Mikro-Frontend Karar & Mimari Planlama Şablonu — Yazılım / Micro-Frontend (v1.0)

Asset Amaç: Bu şablon, mikro-frontend mimari geçiş kararını bir teknoloji trendi veya popülerlik unsuru yerine tamamen ölçülebilir teknik ve operasyonel kriterlere (takım organizasyon yapısı, bağımsız modül sayısı, otonom canlıya alım ihtiyacı, SSR/SEO riskleri ve performans bütçesi disiplini) dayandırarak vermenizi sağlar. Sistemdeki modül/ekip sahipliklerini netleştirerek risksiz bir Kavram Kanıtlama (POC) kapsamı çıkarır ve sürdürülebilir yayına alım (release) ile QA kapılarını planlar. Yanlış mimari kurulumlarda ortaya çıkan yüksek debugging sürelerini ve performans maliyetlerini engellemek adına "platform standartları" ve kalite bariyerlerini zorunlu tutar.

Kim Kullanır?: Teknik Liderler (Tech Lead), Platform Mühendisleri (Platform Engineering), Ürün Sahipleri (Product Owner), SEO ve Performans Uzmanları ile DevOps/QA Mühendisleri.

Nasıl Kullanılır?

1. Dijital platformunuzdaki tüm bağımsız modülleri ve bu modüllerden sorumlu olan ekipleri haritalandırarak, aralarındaki teknik bağımlılıkları ve kesişim noktalarını işaretleyin.
2. Şablondaki 5 kritik karar kriterini puanlayarak projenizin mimari ihtiyaç seviyesini ("Monolit", "Modüler Monolit" veya "Mikro-Frontend") matematiksel olarak belirleyin.
3. Paylaşılan yol haritası doğrultusunda 14 günlük bir POC sprint planı uygulayın; elde ettiğiniz ön sayfa performansı ve SEO QA test sonuçlarına göre mimariyi güvenle ölçekleyin.

TEMPLATE — Mikro-Frontend Karar & Mimari Planlama Şablonu

A) Boş Şablon Alanları (Mimari Matris)

1) Ürün / Portal Modül Haritası

Sistemdeki işlevsel alanların sınırlarının çizilmesi ve mimari röntgeninin çıkarılması:

- **Platform Modülleri (Örn: Rezervasyon, Dashboard, Ödeme, Raporlama):**
  _____
- **Modüller Arası Teknik Bağımlılıklar (Veri akışları, ortak state kullanımı):**
  _____
- **Kritik Kullanıcı Akışları (Login, sepet adımları, rapor üretimi vb.):**
  _____

2) Yazılım Takım Yapısı ve Yönetişim (Governance)

Organizasyon şeması ile yazılım mimarisinin uyumluluğunun (Conway Kanunu) denetlenmesi:

- **Platform Üzerinde Çalışan Bağımsız Ekip Sayısı:** `____`
- **Ekiplerin Modül Sahiplikleri (Hangi modülden hangi alt takım sorumlu?):** `_____`
- **Takımların Bağımsız Deploy (Canlıya Çıkış) İhtiyacı Seviyesi:** `[ ] Düşük / [ ] Orta / [ ] Yüksek`

3) Stratejik Karar Puanlama Matrisi (1–5 Puan)

Mimari karmaşıklık ihtiyacını belirlemek için her kriteri 1 (En Düşük) ile 5 (En Yüksek) arasında puanlayın:

- `[ ]` **Modül Sayısı ve İş Bağımsızlığı:** Modüllerin iş mantığı olarak birbirinden ayrışma ve izole olma derecesi.
- `[ ]` **Deploy Sıklığı İhtiyacı:** Ekiplerin birbirini beklemeden gün içinde bağımsız canlıya çıkış yapma ihtiyacı.
- `[ ]` **Platform & Observability Olgunluğu:** Ekibin izleme (monitoring), CI/CD otomasyonu ve altyapı yönetim gücü.
- `[ ]` **SEO & SSR Gereksinimi:** Sayfaların arama motoru botları tarafından taranma ve sunucu taraflı işlenme (SSR) zorunluluğu.
- `[ ]` **Performans Budget Disiplini:** Ekibin tarayıcı bundle boyutları ve Core Web Vitals sınırlarına sadık kalma yetisi.

