ThoughtWorks®

TEKNOLOJİ RADARI *KASIM '16*

Geleceği şekillendiren
eğilimler ve teknoloji üzerine
görüşlerimiz

thoughtworks.com/radar
#TWTechRadar

YENİLİKLER

Bu sayıda öne çıkan konular şunlar:

"PROSES OLARAK DOCKER, MAKİNE OLARAK PAAS, PROGRAMLAMA MODELİ OLARAK MİKRO SERVİS MİMARİSİ"

Mikro servis tarzı mimari, konteynerleştirme ve düşük seviyeli bağdaştırmanın vurgulanması nedeniyle geliştiricilerin dünyasında artan soyutlamalara dikkat çekiyor ve yüksek seviyede operasyonel izolasyon sağlıyor. Geliştiriciler konteyneri kendi kendine yeten bir proses olarak, PaaS'ı ise mikro servis mimarisini ortak tarz olarak kullanan ortak kurulum hedefi olarak düşünebilirler. Mimarinin ayrıştırılması ekipler için aynı olanağı sağlar ve silolar arasında koordinasyon masrafını aşağı çeker. Hem geliştiriciler hem de DevOp'lar için cazip olması, bunun birçok kuruluşta yeni geliştirme için fiili standart haline gelmesini sağladı.

AKILLI YETKİLENDİRME

Makine öğrenimi ve yapay zeka gibi uzun zamandır Ar-Ge çalışmalarının gündemini oluşturan konular, birdenbire Nuance Mix ve TensorFlow gibi framework'ler üzerinden pratik uygulama alanları buldular. Geliştiriciler, NLP'den makine öğrenimi kütüphanelerine kadar uzanan framework'leri indirebilirler. Şirketlerin, 10 yıl önce astronomik fiyatları nedeniyle kısıtlı olan bu alandaki sofistike kütüphane ve araçları çoğu zaman açık kaynak yaptıklarını ve geliştiriciler arasında geniş bir alıcı kitlenin erişimine açtıklarını memnuniyetle görüyoruz. Dönüşüm geçiren ve bir araya gelen birçok etken yeni araçlar yapılmasını mümkün kıldı: GPU (Grafik İşleme Ünitesi) ve bulut kaynakları gibi belli donanımları hedefleyen sıradan bilişim. Belki de istiflediğiniz Büyük Veri işe yaramaya başlıyor...

EKİP YAPISININ BÜTÜNSEL ETKİSİ

Çok çeşitli yazılım geliştirme konuları üzerinde büyük etkisi olan ekip yapısı, kendi kendine PaaS ve mikro servisler gibi sebeplerle giderek büyüyen bir ilgi odağı haline geliyor. Şirketler şu anda projelerden çok ürün üzerine düşünmeyi tercih ediyor; teknoloji şirketleri "sen yap, sen çalıştır" tarzı ekip otonomisini yaygınlaştırıyor ve biz aynı ürün düşünme yönteminin kurumsal projelere de uygulanmakta olduğunu görüyoruz. Ekiplerin yeniden yapılandırılmasının daha iyi sonuçlar vermesi, yazılım geliştirmenin daha çok bir iletişim sorunu olduğuna yeni bir örnek oluşturuyor. İşlevler arası ekipler oluşturmak, geleneksel olarak ayrı olan görevler arasında yararlı bir iletişim yüzeyi alanının genişlemesini sağlıyor ve bu da silolar gibi yapay yapıların neden olduğu engel ve sürtünmeleri ortadan kaldırır.

ANA AKIM DOĞRULTUSUNDA AR VE VR KOLAYLIĞI

Artırılmış ve sanal gerçekliğin (AR/VR) iş dünyasında ilgi topladığını görüyoruz. Her iki teknoloji bir zamanlar sadece oyunlardan ve ilginç yeniliklerden ibaret olmaya indirgeniyordu. Sanal çizgi filmlere büyük ilgi nedeniyle AR, mobil SDK'lar aracılığıyla toplumun ilgi odağına taşınırken, Oculus Rift, HTC Vive ve Microsoft HoloLens örneklerindeki gibi donanım, ilk benimseyenlerin olgunlaşmamış teknolojilerde boş yere arama tarama yapmadan bunun faydalarını hasat edebilecekleri noktaya kadar olgunlaşıyor. OpenVR ve Unity gibi yazılım platformları uzun zamandır olgunlaşmış olmasına rağmen, Nuance Mix gibi yeni doğal dil işleme (NLP) araçlarının ve doğal etkileşimler sağlayan donanımın, AR ve VR'nin benimsenmesi üzerinde çok büyük etkisi olacaktır. Şimdiden ofislerimizde, uzaktan etkileşimler veya bireylere yönelik yol bulma gibi gelecek uygulamaları araştırmak için VR and AR laboratuvarları yürütüyoruz. Deneylerimiz VR'nin, soyutlamayı baypas etme ve kullanıcıları doğrudan bir deneyin içine daldırması yeteneği sayesinde daha fazlasıyla, empatiyle uzaktan işbirliği kurmak ve hikaye anlatmak için sürpriz derecede güçlü bir ortam olduğunu kanıtlıyor. Ama özellikle şirkette olmak üzere, VR ve AR içeriğinin oluşturulması ve sunulmasında, yetenekler ve imkanların donanımın temposunun gerisinde kalmasından dolayı önemli zorluklar göreceğiz.

