TÜBİTAK BİLGEM YTE | YAZILIM TEKNOLOJİLERİ ARAŞTIRMA ENSTİTÜSÜ

MİKRO ÖNYÜZ MİMARİSİ

SAYI: 03

ARAŞTIRMA SERİSİ

Simge ve Kısaltmalar

Kısaltmalar	Açıklama
SPA	Single Page Application (Tek Sayfa Uygulama)
API	Application Programing Interface (Uygulama Programlama Arayüzü)
SEO	Search Engine Optimization (Arama Motoru Optimizasyonu)
SSR	Server Side Rendering (Sunucu Tarafı İşleme)
CDN	Content Delivery Network (İçerik Dağıtım Ağı)
TÜBİTAK	Türkiye Bilimsel ve Teknolojik Araştırma Kurumu
BİLGEM	Bilişim ve Bilgi Güvenliği İleri Teknolojiler Araştırma Merkezi
YTE	Yazılım Teknolojileri Araştırma Enstitüsü

Yazar
Sacit GÖNEN

Yayın Koordinatörü
Elif ŞENYİĞİT

Editörler
Ahmed Enis ERKAYA
Sevinç KARAKAŞ
Tuğçe YILMAZ

Tasarım
Şeyma KOÇER

İletişim: 0 (312) 289 92 22 - yte.bilgi@tubitak.gov.tr

yte.bilgem.tubitak.gov.tr

İçindekiler

Önsöz

TÜBİTAK BİLGEM Yazılım Teknolojileri Araştırma Enstitüsü, 2012 yılından bu yana yazılım teknolojilerinde AR-GE faaliyetleri yürüten bir araştırma kuruluşudur. Araştırma faaliyetlerinde elde ettiği birikimini stratejik, hassas ve kritik projeler yürüterek kamu adına hayata geçirmekte; kurumlarımıza dijital dönüşüm, yazılım geliştirme teknolojileri ve kalite süreçleri konusunda danışmanlık vermektedir.

Araştırma Serisi ile TÜBİTAK BİLGEM YTE kurum içi çalışmaların yaygınlaştırılması ve sektörün erişimine açılması amaçlanmaktadır. Araştırma Serisi'nde yayınlanan çalışmalar TÜBİTAK BİLGEM YTE çalışanlarının projelerde elde ettiği bilgi birikimini paylaşmak adına derlenmiştir. Bu çalışmalar ile ülkemizin yazılım sektörüne katkı sağlanması hedeflenmektedir.

Giriş

Dünyanın giderek dijitalleştiği, her şeyin sanal ortama aktarıldığı bir dönemde yaşıyoruz. Eskiden sadece bilgisayarlara programlar yükleyerek yapılabilen birçok işlem, artık farklı platformlar üzerinden onlarca değişik cihaz yardımı ile kolayca yapılır hale geldi. Teknolojideki bu baş döndürücü ilerleyiş, şirketlerin farklı yeni fikirler ve yeni uygulamalar ortaya çıkarmasına olanak sağladı. Önceden sadece ürün satışına yönelik e-ticaret siteleri varken, günümüzde çoklu ürün ve hizmet yelpazesine sahip e-ticaret siteleri ortaya çıktı. Elbette bütün bunların gelişimi bir anda olmadı. Son kullanıcıların zaman içerisinde değişen arzuları farklı ihtiyaçları, farklı ihtiyaçlar yeni teknolojileri, yeni teknolojiler de bu ihtiyaçları karşılayan yazılım projelerinin hayata geçirilmesine zemin hazırladı. Üretken bir döngü içerisinde dijital dönüşümlerin daha kolay yapılabilmesi için yeni teknoloji ve mimariler sürekli gelişti ve gelişiyor. Günümüzde oldukça popüler hale gelen mikroservis ve mikro önyüz (micro frontend) mimarileri de bu döngü sonucunda ortaya çıkmışlardır.

Uygulamalar, fiziki ortamdaki bir işin daha düşük maliyetler ile sanal ortamda gerçekleşmesini sağlarlar. Her üründe olduğu gibi uygulamalar da son kullanıcılarını memnun ettiği, yasal zorunluluklarını yerine getirdiği ve hatasız çalıştıkları sürece başarılıdırlar. Son kullanıcıların istekleri, uygulamanın sunduğu hizmetler ve uygulamanın yasal zorunlulukları zaman içerisinde değişime uğrarken, uygulamalarda hataların oluşması da olağan bir durumdur. Bu değişimlere ve hatalara gerekli güncellemeler ile hızlı bir şekilde cevap verilmesi oldukça önemlidir. Büyük ve karmaşık uygulamalarda güncellemelerin hızlı ve direkt bir şekilde yapılabilmesi için uygulamanın bağımsız küçük parçalardan oluşuyor olması ciddi öneme sahiptir. Mikroservis ve mikro önyüz mimarileri büyük ve karmaşık yazılım projelerinin küçük parçalara bölünerek kolayca güncellenmesini sağlamaktadırlar. Bu durumu, Cloves CARNEIRO ve Tim SCHMELMER yazdıkları “Microservices from Day One” adlı kitapta şu şekilde açıklamaktadırlar: “Her başarılı uygulama değişim geçirmek zorundadır. Önemli olan, maliyeti düşük tutarak uygulamanın ihtiyaç duyduğu değişimleri güvenli bir biçimde sürekli yapabilmektir. Bunu yapmanın en kolay yolu uygulamayı mikroservis ve mikro önyüz mimarileri kullanarak geliştirmektir.” Bu çalışmada büyük ve karmaşık projelerin daha düşük maliyetler ile daha sürdürülebilir bir biçimde oluşturulmasına imkân sağlayan mikro önyüz mimarisi incelenmiştir.

Mikro Önyüz Mimarisi

Monolit mimariye sahip uygulamalarda tüm işlemler tek bir yerden, tek bir kod üzerinden gerçekleşir. Uygulama; tek bir parça olarak geliştirilir, yayınlanır ve ölçeklendirilir. Birden çok geliştirme ekibinin çalıştığı projelerde ekiplerin birbirlerinin çalışma alanlarını etkilememeleri için ekipler arası sürekli koordinasyon vardır. Bütün uygulama tek bir parça olduğu için uygulamanın farklı bölümlerinde farklı teknolojiler kullanılamaz, tüm uygulama tek bir teknoloji üzerine geliştirilir. Uygulamada yapılan her değişiklikte tüm uygulama baştan sona test edilir ve yeni bir paket oluşturulur. Tüm bunlar küçük ve karmaşıklığın az olduğu projelerde yönetilebilir problemlerdir ancak uygulamalar büyüdükçe ve karmaşıklık seviyeleri arttıkça bu problemler ile baş etmek zorlaşır.