Toplam Skor: `___ / 25`

- **Karar Eşiği: 1-10 Puan:** Monolit | **11-18 Puan:** Modüler Monolit | **19-25 Puan:** Mikro-Frontend

4) Mimari Tercih ve Platform Altyapı Katmanı

Seçilen mimarinin teknik birleştirme (composition) ve ortak kütüphane kurgusu:

- **Composition Yöntemi:** `[ ] Route-Level (Nginx/Gateway)` / `[ ] Module Federation (Webpack/Vite)` / `[ ] Build-Time`
- **Paylaşılan Ortak Kütüphaneler (Shared Libs - Örn: React, Tailwind, Utils):** `_____`
- **Merkezi Kimlik Doğrulama ve UI Platform Katmanı (Auth & UI Kit):** `_____`

5) SEO & Sunucu Taraflı İşleme (SSR) Notları

Çoklu uygulama yapılarında arama motoru görünürlüğünün riske atılmaması için planlama:

- **İndekslenebilir / Organik Trafik Alan Kritik Sayfa Tipleri:** `_____`
- **Mikro-Frontend SSR Stratejisi (Örn: Server-Side Composition, Edge SSR):** `_____`

- **Head, Meta ve Yapılandırılmış Veri (Schema) Yönetim Standardı:**

  _____

6) Kalite Bariyerleri (QA Gate) & Yayına Alım Standartları

Her bir mikro uygulamanın ana platforma dahil edilmeden önce geçmek zorunda olduğu kapılar:

- **Performans Testi (Perf Gate):** Modül bazlı bundle bütçesi sınırları ve otomatik Core Web Vitals (CWV) denetimi.
- **SEO QA Denetimi:** Sayfa kaynak kodunda canonical, robots.txt kuralları ve dinamik sitemap eşleşme kontrolü.
- **Güvenlik Kapısı (Security Gate):** Rol bazlı erişim kontrolü (RBAC) testleri ve kaynak kodda açık "secret/token" taraması.
- **Geri Alma Planı (Rollback Kurgusu):** Hatalı deploy anında ana platformu etkilemeden ilgili modülü saniyeler içinde eski sürüme çekme senaryosu.

B) Nasıl Doldurulur? (5 Kritik Kural)

1. **Teknolojiyi Değil, Ekip ve Modül Yapısını Önceliklendirin:** Mikro-frontend kararı teknik bir hevesle verilmez. Eğer platformda çalışan bağımsız ekip sayınız azsa ve modülleriniz sıkı sıkıya birbirine bağlıysa mikro-frontend mimarisinden kaçının; önce organizasyonel sınırları netleştirin.
2. **Shared Katmanını Netleştirmeden Kod Yazmaya Başlamayın:** Merkezi kimlik doğrulama (auth) akışları, ortak state yönetim kuralları ve kurumsal tasarım sistemi (UI Kit/Design Tokens) standartları koda dökülmeden mikro-frontend mimarisine geçiş yapmayın.
3. **Büyük Bir Geçişe Girişmeden Önce Mutlaka POC Yapın:** Tüm platformu aynı anda dönüştürmeye çalışmayın. Riski en düşük, bağımsızlığı en yüksek tek bir yan modül seçerek (örn: raporlama veya admin paneli) 14 günlük bir deneysel çalışma (POC) ile sınırları test edin.
4. **Her Modül İçin Katı Performans Bütçeleri (Budget) Koyun:** Mikro-frontend yapılarında her ekip kendi modülüne bağımsız kütüphaneler ekleyerek toplam sayfa boyutunu kolayca şişirebilir. Bunun önüne geçmek için her modüle katı bundle boyutu sınırları (örn: max 50kb) atayın.
5. **SEO ve SSR Etkisini İlk Günden Mimariye Dahil Edin:** Eğer kullanıcıya açık, indekslenen bir web sitesi veya portal yönetiyorsanız, mikro uygulamaların tarayıcıda birleştirilme (client-side composition) aşamasındaki SEO risklerini baştan hesaplayın. Sunucu taraflı çözümleri sonradan eklemek çok maliyetlidir.