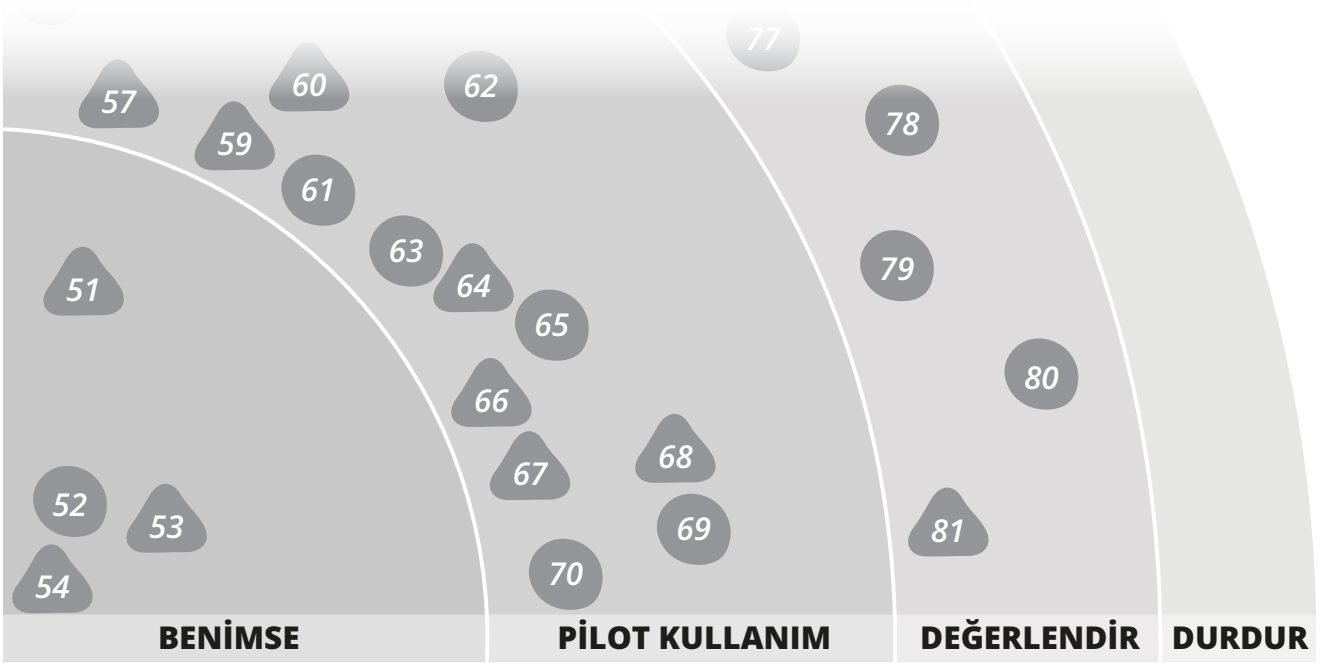
KATKIDA BULUNANLAR

Teknoloji Radarı aşağıdaki ThoughtWorks Teknoloji Danışma Kurulu tarafından hazırlanmıştır:

Rebecca Parsons (CTO)
Martin Fowler (Baş Bilim İnsanı)
Anne J Simmons
Badri Janakiraman
Bharani Subramaniam
Camilla Falconi Crispim
Erik Doernenburg
Evan Bottcher
Fausto de la Torre
Hao Xu
Ian Cartwright
James Lewis
Jiaxing Chen
Jonny LeRoy
Marco Valtas
Mike Mason
Neal Ford
Rachel Laycock
Scott Shaw
Srihari Srinivasan
Zhamak Dehghani

TEKNOLOJİ RADARI HAKKINDA

ThoughtWorks çalışanları için teknoloji bir tutkudur. Teknoloji üretiyoruz, araştırmalar ve deneyler yapıyoruz, teknolojiyi açık kaynak olarak sunuyoruz, teknoloji hakkında yazılar yazıyoruz ve herkes için teknolojiyi geliştirmek amacıyla sürekli çalışıyoruz. Hedefimiz, yazılımda mükemmeliyeti savunmak ve BT alanında devrim yapmaktır. Bu misyon doğrultusunda ThoughtWorks Teknoloji Radarı'nı çıkarıp paylaşıyoruz. Radarı, ThoughtWorks'teki üst düzey teknoloji liderlerinden oluşan ThoughtWorks Teknoloji Danışma Kurulu hazırlıyor. Kurul sık sık toplanarak, ThoughtWorks'ün küresel teknoloji stratejisini ve sektörümüzü ciddi olarak etkileyen teknoloji trendlerini tartışıyor. Radar, Teknoloji Danışma Kurulunun bu tartışmalarını CIO'lardan geliştiricilere kadar uzanan geniş bir paydaş yelpazesine hitap eden bir formatta sunuyor. İçeriğin az ve öz olması amaçlanıyor. Sizi bu teknolojileri daha detaylı bir şekilde incelemeye davet ediyoruz. Esas olarak grafiksel bir yayın olan Radar, konuları maddeleri teknikler, araçlar, platformlar ve diller ve çerçeveler bazında gruplandırıyor. Radarın birden fazla çeyrek dairede çıkabilen maddeleri için en uygun göründükleri çeyrek daireyi seçiyoruz. Bu maddeleri ayrıca şu anda onlar hakkındaki tavrımızı yansıtan dört halka halinde gruplandırıyoruz. Bu halkalar şunlardır:



Sektörün bu teknolojileri kullanması gerektiğine kesinlikle inanıyoruz, biz de yeri geldikçe kendi projelerimizde de kullanıyoruz.