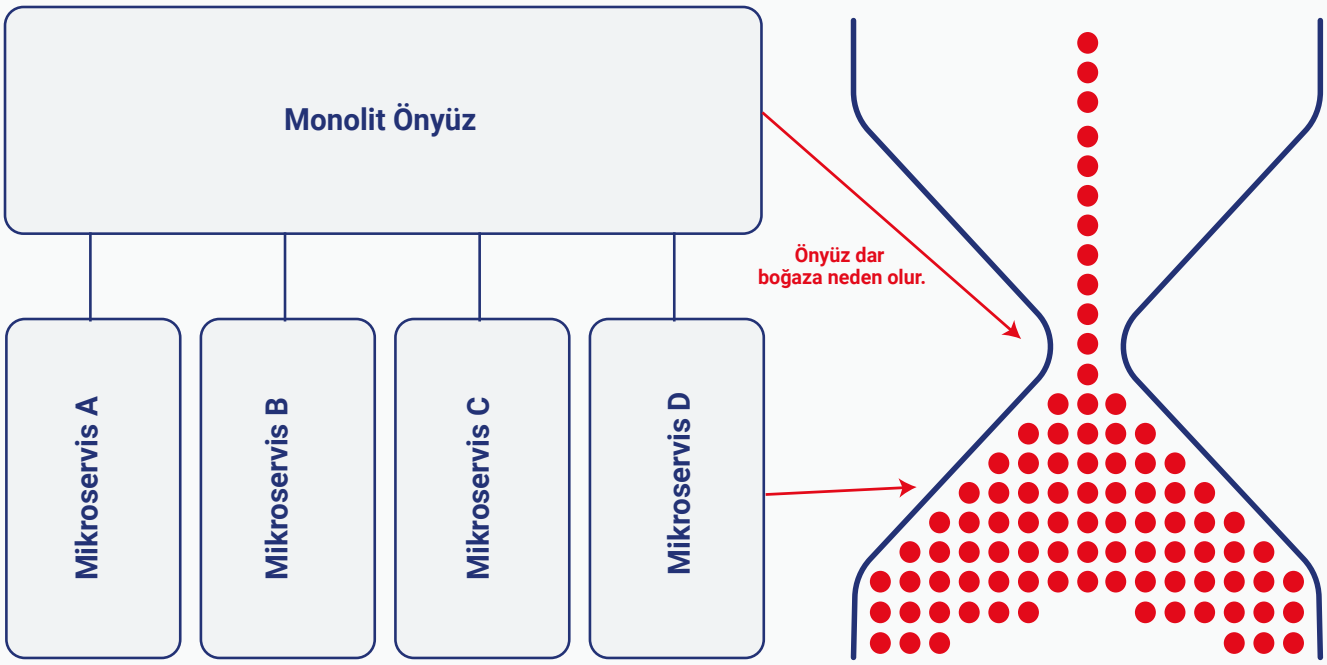
Mikroservis mimarisinde, uygulama her biri farklı bir işten sorumlu ve birbirinden bağımsız, anlamlı küçük servislere ayrılır. Bu ayrıştırma sayesinde servisler modüler hale gelirken;

- Hızlı ve bağımsız olarak kolayca geliştirilebilir.
- Hızlıca canlı ortama sürüm çıkabilir.
- Etkin bir şekilde ölçeklenebilir.
- Hatalarını kendi içinde daha iyi izole edebilir.
- Rahatça farklı teknolojileri kullanabilir.
- Diğer servislerdeki farklı teknolojiler ile kolayca entegre olabilir.
- Değişen iş ihtiyaçlarına kolayca adapte olabilir.

Bu sebeplerden dolayı büyük ve karmaşık uygulamalarda mikroservis mimarisinin kullanımı oldukça yaygındır.

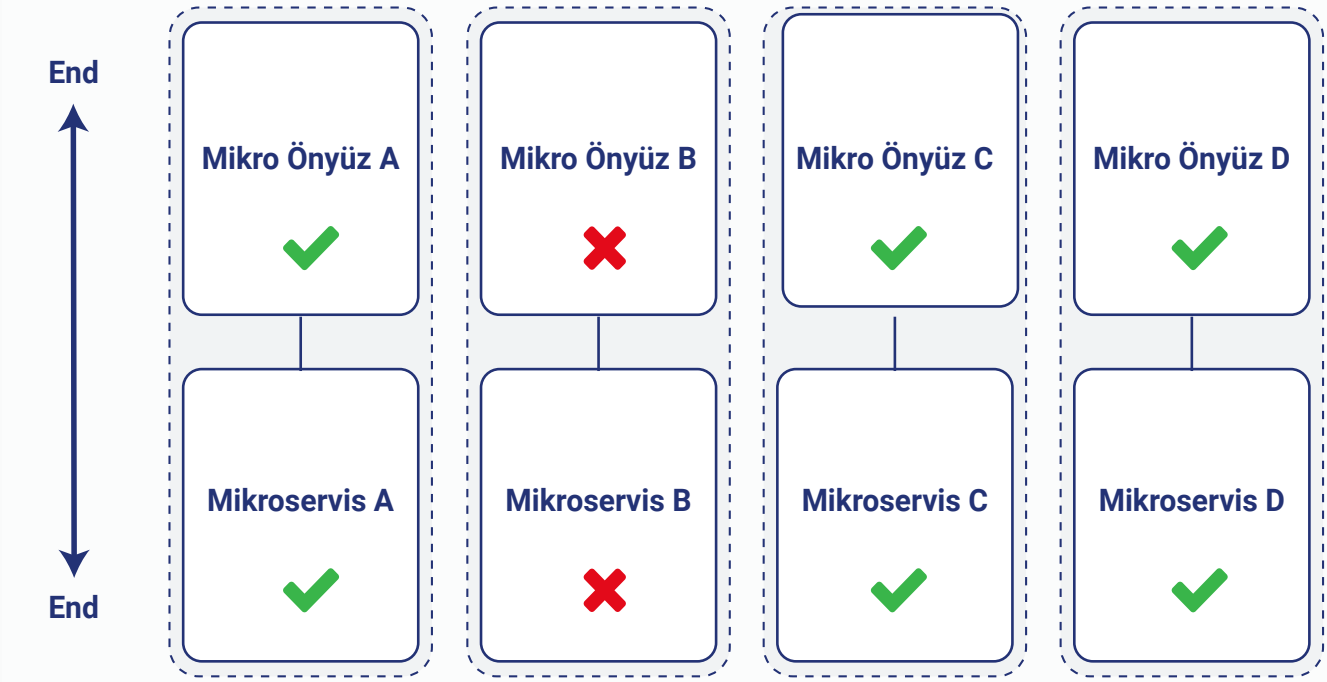
Mikroservis mimarisi birçok problemi çözüyor olsada elbette bazı dezavantajları da vardır. Dağıtık yapısından dolayı mikroservis mimarisi, dağıtık sistemlerin getirdiği dezavantajlara sahiptir ve bunlara ek olarak servislerin otomatik yönetimi ve izlenmesi için ek araçlara ihtiyaç duyulurken servisler arası iletişimlerde de gecikmeler olabilir. Mikroservis mimarisi uygulamaların görünmeyen bölümleri için kullanılan bir mimari iken mikroservis mimarisini tamamlayan ve uygulamaların görünen kısımlarında kullanılan mimari **“Mikro Önyüz Mimarisi (Micro Frontend Architecture)”**dir. Mikro önyüz mimarisi, mikroservis mimarisinin avantaj ve dezavantajlarına sahiptir. İyi bir kullanıcı deneyimi için iyi bir mikro önyüz mimarisinde;

- Uygulama hızlı açılmalıdır.
- Mikro önyüzler arasında bileşenler; düğme (button), girdi alanları (input fields) vb. tutarlı olmalıdır.
- Mikro önyüzler arası geçişler akıcı olmalıdır.
- Hızlı açılma ve akıcı geçiş için mikro önyüzlerin paket boyutları küçük olmalıdır.
- Mikro önyüzler bağımsız bir şekilde yayınlanabilmelidir.
- Mikro önyüzler teknolojik açıdan özgür olmalıdır.



Şekil 1. Mikroservisler & Monolit Önyüz

Mikro önyüz mimarisi, mikroservis mimarisini tamamlayan bir mimaridir. Mikroservis mimarisinin sahip olduğu çeviklik ve esneklikten dolayı hızlı uygulama geliştirmek mümkünken önyüz tarafında monolit bir mimarinin kullanılması, Şekil 1'de olduğu gibi, geliştirmeler sırasında dar boğazın oluşmasına, geliştirmelerin sekteye uğramasına neden olur. Mikro önyüz mimarisi; geliştirmeler sırasında dar boğazların oluşmaması ve ilgili mikroserviste geliştirme yapan ekibin başka bir yere bağımlı olmadan gerekli güncellemeleri önyüzde de yapıp yeni özelliği canlı ortama olabildiğince hızlı kurabilmesi, başka bir deyişle ekibin çok yönlü (cross-functional) olarak baştan sona gerekli güncellemeleri gerçekleştirebilmesi için ortaya çıkmıştır.



