

HUBX

ANTI-PATTERNS IN JAVASCRIPT WORLD

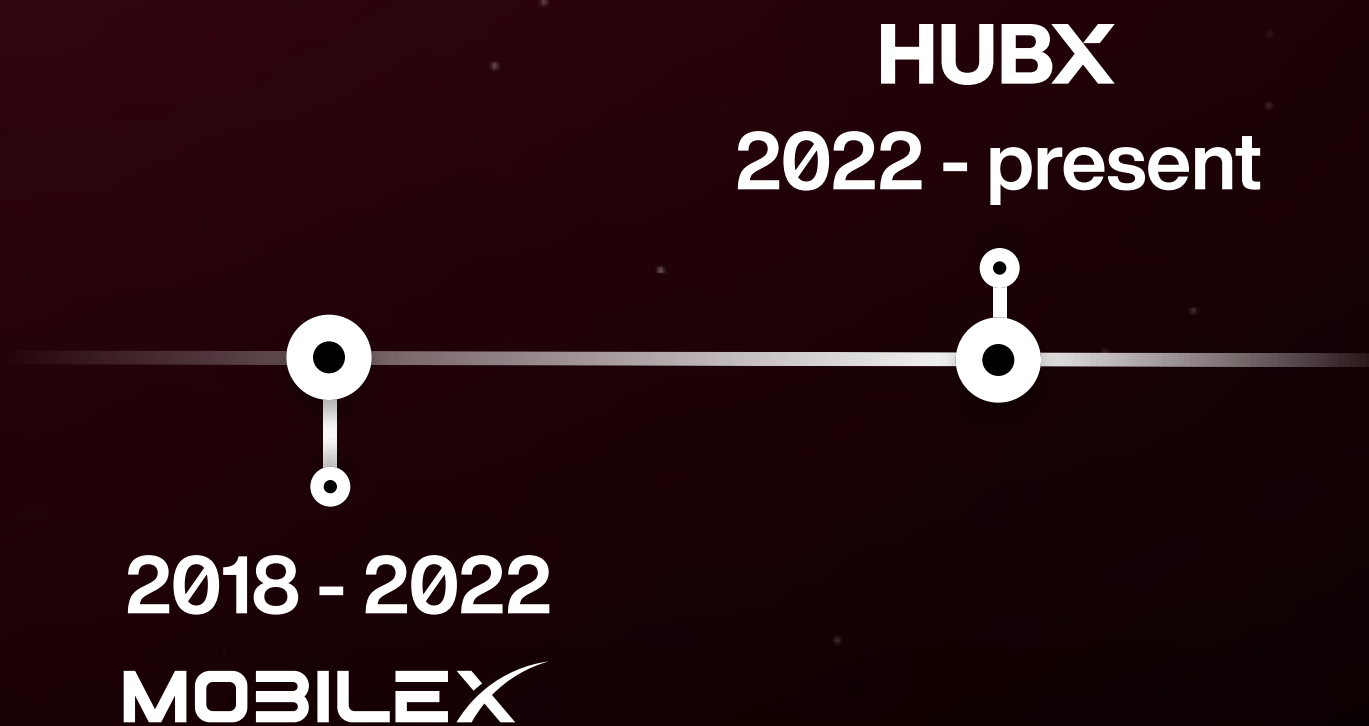
Atakan Şengöz



HUBX

Who is Atakan?

- ✓ A Venture Builder developing mobile applications across multiple verticals



- ✓ Software Engineer with 7 years experience
- ✓ IoT, Simulation, Embedded Systems, B2B, B2C
- ✓ Currently Sr. Backend Developer at HubX





TANIMLAR



Sıkça tekrarlanan ancak olumsuz sonuçlara yol açan bir tasarım veya uygulama çözümüdür.

Anti-pattern



Büyük yapıları küçük, bağımsız bir şekilde birbirine bağlı hizmetlere ayırma mimari yaklaşımıdır.