Takip etmeye değer. Bu maddeleri kullanma kapasitesine nasıl sahip olabileceğinizi anlamak önemlidir. Kurumlar bu teknolojiyi, yalnızca riski kaldırabilecek projelerde bir deneme süreci olarak uygulamalıdır.

Kurumunuzu nasıl etkileyeceğini anlamak amacıyla takip etmekte fayda var.

Temkinli adım atın.

Yeni veya son radardan beri büyük değişim geçiren maddeler üçgen olarak, aynı kalan maddeler ise yuvarlak olarak verilir. Her bölümdeki detaylı grafikler maddelerin hareketlerini gösterir. Bu büyüklükteki bir belgeye sığdıramayacağımız kadar fazla madde ile ilgilendiğimiz için eski radardan kalan birçok maddeyi yenilerine yer açmak için siliyoruz. Fakat bu sildiğimiz ürünler artık umurumuzda değil anlamına gelmiyor.

Radar ile ilgili daha fazla bilgi için: thoughtworks.com/radar/faq

RADAR

TEKNİK

BENİMSE
1. Tüketici kaynaklı sözleşme testi
2. Kod olarak ardışık düzen yeni
3. Threat Modeling

PİLOT KULLANIM
4. Ürün olarak API'ler yeni
5. Bug Avı
6. Veri Gölü
7. AB'de PII verilerinin barındırılması
8. Hafif Mimari Karar Kayıtları yeni
9. Reaktif mimariler
10. Sunucusuz mimariler

DEĞERLENDİR
11. Müşteri tarafından yönlendirilen sorgulama yeni
12. Konteyner güvenlik taraması yeni
13. İçerik Güvenlik Politikaları
14. Türevsel Gizlilik yeni
15. Mikro Ön yüz yeni
16. OWASP ASVS
17. Unikernels
18. Oyun ötesi VR

DURDUR
19. Tüm ekipler için tek CI örneği
20. Zayıf REST yeni
21. Büyük veri özentisi
22. Buluta taşıma