Şekil 2. Mikroservis ve Mikro Önyüz Mimarisi ile Uçtan Uca Geliştirme

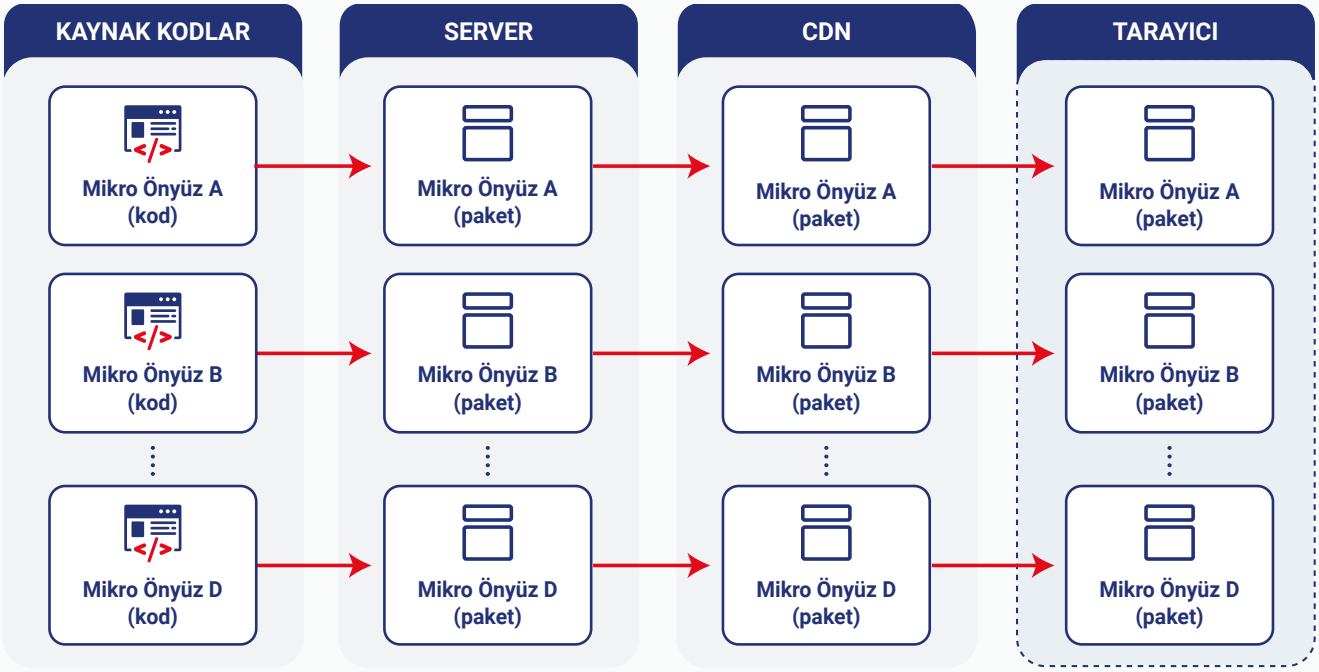
Mikro önyüz mimarisi ve mikroservis mimarisi birlikte kullanıldığı zaman oldukça etkin mimarilerdir. Mikroservis mimarisindeki mikroservisler ve mikro önyüz mimarisindeki mikro önyüzler Şekil 2'de olduğu üzere bire bir eşleştirildikleri zaman ideal uçtan uca geliştirme (end-to-end development) mümkün olur. Mikroservislerden veya mikro önyüzlerden birisi güncellendiğinde veya hataya düştüğü zaman sadece ilgili mikroservis veya mikro önyüz etkilenir. Bu sayede birbirlerinin varlıklarından etkilenmemesi gereken mikroservisler ve mikro önyüzler uçtan uca ilgili iş alanının bir problemini çözerler. Mikro önyüzlerin kendi ilgi alanları dışında bir noktaya yönelmesi mikro önyüz mimarisine ters olan birçok durumun, dolayısı ile problemin ortaya çıkmasına neden olur. Mikro önyüz mimarisi Luca MEZZALIRA'nın da dediği gibi birden fazla takımın çalıştığı ve geliştirme sırasında iteratif yaklaşımın izlendiği projelerde mantıklı bir mimaridir. Geliştirilecek olan uygulama için mikro önyüz mimarisinin uygunluğuna karar verildikten sonra mimarinin doğru bir şekilde uygulanması da oldukça önemlidir.

Uygulamanın Derlenmesi

Mikro önyüz mimarisinde uygulamanın bir bütün olarak kullanıcılara nasıl sunulacağı son derece önemlidir. Uygulama bütünü, kod seviyesi ve kullanıcının cihazı dâhil olmak üzere iki uç arasındaki herhangi bir yerde bir bütün haline getirilerek kullanıcıya sunulur. Mikro önyüzler bir araya getirilirken temelde 4 farklı derleme (composition) yöntemi vardır ve her derleme yöntemi altında farklı entegrasyon seçenekleri mevcuttur. Zamanla gelişen yeni teknoloji ve yaklaşımların daha farklı derleme ve entegrasyon yaklaşımlarını ortaya çıkarabileceği göz ardı edilmemelidir.

1. Client Side Derleme

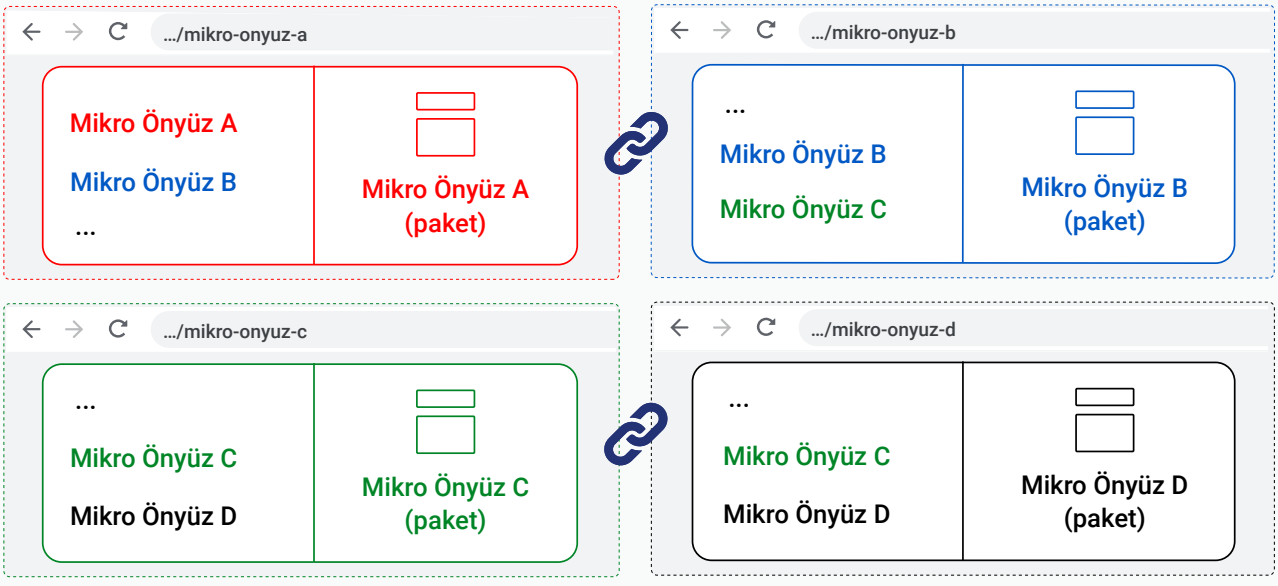
Client side derlemede mikro önyüzler kullanıcının cihazında bir araya getirilir. Bu derleme, mikro önyüz mimarisinin en kolay uygulandığı ve anlaşıldığı yöntemdir. Kullanıcının uygulamaya erişim için yaptığı ilk istek sonrası hangi mikro önyüzlerin ekran üzerinde nasıl ve nerede gösterileceğine karar verebilmek ve bu işlemler sırasında kullanıcı deneyimini düşürmeden anlamlı bir içerik sunabilmek için basit bir JavaScript ve HTML dönülür. Bu yöntemde ölçeklendirme (scalability) problemi yoktur. JavaScript dosyasının çalıştırılması ve sonrasında ilgili mikro önyüz paketlerinin dinamik yüklenmesinden dolayı kullanıcı deneyimi açısından uygulamanın yavaş açılması gibi üstesinden gelinebilecek riskler vardır. Mikro önyüzlerin farklı çerçeveler (frameworks) kullandığı durumda çerçevelerin getirdiği boyut probleminden dolayı uygulamanın var olan yavaş açılma riski artabilir. Dolayısı ile bu gibi durumlarda boyut optimizasyonu konusunda daha dikkatli olunmalıdır. Diğer bir taraftan uygulamayı oluşturan mikro önyüzler kullanıcının cihazında bir araya getirildiği için SEO konusunda da sorunlarla karşılaşılabilir. Client side derleme, SEO, SSR ve uygulamanın hızlı açılmasının proje açısından ciddi bir öneme sahip olmadığı durumlarda kullanılması önerilen derleme yöntemidir.



