibi sorunlarla karşılaşılmasını önleyerek, sistemin sağlıklı çalışmasını amaçlar.**

Yazılım sistemlerinde farklı servis veya sistemlere istekte bulunmak/çağrı yapmak oldukça sık kullandığımız bir yöntemdir. Microservice mimarisindeki servisler arası senkron iletişim de buna dahil edilebilir. Özellikle çoğu kurumsal yazılım sisteminde Saas, CRM, SAP gibi birçok sisteme entegre olma çoğu yazılımın kaderi haline geliyor.



**Microservice chassis pattern' ı kısaca açıklayınız.**