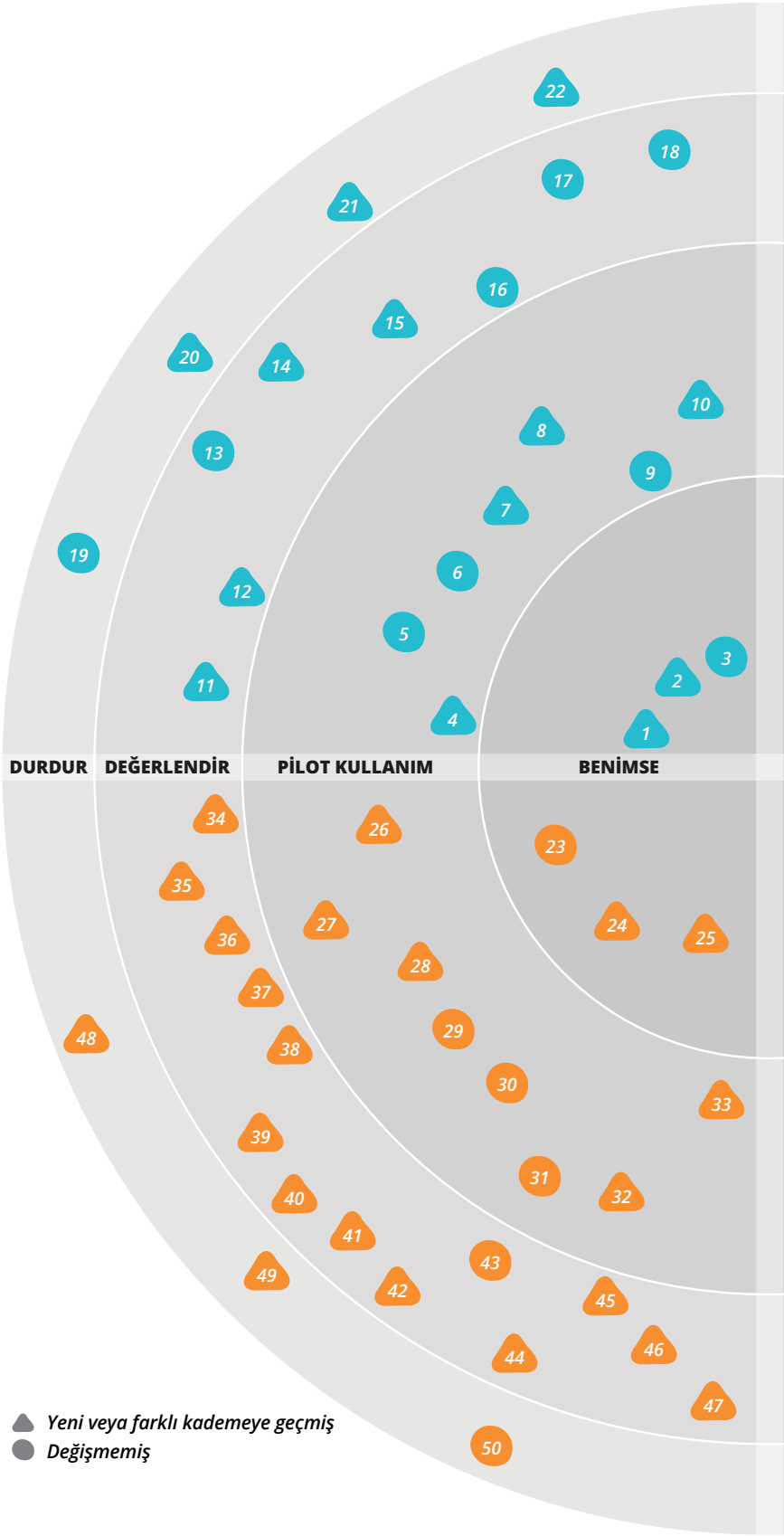
PLATFORMLAR

BENİMSE
23. Docker
24. HSTS
25. Linux güvenlik modülleri

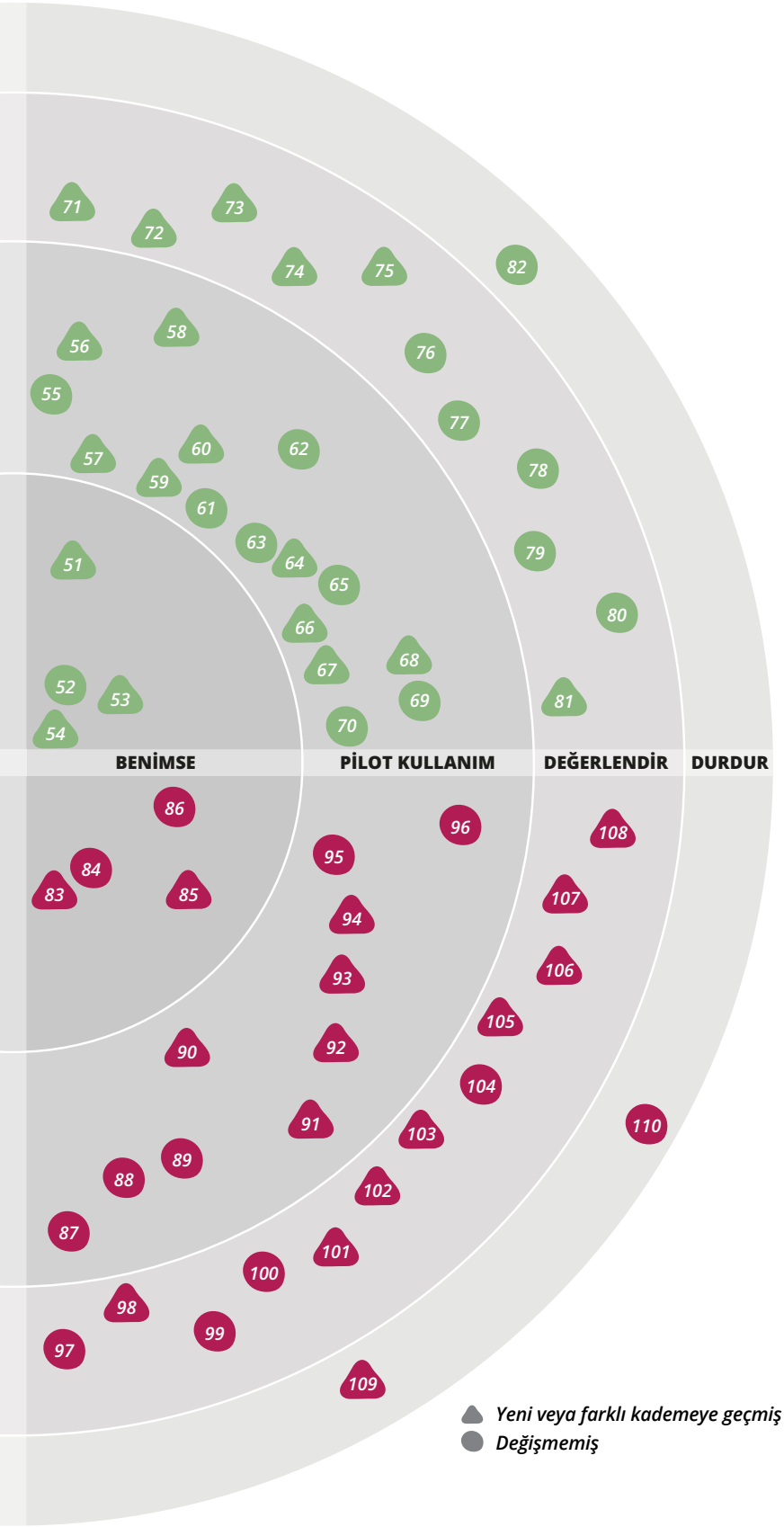
PİLOT KULLANIM
26. Apache Mesos
27. Auth0 yeni
28. AWS Lambda
29. Kubernetes
30. Pivotal Cloud Foundry
31. Rancher
32. Realm
33. Oyun Ötesi Unity yeni