Mikro Servis



Olası bir tehdit veya zarardan kaynaklanan ve henüz gerçekleşmemiş olan bir tehlikenin olasılığıdır.

Potansiyel Risk



TANIMLAR

Single Point of Failure

Tek Noktadan Arıza

Over Communication

Aşırı İletişim

Database Sharing

Veritabanı Paylaşımı

Centralized Configuration

Merkezi Yapılandırma

Over-Specialized Services

Tek Boyutlu Görevler

Tight Coupling

Sıkı Bağlantı

Low Automation

Düşük Otomasyon

Spaghetti Architecture

Spagetti Mimarisi

Distributed Monolith

Dağıtılmış Monolit

Ignoring the Human Cost

İnsan Maliyetini Göz Ardı Etme

X

Single Point of Failure

Tek Noktadan Arıza

- ✓ Tüm servislerin tek bir veritabanına veya özel bir servise bağılı olması ve bu noktada meydana gelebilecek bir sorunun tüm sistem üzerinde yıkıcı etkilere yol açmasıdır.
- ✓ Bunu önlemek için; yedekleme, load balancer, health checker, decoupling ve modülerlik gibi yöntemler kullanılabilir.



X

Over Communication

Aşırı İletişim

- ✓ Servisler arasında aşırı miktarda senkron iletişimin olması sistemin performansını düşürebilir, yapıyı karmaşıktırabilir veya kaynaklarını aşırı kullanabilir.
- ✓ Bunu önlemek için; asenkron iletişim, Event-Driven mimari veya verimli Data Fetching gibi yöntemler kullanılabilir.



X

Database Sharing

Veritabanı Paylaşımı

✓ Farklı servislerin aynı veritabanı şemalarını veya tablolarını kullanması, servisler arasında istenmeyen bağımlılıklara yol açar ve her bir servisin bağımsız olarak geliştirilmesini ve ölçeklendirilmesini zorlaştırır.

✓ Bunu önlemek için; database encapsulation, servis tabanlı şemalar veya Command Query Responsibility Segregation(CQRC /okuma yazma dağıtımı) gibi yöntemler kullanılabilir.