Şekil 3. Client Side Derleme Örneği

1.1. Hyperlink Entegrasyonu

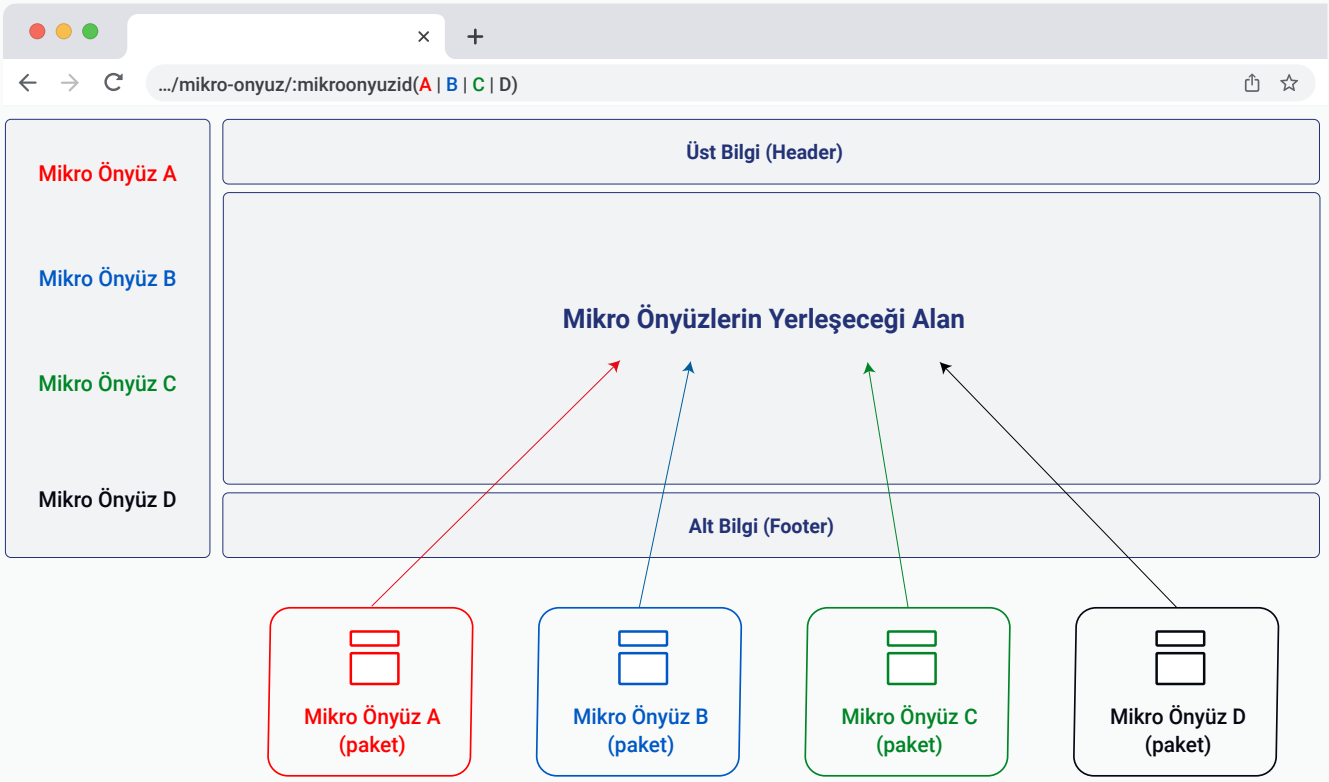
Client side derleme örneklerinden birisi olarak ele alabileceğimiz hyperlink entegrasyonunda mikro önyüzler tüm ekranı kapsayacak şekilde parçalara ayrılmıştır. Genel olarak bir ekranda tek bir mikro önyüz yer alırken farklı bir kaç mikro önyüzün de yer alması mümkündür. Bu yaklaşımın en güzel tarafı mikro önyüzlerin birbirlerinden tamamen bağımsız olmasıdır. Kötü tarafı ise üst ve alt bilgi (header ve footer) gibi ortak bölümlerin yönetim problemidir. Mikro önyüzler arasında arayüz bütünlüğü ve akıcı kullanıcı deneyimi sıkıntılı bir hal alırken ortak içeriklerin her mikro önyüzde ayrı ayrı yer alması gerekliliğinden dolayı mikro önyüzlerin paket boyutları (bundle size) da gereksiz yere büyük olur. Ancak bağımsız sürüm yönetimi, teknoloji özgürlüğü ve hızlı açılma konusunda en iyi entegrasyon yöntemlerinden birisidir.



Şekil 4. Hyperlink Entegrasyonu

1.2. App Shell Entegrasyonu

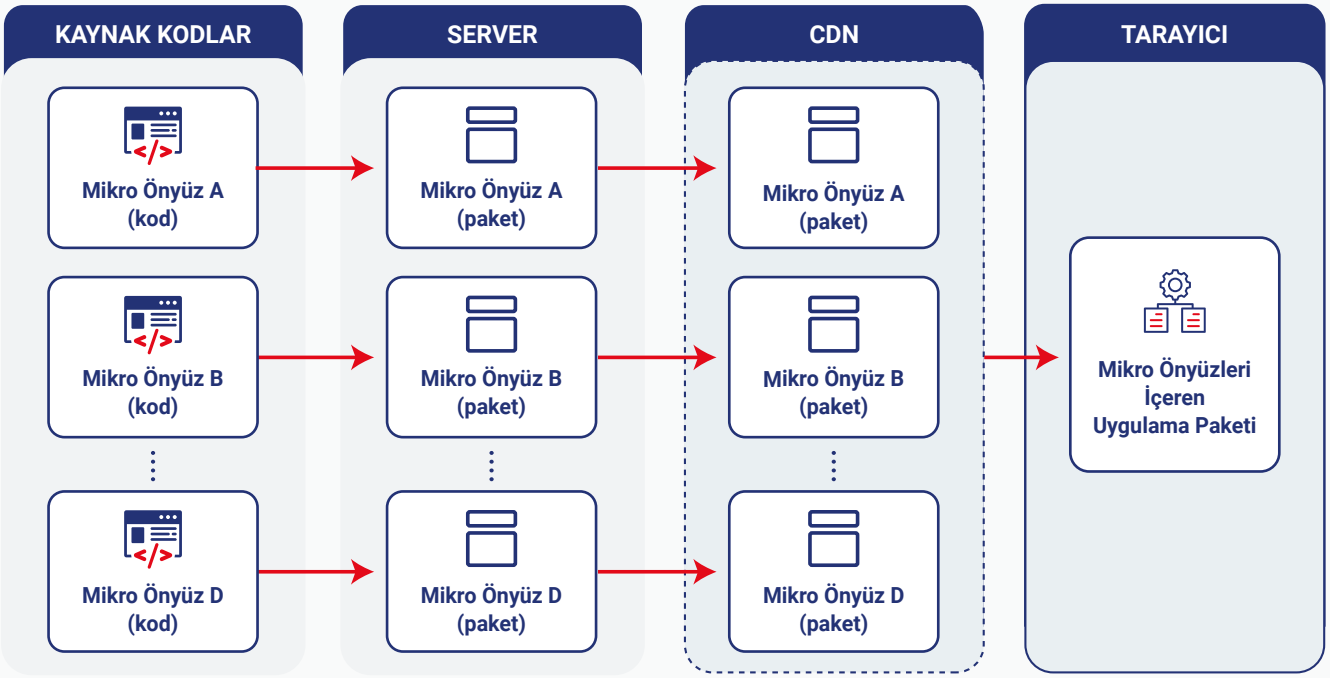
Client side derleme örneği olarak ele alabileceğimiz bir diğer entegrasyon yöntemi ise app shell entegrasyonudur. App shell entegrasyonu hyperlink entegrasyonunda karşılaştığımız bazı problemleri ortadan kaldırmaktadır. App shell entegrasyonu kullanılan uygulamalarda sadece belirli bir bölüm mikro önyüzlere ayrılmıştır ve dinamiktir. Hyperlink entegrasyonunda mikro önyüzler arası üst ve alt bilgi (header ve footer) gibi ortak olan bölümlerin yönetimi bu entegrasyonda (app shell) ana uygulamaya (host) bırakıldığı için ortak bölümlerin yönetimi daha kolaydır. Hyperlink'e çok benzeyen bu yöntem, ortak kodların sadece ana uygulamada yer almasından dolayı arayüz bütünlüğü ve paket boyutu konusunda daha iyi bir noktadadır. Bağımsız sürüm yönetimi ve hızlı açılma konusunda hyperlink ile aynı seviyede olsa da teknoloji özgürlüğü konusunda bazı kısıtlamalar ortaya çıkabilir.