DEĞERLENDİR
34. .NET Core
35. Amazon API Gateway
36. Apache Flink
37. AWS Application Load Balancer yeni
38. Cassandra: dikkatli kullanın yeni
39. Electron yeni
40. Ethereum yeni
41. HoloLens yeni
42. IndiaStack yeni
43. Nomad
44. Nuance Mix yeni
45. OpenVR yeni
46. Tarantool yeni
47. wit.ai yeni

DURDUR
48. Platform olarak CMS
49. Aşırı iddialı API gateway
50. Yüzeysel özel bulut



RADAR



ARAÇLAR

BENİMSE
51. Babel new
52. Consul
53. Grafana new
54. Packer

PİLOT KULLANIM
55. Apache Kafka
56. Espresso
57. fastlane new
58. Galen new
59. HashiCorp Vault
60. JSONassert new
61. Let's Encrypt
62. Load Impact
63. OWASP Dependency-Check
64. Pa11y new
65. Serverspec
66. Talisman new
67. Terraform
68. tmate new
69. Webpack
70. Zipkin

DEĞERLENDİR
71. Android-x86 new
72. axios new
73. Bottled Water new
74. Clojure.spec new
75. FBSnapshotTestcase new
76. Grasp
77. LambdaCD
78. Pinpoint
79. Pitest
80. Repsheet
81. Scikit-learn new

DURDUR
82. Kurulum ardışık düzeni olarak Jenkins

DİLLER VE FRAMEWORK'LER

BENİMSE
83. Ember.js
84. React.js
85. Redux
86. Spring Boot

PİLOT KULLANIM
87. Butterknife
88. Dagger
89. Dapper
90. Elixir
91. Enzyme new
92. Immutable.js
93. Phoenix new
94. Quick and Nimble new
95. React Native
96. Robolectric

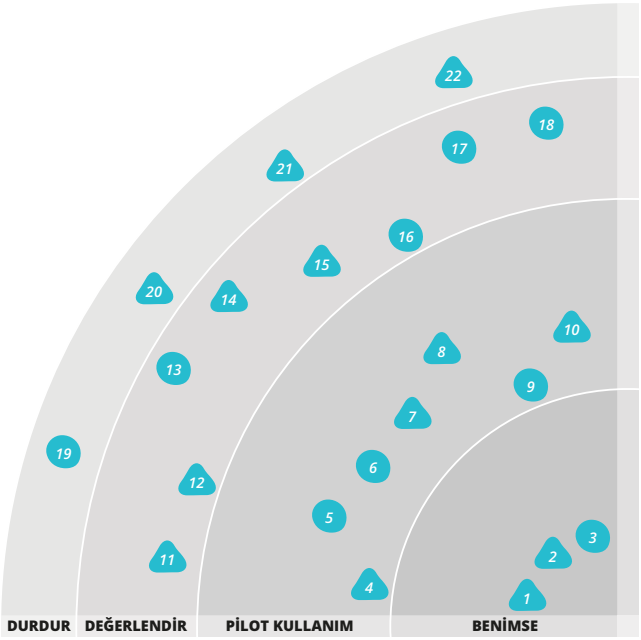
DEĞERLENDİR
97. Aurelia
98. ECMAScript 2017 new
99. Elm
100. GraphQL
101. JuMP new
102. Physical Web new
103. Rapidoid new
104. Recharts
105. ReSwift new
106. Three.js new
107. Vue.js new
108. WebRTC new

DURDUR
109. AngularJS
110. JSPatch

TEKNİKLER

Geçmişte unutulup gitmesine göz yumduğumuz **tüketici kaynaklı sözleşme testini** bu sayıda arşivden yeniden çıkarmaya karar verdik. Bu konsept yeni olmasa da ana akım olarak mikro servislerin kabul edilmesiyle birlikte, tüketici kaynaklı sözleşmelerin bağımsız hizmet kurulumlarını mümkün kılan gelişkin bir mikro servis testi işleminin esas parçalarından biri olduğunu hatırlatmamız gerekiyor. Ancak bunlara ilave olarak şunu belirtmek isteriz ki tüketici kaynaklı test, uygulanması için özel bir araç gerektirmeyen bir tekniktir. Pact gibi framework'leri seviyoruz çünkü bunlar belli bazı bağlamlarda düzgün bir sözleşme testi yapmayı kolaylaştırıyor. Ancak ekiplerde genel uygulamaya odaklanmaktan ziyade framework'lere odaklanma eğilimi olduğunu fark ettik. Pact testleri yazmak, tüketici kaynaklı sözleşmeler yarattığınızın garantisi değildir ve benzer şekilde, önceden hazırlanmış test aracı bulunmadığı durumlarda iyi bir tüketici kaynaklı sözleşme yaratmanız gereken birçok durum vardır.

Ekipler geliştirme altyapısı dahil olmak üzere ortamlarının her yerinde otomasyonu sağlamak için çalışıyorlar. **Kod olarak ardışık düzen**, işleyen bir CI/CD aracı konfigürasyonu yapmak yerine kod üzerinden bir kurulum ardışık düzeni tanımlamaktır.