C) Doldurulmuş 1 Pratik Örnek (Kısa Senaryo)

- **Platform Modül Kapsamı:** Sistem; kullanıcı ana ekranı (dashboard), ağır veri sorgularının döndüğü "raporlama" modülü ve içerik yönetiminin yapıldığı "admin" paneli olmak üzere 3 ana parçaya ayrılmıştır.
- **Mimari Karar ve Strateji:** Yapılan puanlama sonucunda sistem "Modüler Monolit" yapısında tutulmuş; ancak ekiplerin bağımsız çalışabilmesi adına sadece "raporlama" modülü ayrıştırılarak Webpack Module Federation ile ilk mikro-frontend POC çalışması başlatılmıştır.

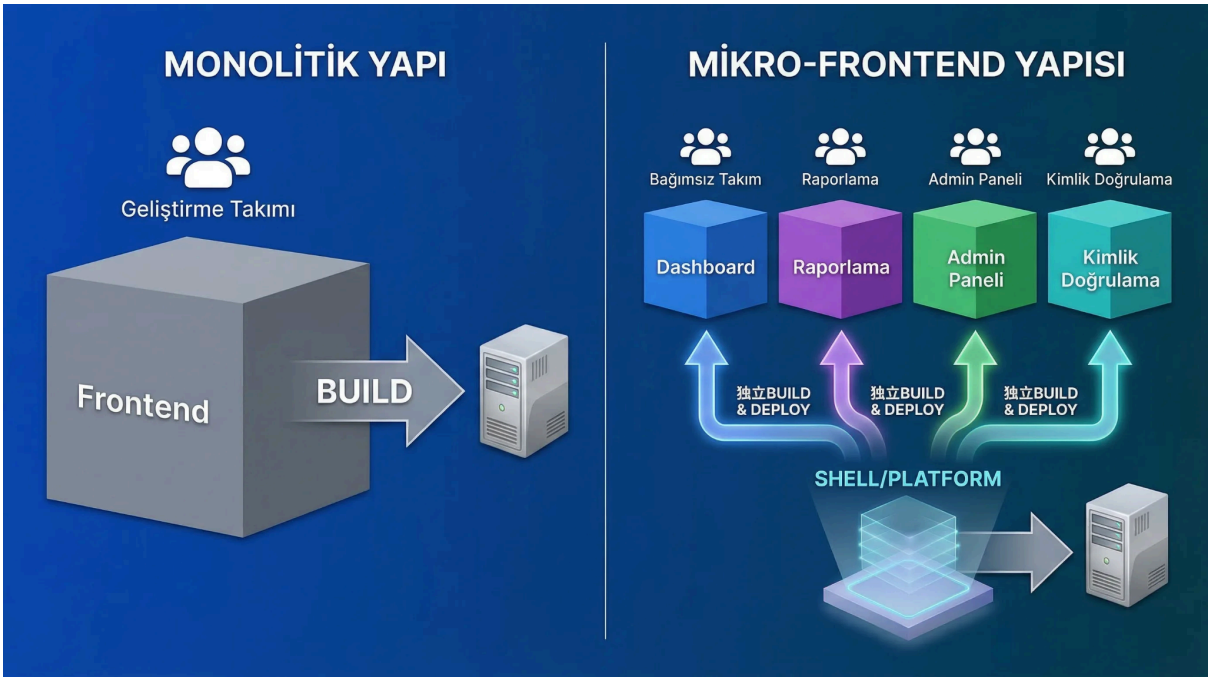
- **Uygulanan Kalite Kapıları (QA Gate):** Derleme (build) hattına her modül için maksimum 75kb bundle limit kontrolü eklenmiş, tarama botları için otomatik SEO smoke test kurgusu kurulmuş ve modüller arası yetkilendirmeyi doğrulamak adına RBAC kuralları kilitlenmiştir.