Şekil 5. App Shell Entegrasyonu

2. Edge Side Derleme

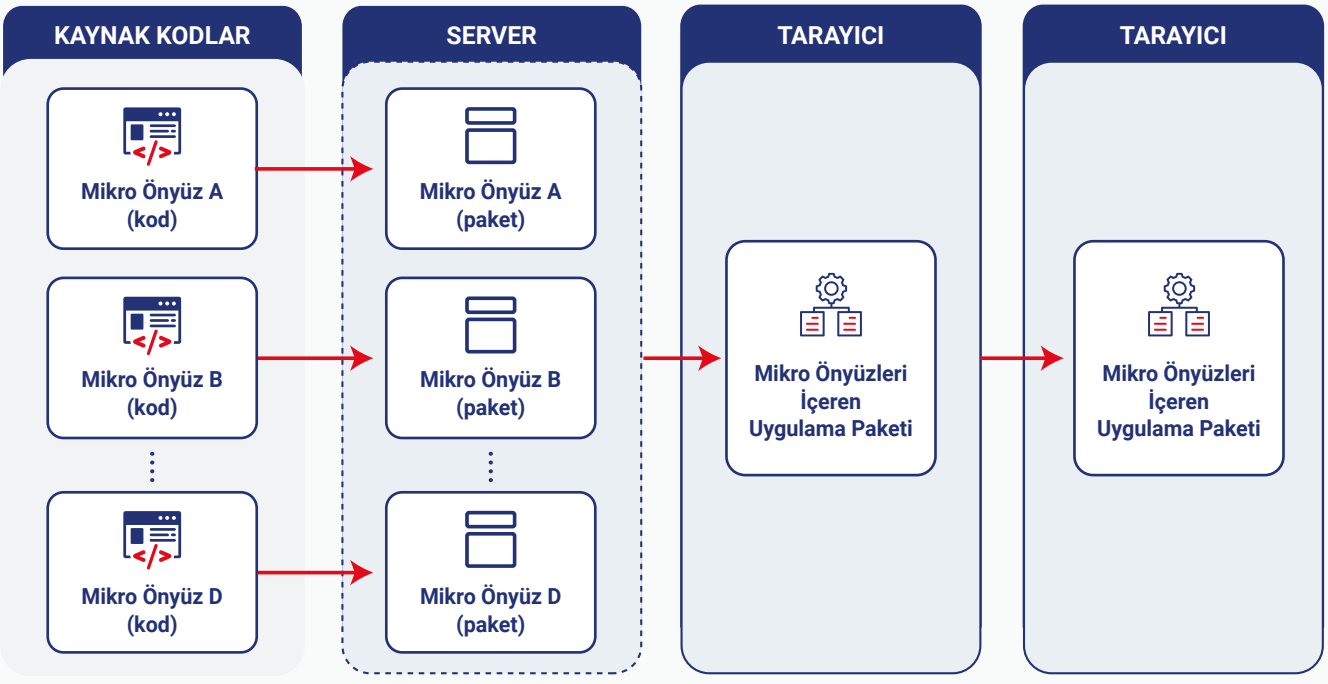
Edge side derlemede mikro önyüzler CDN üzerinde bir araya getirilir ve bunun için genellikle XML tabanlı Edge Side Includes (ESI) işaretleme dili kullanılır. Bu yöntemde en riskli nokta, farklı CDN'lerin ESI'yi farklı şekilde kullanıyor olması ve CDN sağlayıcıların koyduğu bazı işlem limitleridir. Bir CDN'den başka bir CDN'e geçiş sırasında çok fazla yeniden düzenleme (refactor) gerekebilir ve birden çok CDN aynı anda kullanılacak olursa bunların yönetimi oldukça zorlaşır.



Şekil 6. Edge Side Derleme Örneği

3. Server Side Derleme

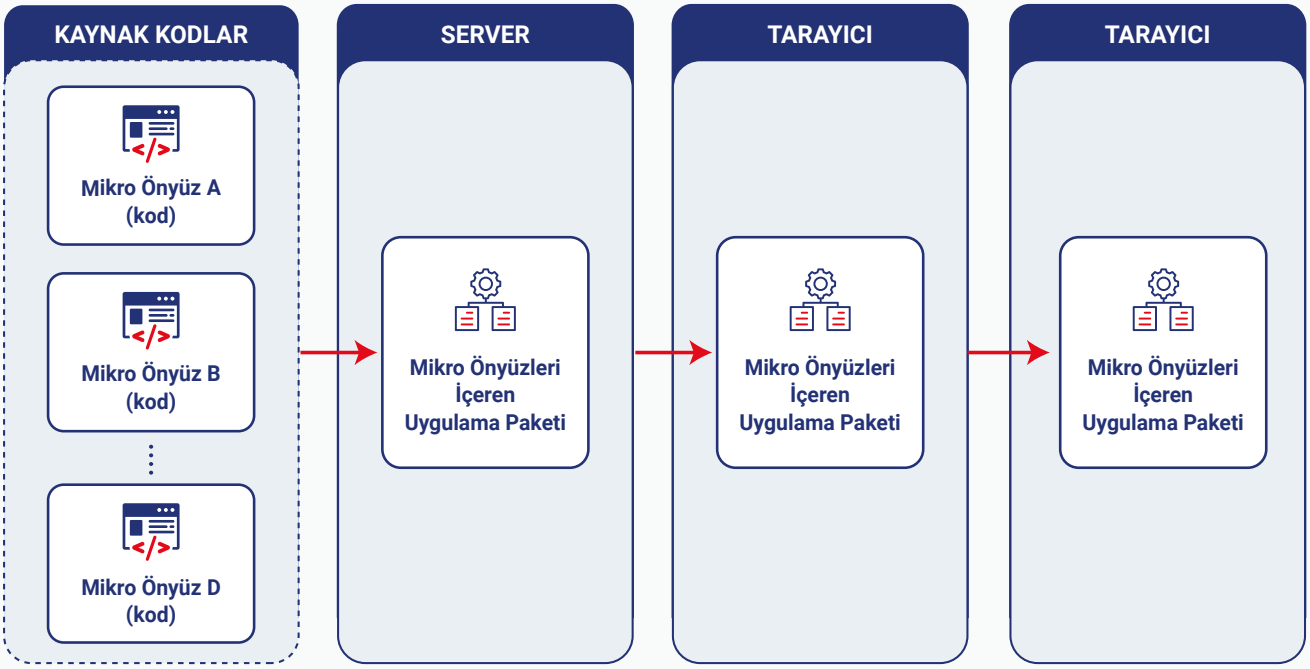
Server side derlemede mikro önyüzler sunucuda bir araya getirilir. Bu yöntemin en avantalı tarafı sunucu tarafında işleme (server side rendering) yapabilmesidir. Eğer uygulamanın kullanıcıya özgü içerikler sunması gerekiyor ise sunucu ciddi anlamda yük altında kalabilir ve ölçeklendirme (scalability) açısından riskler oluşur.