LambdaCD, Drone, GoCD ve Concourse bu tekniğe imkan sağlayan örnekleridir. Aynı zamanda GoMatic gibi CI/CD sistemlerinin konfigürasyon otomasyonu araçları da kurulum ardışık düzenini kod gibi ele almak için kullanılabilir; bunların versiyonlanmış ve test edilmiş durumdadır.

Şirketler, iş imkanlarını hem şirket içindeki hem de şirket dışındaki geliştiricilere açan API'leri yürekten benimsediler. API'ler, şirketlerin en önemli olanaklarını yeniden bir araya getirerek yeni iş fikirleri üzerinde hızlı bir şekilde deneyler yapabilme olanağı vaat ediyor. Ancak bir API'yi sıradan bir işletme entegrasyon servisinden ayıran nedir? Bu farklardan biri, tüketicinin dahili bir sistem olduğunda bile **API'lere ürün olarak** muamele edilmesidir. API'leri inşa eden ekiplerin müşterilerinin ihtiyaçlarını anlaması ve ürünün onlar için zorunlu hale gelmesini sağlaması gerekir. Ürünler aynı zamanda uzun vadeli olarak iyileştirilmekte, bakımı yapılmakta ve desteklenmektedir. Bunların, müşterinin haklarını savunan ve sürekli iyileştirme için gayret gösteren bir sahipleri olmalıdır. Ürünler aktif bir şekilde bakımdan geçirilmeli ve desteklenmeli, bulunmaları ve kullanımları kolay olmalıdır. Bizim deneyimlerimizde sıradan kurumsal entegrasyon ile API'lerden oluşan bir platform üzerine kurulmuş çevik bir iş arasındaki farkı oluşturan şey ürün oryantasyonudur.

Birçok ülkede devlet kurumlarının özel kişisel tanımlama bilgilerine (PII) erişmek istediğini görüyoruz. Halka açık bulut çözümlerinin artan kullanımı, kuruluşların kullanıcıları tarafından kendilerine emanet edilmiş verileri korumalarını ve bir taraftan da ilgili tüm yasalara uymalarını zorlaştırıyor. Avrupa Birliği en ilerici gizlilik yasalarının bazılarına sahiptir ve en büyük bulut tedarikçileri - Amazon, Google ve Microsoft - Avrupa Birliği içinde çok sayıda veri merkezi ve bölgesi sunuyorlar. Bu nedenle, global kullanıcı tabanına sahip olanlar başta olmak üzere şirketlerin, en ileri gizlilik yasalarıyla korunan kullanıcılarının verileri için, **AB'de PII verilerini barındırarak** bir güvenli liman oluşturmanın fizibilitesini değerlendirmelerini tavsiye ediyoruz. Önceki Radar'da bu teknik hakkında yazdığımızdan beri tüm çalışanlarımızla ilgili hassas bilgileri işleyen yeni bir dahili sistemi kullanıma aldık ve bunu Avrupa Birliği'nde bulunan bir veri merkezinde barındırmayı tercih ettik

BENİMSE
1. Tüketici kaynaklı sözleşme testi
2. Kod olarak ardışık düzen - yeni
3. Threat Modeling

PİLOT KULLANIM
4. Ürün olarak API'ler - yeni
5. Bug Avı
6. Veri Gölü
7. AB'de PII verilerinin barındırılması
8. Hafif Mimari Karar Kayıtları – yeni
9. Reaktif mimariler
10. Sunucusuz mimariler

DEĞERLENDİR
11. Müşteri tarafından yönlendirilen sorgulama – yeni
12. Konteyner güvenlik taraması – yeni
13. İçerik Güvenlik Politikaları
14. Türevsel Gizlilik – yeni
15. Mikro Ön yüz – yeni
16. OWASP ASVS
17. Unikernels
18. Oyun ötesi VR

DURDUR
19. Tüm ekipler için tek CI örneği
20. Zayıf REST – yeni
21. Büyük veri özentisi
22. Buluta taşıma

TEKNİKLER *devamı*

Okunabilirliği yüksek olan kod ve testler yerine birçok dokümantasyon kullanılabilecek olmakla birlikte evrimsel mimari dünyasında, bazı tasarım kararlarının geleceğin ekip üyeleri ve dışarıdan bakanların faydalanması için kaydedilmesi önemlidir. **Hafif Mimari Karar Kayıtları** önemli mimari kararlarını ve bunların bağlam ve sonuçlarını kayıt altına almayı amaçlayan bir tekniktir. Bu kayıtlar çoğu zaman bir wiki veya ortak çalışma aracı içinde saklanmakla birlikte biz genellikle bunları basit bir işaretleme ile kaynak kod denetimi içinde saklamayı tercih ediyoruz.

Sunucusuz mimari uzun zamandır kullanılan uzun soluklu sanal makineler yerine, talep üzerine ortaya çıkan ve kullanıldıktan sonra hemen ortadan kalkan kısa vadeli bilişim gücünü getiren bir yaklaşımdır. Bir önceki radardan beri çeşitli ekiplerimiz «sunucusuz» bir tarz kullanarak uygulamaları üretime geçirdi. Ekiplerimizin beğendiği bu yaklaşım onlar açısından iyi işliyor ve biz bu yaklaşımı geçerli bir mimari tercihi olarak görüyoruz. Sunucusuz mimarinin ya hep ya hiç anlayışıyla yapılması gerekmemesi de önemlidir: Ekiplerimizden bazıları sistemlerinin yeni bir kısmının kurulumunu sunucusuz mimari kullanarak yaptılar, diğer kısımlarında ise geleneksel mimari yaklaşımına bağlı kaldılar.