D) Teknik Kontrol Listesi (Checklist)

- [ ] **Haritalama Tamamlandı:** Platformdaki tüm modüller, veri bağımlılıkları ve bunlardan sorumlu olan yazılım ekipleri listelendi.
- [ ] **Matematiksel Puanlama Yapıldı:** 5 temel kriter üzerinden karar puanlaması yapılarak projenin gerçek mimari ihtiyaç seviyesi tescillendi.
- [ ] **POC Sınırları Çizildi:** Dönüşüm riskini sıfırlamak amacıyla ilk etapta dönüştürülecek MVP / deneme modülü seçildi.
- [ ] **Kalite Kapıları (QA Gate) Kuruldu:** Dağıtım hattına (pipeline) entegre performans bütçesi, SEO denetimi ve güvenlik tarama kuralları tanımlandı.
- [ ] **Acil Durum Senaryosu Hazır:** Olası canlı sistem hatalarında ana platformun çalışmaya devam edebilmesi için modül bazlı rollback planı yapıldı.

Deliverables (Teslim Edilecek Çıktılar)

- Ekiplerin Sınırlarını Belirleyen 1 Adet Detaylı Modül ve Ekip Sahipliği (RACI) Tablosu
- Altyapı Dönüşümünün Gerekçelerini Sunan 1 Adet Resmi Mimari Karar Dokümanı (ADR)
- İlk Deneme Sonuçlarını Metriklerle Açıklayan 1 Adet Öncesi / Sonrası POC Performans Raporu
- Canlıya Alım Güvenliğini Sağlayan 1 Adet Entegre QA Gate ve Kalite Kontrol Listesi



[Architecture Micro-Frontend Diagram]

Modern bir mikro-frontend mimarisinin birleştirme ve dağıtım katmanlarını gösteren, 16:9 formatında tasarlanmış teknik bir mimari akış şeması. Gece siyahı arka plan

üzerinde, en üstte bir şemsiye görevi gören "App Shell / Container Katmanı" parlıyor. Bu şemsiyenin altında, birbirine sadece ince neon hatlarla bağlı, tamamen izole kutular halinde duran 3 farklı mikro uygulama modülü (Dashboard Micro-App, Analytics Micro-App, Auth Katmanı) yer alıyor. Modüllerin bağımsız CI/CD hatlarından geçerek tarayıcıda Webpack Module Federation ile nasıl tek bir pürüzsüz arayüze dönüştüğü, net yön okları ve scannable teknik etiketlerle görselleştiriliyor.

[Team Ownership Grid Mockup] — 16:9 — ekip-modul-sahipligi-tablo-tasarimi.webp *Yazılım direktörleri ve proje yöneticileri için tasarlanmış, 16:9 formatında minimalist bir ekip sahipliği ve QA yönetim matrisi mockup görseli. Mat antrasit zemin üzerinde, net satır ve sütun çizgileriyle ayrılmış bir tablo yapısı bulunuyor. Tablonun dikey ekseninde platform modülleri (Ödeme, Katalog, Üye Paneli), yatay ekseninde ise bu modüllerden sorumlu olan bağımsız ekipler (Team Alfa, Team Beta) ve her ekibin kendine ait bundle boyutu bütçesi (Budget: 60kb) ile aktif QA kapısı durumları (Neon yeşil onay ikonlu Perf Gate, SEO Gate, Security Gate) "clarity at a glance" felsefesiyle sunuluyor.*