Şekil 7. Server Side Derleme Örneği

4. Build Time Derleme

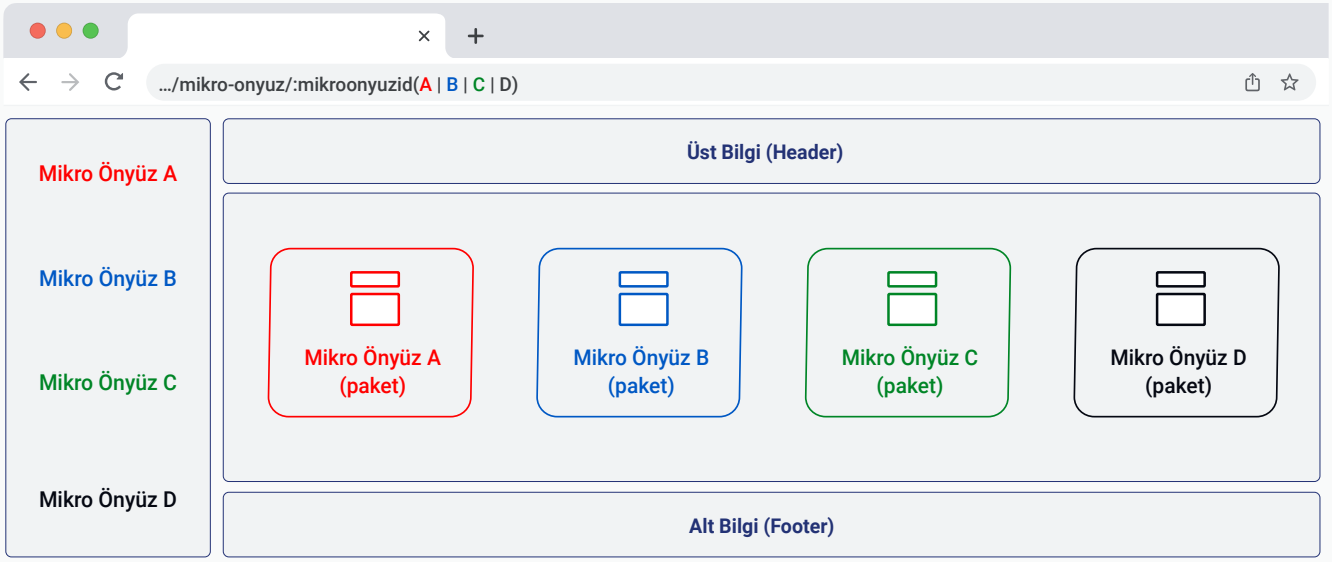
Build time derlemede paketlenmiş mikro önyüzler uygulama bütününü oluşturmak için farklı bir proje üzerinde paketlenme sırasında bir araya getirilir. Mikro önyüzlerin çerçeveler (frameworks) kullanılarak oluşturulduğu durumlarda, başta teknoloji özgürlüğü olmak üzere farklı kısıtlamalar ortaya çıkmaktadır. Bu kısıtlamaların üstesinden gelinebileceği gibi bu kısıtlamalar toplam uygulama boyutu açısından artı yönde de değerlendirilebilir. Ancak bu yöntem sonucunda tek bir uygulama paketi oluşacağı için ve bağımsız sürüm yönetimi diğer yöntemlerdeki kadar serbest olmadığından bu yöntem çok fazla benimsenmiş bir mikro önyüz mimarisi derleme yöntemi değildir.



Şekil 8. Build Time Derleme Örneği

4.1. Build Time Entegrasyonu

Build time derleme örneği olarak build time entegrasyonu incelenebilir. Monolit mimariye çok yakın olan bu entegrasyon yönteminde mikro önyüzlerin yapısında bazı limitlemeler oluşur. Örneğin mikro önyüzlerden birisi React.js ile yazılırken diğerinin Vue.js ile yazılması uygulamanın çalışma zamanı (runtime) performansını kötü yönde etkileyecektir ve bazı durumlarda teknoloji farklılığından dolayı derlenememesine bile neden olabilir. Ortak bileşen kütüphaneleri kullanılarak arayüz bütünlüğü, akıcı kullanıcı deneyimi ve tüm uygulamanın paket boyutu küçük tutulabilmektedir. Ancak, bağımsız sürüm yönetimi, teknoloji özgürlüğü ve hızlı açılma noktasında göz ardı edilemeyecek problemler vardır.



Şekil 9. Build-Time Entegrasyonu

Uygulamaların bir araya getirilmesi ile ilgili derleme yöntemleri altında örnek olarak ele aldığımız farklı entegrasyonların, iyi bir mikro önyüz mimarisinde olması gereken özellikleri karşılamadaki başarıları 10 üzerinden değerlendirilmiştir.

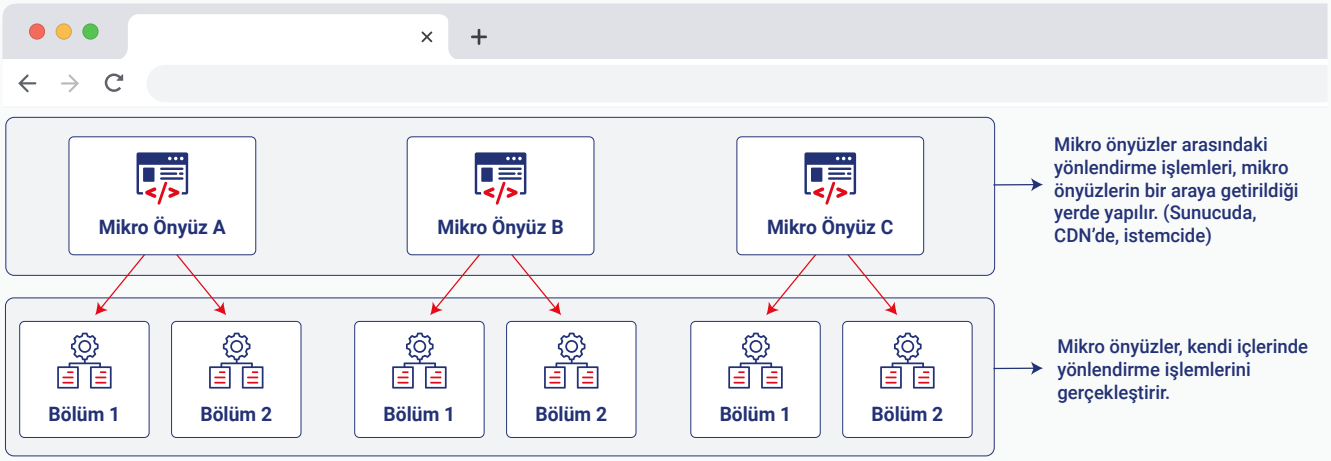
	Hyperlink Entegrasyonu	App Shell Entegrasyonu	Build Time Entegrasyonu
Uygulamanın Hızlı Açılması	7	7	2
Mikro Önyüzler Arası Arayüz Tutarlılığı	3	5	8
Mikro Önyüzler Arası Akıcı Geçiş	3	4	10
Mikro Önyüzlerde Paket Boyutunu Küçük Tutma	7	8	5
Mikro Önyüzlerin Bağımsız Yayınlanabilmesi	10	10	0
Mikro Önyüzlerin Teknoloji Özgürlüğü	10	9	5

Tablo 1: İdeal Mikro Önyüz Mimarisinin Özelliklerinin Entegrasyon Yöntemleri Açısından Değerlendirilmesi

Ekranlar Arası Yönlendirme

Mikro önyüz mimarisinde ekranlar arasında yönlendirme (routing) yapmak sanıldığı kadar zor bir işlem değildir. Yönlendirme yönetimi projeden projeye ciddi anlamda farklılık gösterebilir. Mikro önyüz mimarisi kullanılan projelerin bazılarında hiç yönlendirme yapılma ihtiyacı yok iken bazılarında hem ana uygulama hem de mikro önyüzler seviyesinde yönlendirme ihtiyacı söz konusu olabilir.