API'lere RESTful yaklaşımlar konusunda insanların karşılaştığı yaklaşımların çoğu «Zayıf REST» anti-modeline yüklenebilecek olmakla birlikte, bazı kullanım senaryoları diğer yaklaşımları keşfetmeyi gerektiriyor.Özellikle uzun bir müşteri uygulamaları kuyruğunu (ve tüketici kaynaklı sözleşmeler kullanıyor olsalar bile muhtemelen yaygınlaşan API versiyonlarını) desteklemek zorunda olan ve API'lerinin büyük bir kısmı sonsuz liste tarzındaki aktivite beslemelerini destekleyen kuruluşlar, RESTful mimarilerde bazı limitlerle karşı karşıya kalabilirler. Bunlar bazen, istemci-sunucu etkileşiminde **müşteri tarafından yönlendirilen sorgulama** yaklaşımını kullanarak hafifletilebilir. Bu yaklaşımın müşterilerin kendilerine geri dönen verilerin hem içeriği hem de detaylandırılması üzerinde daha fazla kontrole sahip olduğu GraphQL ve Falcor'da başarıyla kullanılmakta olduğunu görüyoruz.Bu, servis katmanı üzerine daha fazla sorumluluk yükler ve yine de alttaki veri modeline sıkı bağlaşmaya neden olabilir ancak iyi modellenmiş RESTful API'ler sizde işlemiyorsa, bu yaklaşımın faydaları denemeye değebilir.

Docker tarafından başlatılan konteyner devrimi, uygulamaların ortamlar arasında taşınmasındaki engel ve sürtünmeyi büyük ölçüde azalttı ancak aynı zamanda nelerin üretime girebileceği konusundaki geleneksel kontrol mekanizmalarında oldukça büyük bir açığa neden oldu. **Konteyner güvenlik taraması** tekniği bu tehdit vektörüne karşı gerekli bir cevaptır.

Docker şu anda kendi güvenlik tarama araçlarını sağlıyor. Aynı şekilde CoreOS'da kendi araçlarını sağlıyor. Biz CIS Security Benchmarks'ı başarılı bir şekilde kullandık. Hangi yaklaşımı benimserseniz benimseyin, otomatik konteyner güvenlik validasyonu konusunun çok değerli olduğuna ve PaaS düşüncesinin gerekli bir parçası olduğuna inanıyoruz.

«Anonimleşmiş» yığınsal veri setlerinin, özellikle çoklu veri setlerinin birlikte çapraz referanslandırılmış olduğu durumlarda, bireyler hakkında bilgi verebileceği uzun zamandır biliniyor. Kişisel gizlilik konusunda kaygılar giderek artarken Apple ve Google gibi bazı şirketler bireysel gizliliği iyileştirirken çok sayıda kullanıcı hakkında yararlı mantıksal ve matematiksel analizler yapabilme imkanını da korumak için Türevsel Gizlilik tekniklerine dönüyor. **Türevsel Gizlilik**, bir veri tabanından yapılan istatistiksel sorgulamaların doğruluğunu maksimize ederken, kayıtlardan kimliklerin belli olması ihtimalini minimize etmeye çalışan bir şifreleme tekniğidir. Bu sonuçlar, verilere az miktarda «gürültü» ilave ederek elde edilebilir ancak bu konudaki araştırmaların henüz devam etmekte olduğunu unutmamakta fayda vardır. Apple ürünlerinde farklı güvenlik seviyeleri tekniklerini kullanma planını açıkladı ve biz de şirketin müşterinin gizliliğine gösterdiği özeni yürekten alkışladık ancak Apple'ın her zamanki ketumluğu karşısında bazı güvenlik uzmanları ne yapacağını şaşırmıştı.

Farklı bir yaklaşım olarak Datensparsamkeit'ı önermeye devam ediyoruz - sadece gerçekten ihtiyacınız olan minimum miktarda veri depolamak birçok durumda gizlilik açısından çok daha iyi sonuçlar verecektir.

Ekiplere bağımsız olarak kurulumu ve bakımı yapılan servislerin teslimatını ölçeklendirme yapma imkanı sunan Mikroservis mimarilerini kullanmaya başlamaktan büyük faydalar gördük. Ancak ekipler çoğu zaman ön yüz monolitleri oluşturmaktan kaçınmışlardı. Bu büyük ve genişleyen tarayıcı uygulamalarının bakımı ve dönüştürülmesi bizim terk ettiğimiz monolitik sunucu tarafı uygulamaları kadar zordur. Ekiplerimizin **Mikro ön yüzler** adını verdiği yeni bir yaklaşımın ortaya çıkmakta olduğunu görüyoruz. Bu yaklaşımda bir web uygulaması sayfalarına ve özelliklerine göre bölümlere ayrılıyor ve her özelliği bir ucundan diğer ucuna tek bir ekip ele alıyor. Uygulama özelliklerini uyumlu bir kullanıcı deneyimi olarak bir araya getirmek için kullanılan, bazıları eski bazıları yeni çeşitli teknikler bulunuyor ancak her özelliğin diğerlerinden bağımsız olarak geliştirilmesine, test edilmesine ve kurulmasına olanak sağlama amacı devam ediyor. BFF - backend for frontends yaklaşımı burada iyi işliyor ve her ekip kendi uygulama özelliklerini desteklemek için bir BFF geliştiriyor.