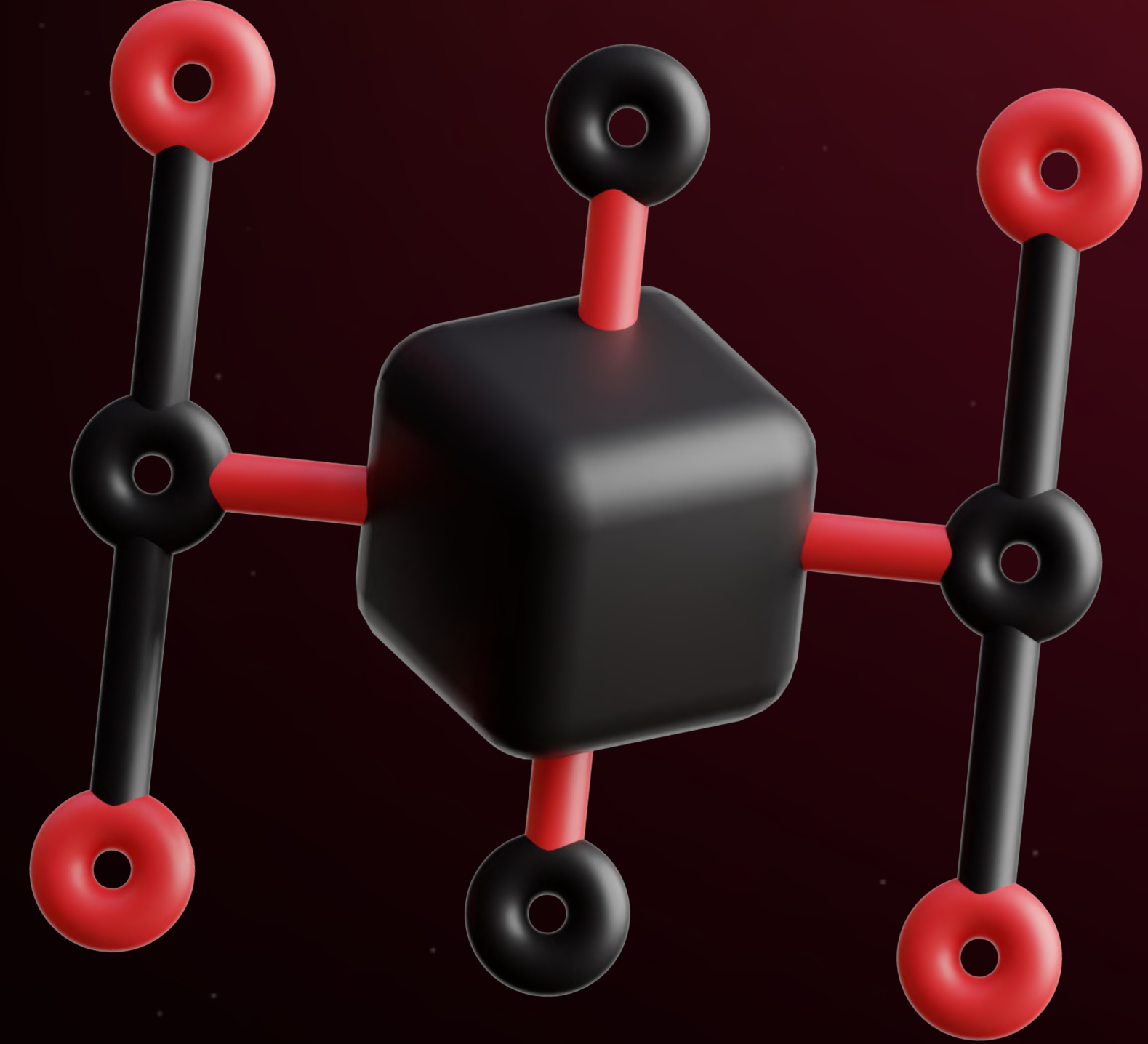


X

Centralized Configuration

Merkezi Yapılandırma

- ✓ Servislerin konfigürasyonlarını tek bir yerden yönetmesi ve bu yapıda bir hata halinde tüm sistemin etkilenmesidir.
- ✓ Bunu önlemek için; dağıtık konfigürasyonlar kullanma, feature flag, versiyonlama veya config şifreleme gibi yöntemler kullanılabilir.



X

Over-Specialized Services

Tek Boyutlu Görevler

- ✓ Her servisin çok spesifik bir fonksiyonu yerine getirmesi, sistemlerin yönetilmesini ve anlaşılmasını zorlaştırabilir. Çok fazla servis oluşması, entegrasyon ve ağ yükü gibi sorunlara yol açabilir.
- ✓ Bunu önlemek için; Domain-Driven Design (DDD), ortak kütüphaneler veya performans ve maliyet analizi gibi yöntemler kullanılabilir.



X

Tight Coupling

Sıkı Bağlantı

- ✓ Farklı servisler arasında yüksek derecede bağımlılık olması ve buna bağlı ölçeklenebilirliği sınırlamasıdır.

-
- ✓ Bunu önlemek için; Event-Driven Architecture, Domain-Driven Design (DDD), asenkron iletişim veya orkestrasyon araçları gibi yöntemler kullanılabilir.



X

Ignoring the Human Cost

İnsan Maliyetini Göz Ardı Etme

- ✓ Teknoloji yoğun ortamlarda veya mikro servis mimarileri gibi karmaşık sistemlerde çalışan kişilerin sistemleri yönetemeyeceği kadar iş yoğunluğu verilmesidir.
- ✓ Bunu önlemek için; eğitim- gelişim olanakları, lider/yöneticilerin sosyal eğitimleri veya çalışan deneyimini öncelikli kılma gibi yöntemler kullanılabilir.



HUBX

Sunum Bitti 🙌

Dinlediğiniz için
Teşekkürler!

Atakan Şengöz



HUBX

Any Questions?