Mikro önyüzlerin basit parçalardan oluştuğu durumda yönlendirme işlemleri çoğunlukla sunucu üzerinde Server Side Includes (SSI) ile veya CDN üzerinde Edge Side Includes (ESI) ile gerçekleştirilir. Bunlara ek olarak istemci tarafında iFrame'de kullanılabilir ama getirdiği ek yükten dolayı tercih edilmez.

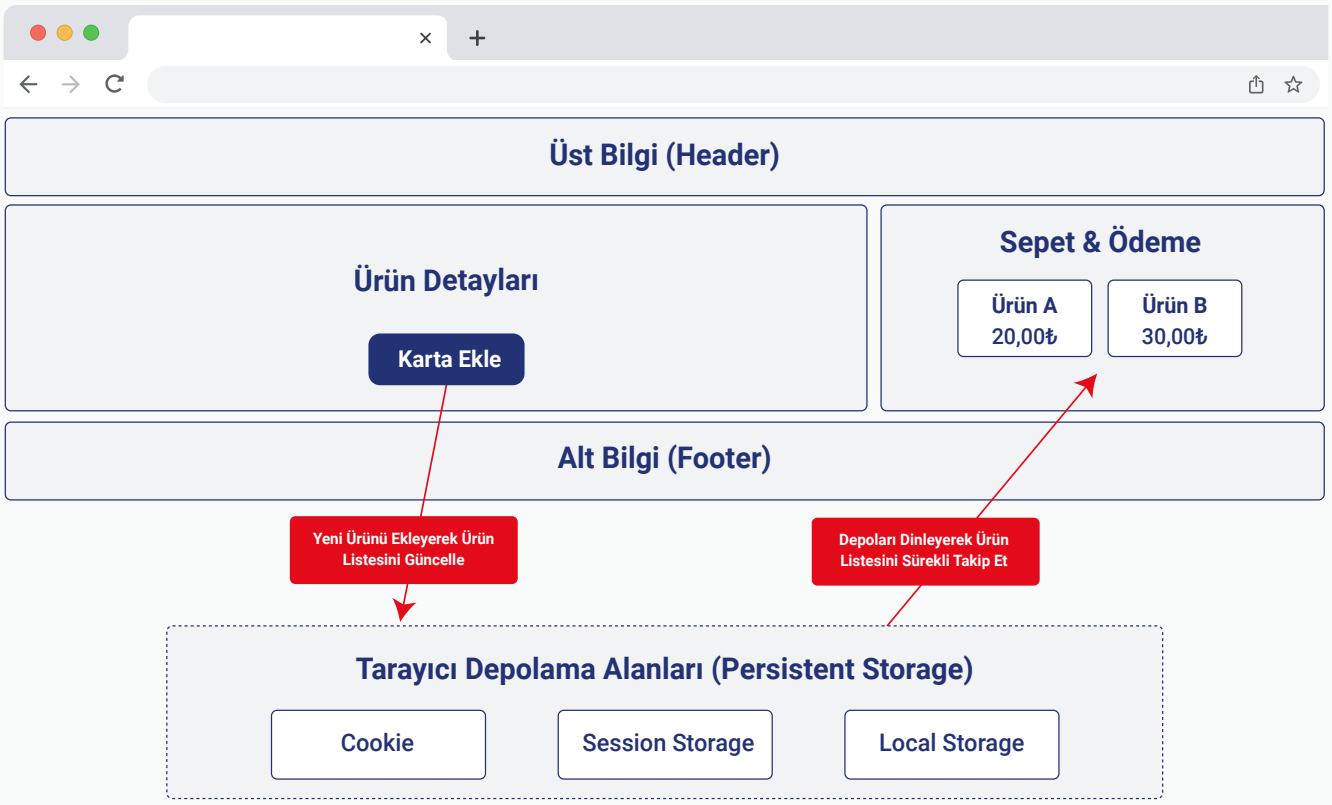


Şekil 10. Yönlendirme Yönetimi

Mikro önyüzlerin tek başlarına uygulama olduğu durumda, ana uygulama (host) üzerinde en üst seviyede yönlendirme işlemlerini yaparak ilgili mikro önyüzü ekrana yerleştirdikten sonra detaylı yönlendirmeler mikro önyüzlerin kendileri tarafından gerçekleştirilir.

Mikro Önyüzler Arasında Veri Yönetimi

Mikro önyüzler arası iletişim konusu da oldukça önemlidir. Optimum yaklaşımda mikro önyüzler birbirlerinden haberdar değildir ve olmamalıdır. Mikro önyüzlerin her biri farklı bir işten sorumlu ve birbirinden bağımsız olduğu için birbirleri ile bilgi paylaşmaya hiç ihtiyaç duymazlar veya çok az duyarlar. İstisnai durumlarda mikro önyüzler arası veri paylaşımı söz konusu olabilir. Bu durumda da paylaşılacak olan veriye göre farklı yaklaşımlar benimsenebilir. Örneğin mikro önyüzler arası veri paylaşımı için query-string kullanılabilir. Ancak bu yöntem ile büyük verileri veya güvenlik açısından önemli verileri (örneğin parolaları) paylaşmak çok uygun değildir. İhtiyaç duyulan zorunlu durumlarda büyük verilerin paylaşılması için üretici/tüketici (pub/sub) yaklaşımını benimseyen eventbus yöntemi kullanılabilir. Bu yöntemde mikro önyüzlerden birisi bir veri paylaştığı zaman, paylaşılan veriye ihtiyaç duyan diğer mikro önyüz(ler) kullanır. Üretici/tüketici yaklaşımında tarayıcıların kendine has (native) araçları kullanılabileceği gibi projenin durumuna göre ortak bir kütüphane de kullanılabilir.



Şekil 11. Mikro Önyüzler Arası Veri Paylaşımı

Kullanıcıların oturum verileri gibi her bir mikro önyüzün sürekli ihtiyaç duyacağı veriler ise tarayıcıların kalıcı veri depolarında (session storage, local storage, cookie) tutulabilir. Mikro önyüzler arası veri paylaşımı için Redux gibi veri yönetim (state management) kütüphanelerinin kullanılması tavsiye edilen bir yaklaşım değildir. Eğer Redux gibi bir kütüphanenin kullanılması ile mikro önyüzler tarafından sürekli olarak ortak bir takım verilerin manipülasyonu söz konusu ise mimari açıdan yanlış kararlar alınmış olabilir, tekrar değerlendirme yapılarak alınan mimari kararlar gözden geçirilmelidir.

Karşılaşılabilecek Problemler ve Çözümleri

Mikro önyüzler birbirlerinden bağımsız oldukları ve ayrı ayrı geliştirilebildikleri için tutarlı bir arayüzü sağlamak kolay değildir. Aynı anda birden fazla mikro önyüzün aynı ekranda bir arada yer aldığı durumda kendisini daha çok belli edebilecek bu problem, ortak bileşen kütüphaneleri kullanılarak çözülebilir. Uygulamayı oluşturan farklı mikro önyüzlerde farklı teknolojilerin kullanılabileceği göz önüne alındığında ortak bileşen kütüphanelerinin tüm teknolojiler ile uyumluluğu ciddi öneme sahiptir. Ekran ve bileşenlerin tutarlılığını sağlamak için ya tüm teknolojilere desteği bulunan bileşen kütüphaneleri tercih edilmeli ya da tarayıcıların kendine has API'leri kullanılarak oluşturulan Web Component bileşen kütüphaneleri kullanılmalıdır. Şirketin ve projenin yapısı eğer uygun ise şirkete veya projeye özel Web Component bileşen kütüphanesi de oluşturulabilir. Eğer Web Component bileşen kütüphanesi oluşturulmaya karar verilir ise çok karmaşık bileşenler içermemesine dikkat edilmelidir. Zira karmaşık bileşenlerde herhangi bir güncelleme sonucunda bileşenin kullanımı ile ilgili bir değişimin olma ihtimali basit bileşenlere göre daha yüksektir. Projelerin ekran bütünlüğünü

sağlamak için sürüm yükseltmelerinde eski kullanımların güncellenmesi (refactor), dolayısı ile tüm projelerin sürüm yükseltme sırasında birbirine bağımlı olması mikro önyüz mimarisine ters bir durumdur. Bu sebeple ortak kütüphane kullanımı tavsiye edilmez iken, kullanılması halinde ortak kütüphanelerin olgun ve basit kütüphaneler olması oldukça önemlidir.

Ekranlar arasında bütünlüğü ve özelleştirmeleri desteklerken karşılaşılabilecek bir diğer problem ise stillerin yani CSS'lerin birbirlerini ezmesi durumudur. Mikro önyüzlerden birisinin CSS'i aynı ekranda yer alan başka bir mikro önyüzün bileşenlerini etkileyebilir. Bu durumun üstesinden gelebilmek için proje kapsamındaki tüm mikro önyüzlerin CSS'lerine ayırt edici ön ekler (prefix) eklenebilir.

Mikro önyüz mimarisine sahip uygulamaları geliştirirken mikro önyüzleri, ortamları ve bağımlılıkları daha kolay yönetebilmek için mikro önyüz geliştirme üzerine oluşturulmuş kütüphanelerden de (Single-SPA, Bit, Module Federation, vb.) faydalanılabilir. Kütüphane tercihi yapılırken projenin yapısı ve mikroservislerin, dolayısı ile mikro önyüzlerin büyüklükleri dikkate alınmalıdır.

Önemli Noktalar

Mikro önyüz mimarisinin mikroservis mimarisi ile birlikte kullanıldığı zaman büyük ve karmaşık projeler için sağladığı faydalar inkar edilemez. Ancak günümüzde, bu faydalara ihtiyaç duyan veya duymayan birçok şirket, sırf popüler olduğu için mikro önyüz mimarisini kullanmaya çalışmaktadır. Her proje paydaşının istediğini yapabilme özgürlüğü olsa da kötü bir yaklaşımın projeye vereceği zararlar göz ardı edilmemelidir. Örneğin; ihtiyaç duyulmadığı halde mikro önyüz mimarisinin kullanılıyor olması uygulamanın bir araya getirilme ihtiyacından dolayı daha yavaş açılmasına, dağıtık yapısından dolayı kod bakımının zorlaşmasına, geliştirme sırasında uçtan uca testlerin (end-to-end testing) daha zor yapılmasına veya gerekçesiz kararlar alınarak farklı geliştirme opsiyonlarının (örneğin iki farklı SPA'nın birlikte kullanılması) bir arada kullanılmasına, dolayısı ile de performans problemlerinin ortaya çıkmasına neden olabilir. Diğer taraftan ihtiyaç olmadığı halde gereksiz yere mikro önyüz mimarisinin kullanımının yanında, gerçekten bu mimariye ihtiyaç duyulan projelerde sırf yanlış anlaşılmalardan kaynaklı söz konusu mimarinin hatalı uygulanması da sık rastlanan problemlerdendir. Yanlış anlaşılma sebebiyle mikro önyüzler çok küçültülerek birer bileşene dönüştürülmemeli, birbirleri ile sürekli veri alışverişi yaparak birbirleri arasında bağımlılıklar (bounded-context) oluşturmamalı ve kullandıkları ortak kütüphaneler ile sürüm çıkışları sırasında bağımlılıklar oluşturarak bağımsızlığı engellememelidir. Mikro önyüzlerde uygulama bütünü açısından önemli olabilecek özelliklerin kaybedilmesine neden olacak bireysel ve gerekçesiz kararların alınmadığından (gerekçesiz olarak aynı anda farklı SPA'ların kullanılmadığından) emin olunmalıdır. Mimari kararlar hızlı ve kısa araştırmalar sonucu veya kulaktan duyma bilgiler ile alınabilecek kararlar değillerdir. Martin Fowler'ın da belirttiği gibi oldukça önemli ve zor kararlardır. Dolayısı ile bilinçsiz bir şekilde mikro önyüz mimarisinin kullanımının ortaya çıkarabileceği dezavantajlar dikkate alınarak mikro önyüz mimarisinin kullanımı değerlendirilmelidir.

Sonuç ve Öneriler

Dünyada baş döndürücü gelişim ve değişimin yaşandığı günümüzde işler ve işlerin yapılış şekilleri de hızla değişmektedir. Yazılımın günümüz teknolojisinin en güncel alanı olduğu gibi yazılım projeleri de ihtiyaç duydukları araç ve teknolojilerde en günceli kullanmalıdır. Yazılım projelerinin değişen yeni ihtiyaç ve teknolojilere göre kendisini güncel tutabilmesi için gelişmeye açık bir mimari ile desteklenmesi çok önemlidir. Bu gelişmelere ayak uydurabilmek için uygulamaların önyüz tarafında kullanılabilecek öncelikli mimari mikro önyüz (frontend) mimarisidir. Mikro önyüz mimarisi sağlamış olduğu dağıtık yapı gereği bağımsız ve küçük projelerden oluşur. Bu sayede uygulamanın önyüz kısmı bir taraftan birçok kişinin veya takımın aynı anda çalışmasıyla hızlı ve verimli şekilde geliştirilirken, diğer taraftan kolayca güncellenebilme imkânını elde etmiş olur. Ancak bu mimari oluşturulurken uç noktalara gidilerek oluşturulan mikro önyüzlerin fazla küçük olması, sürekli olarak veri alışverişi yapması, kütüphane veya araçlar ile birbirine sıkı bir şekilde bağımlı olması gibi sık yapılan hatalardan kaçınılmalıdır. Bu hatalara meydan vermeden yapılan ve proje iş alanına (domain) uygun olarak geliştirilen projeler uzun süre başarılı şekilde hizmet verebilir.

Kaynakça

1. Mezzalira, L. (2021). Building Micro-Frontends. California: O'Reilly Media, Inc.

2. Schmelmer, T., & Carneiro, C. (2016). Microservices: The What and the Why. T. Schmelmer, & C. Carneiro içinde, Microservices from Day One (s. 3). New York: Apress.

3. https://dev.to/okmttdhr/micro-frontends-architecture-patterns-mfe-in-3-minutes-l0p

4. https://lucamezzalira.medium.com/micro-frontends-decisions-framework-ebcd22256513

TÜBİTAK

BİLGEM

YTE | YAZILIM TEKNOLOJİLERİ ARAŞTIRMA ENSTİTÜSÜ

İşçi Blokları Mahallesi Muhsin Yazıcıoğlu Caddesi No:51/C 06530 Çankaya/ANKARA

+90 (312) 289 92 22 - yte.bilgem@tubitak.gov.tr

TÜBİTAK - BİLGEM Yazılım Teknolojileri Araştırma Enstitüsü (YTE)

yte.bilgem.tubitak.gov.tr