TEKNİKLER *devamı*

Backend for frontends (BFF) örüntüsünün popülerliği ve React.js gibi tek yönlü veri bağlama framework'lerinin kullanımı giderek artarken, REST tarzı mimarilere karşı bir tepki olduğunu fark ettik. Eleştirmenler REST'i sistemler arasında kuralsız ve verimsiz etkileşimlere neden olmak ve müşterilerinin dönüşüm ihtiyaçlarına ayak uyduramamakla suçluyorlar. Müşterinin geri dönen verinin formatını belirlemesine izin veren alternatif veri çekme mekanizmaları olarak GraphQL veya Falcor gibi framework'leri öneriyorlar. Ama bizim deneyimlerimiz bu sorunların kaynağının REST olmadığını gösteriyor. Bunlara daha çok, alanı bir kaynak seti olarak düzgün bir şekilde modelleyememek sebep oluyor. Sadece şablon URL'ler üzerinden statik, hiyerarşik veri modelleri açığa çıkaran servislerini naifçe geliştirmek bir **Zayıf REST** uygulamasıyla sonuçlanıyor. Geniş modellenmiş böyle bir alanda, REST'in basitçe tekrarlanan bir şekilde veri çekmekten daha fazlasına olanak sağlaması gerekir. Dönüşümünü tamamlamış RESTful mimarisinde iş etkinlikleri ve soyut kavramlar da kaynaklar olarak modellenir ve uygulamaya koymanın, üst metin, link ilişkileri ve medya tiplerini etkin bir şekilde kullanarak servisler arasında ayrıştırmayı maksimize etmesi gerekir. Bu bozuk örüntü, Zayıf Domain Model örüntüsüyle yakından ilişkilidir ve sonuçta Richardson Maturity Model'de alt sıralarda yer alan servisler ortaya çıkarır.' Insights yazımızda, etkili REST API'leri tasarlamaya yönelik tavsiyelerden başka tavsiyelerimiz de bulunuyor.

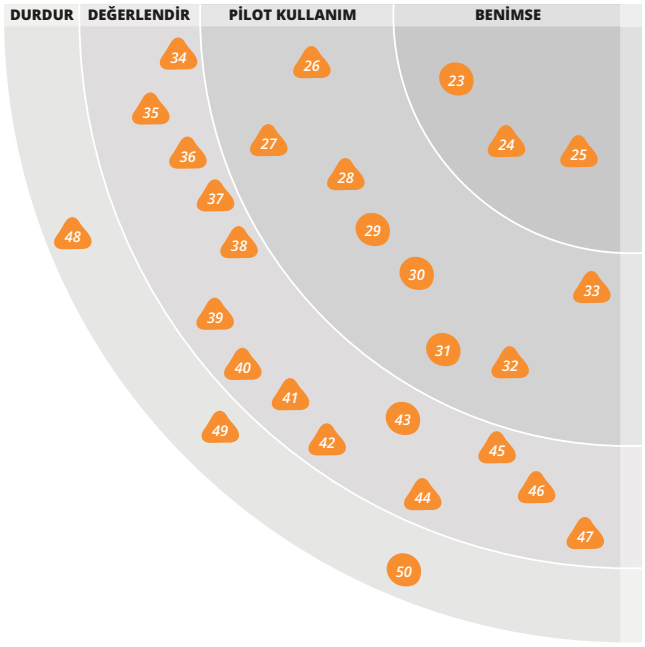
Kuruluşların «şık» teknolojiler peşinde koştuklarını ve daha basit bir çözüm daha iyi olacakken gereksiz karmaşıklık ve riskler üstlendiklerini görmeye devam ediyoruz. Özel konulardan biri, görece küçük veri setleri için, dağıtık «büyük veri» sistemleri kullanmaktadır. Bu davranış, **bizi büyük veri özentisini** bir kez daha beklemeye almaya ve son dönemdeki deneyimlerimizden verilerle ilgili ilave birkaç noktaya değinmeye yöneltiyor. Apache Cassandra veri tabanı, sıradan donanımlar üzerinde devasa bir ölçeklenebilirlik vaat ediyor ancak biz bunun mimari ve operasyonel karmaşıklığından bunalmış ekipler gördük.

100'den fazla aygıt kümesi gerektiren veri hacminiz olmadığı takdirde Cassandra kullanmamanızı tavsiye ediyoruz. Bu şeyin çalışması için bulundurmanız gerekecek işletme ekibi buna değmez. Radar'ın bu sayısını hazırlarken çeşitli yeni veri tabanı teknolojilerini ele aldık. Bunların birçoğu mevcut sistemlere göre «10 kat» performans artışı sağlıyor. Yeni teknolojiler - özellikle veri tabanı gibi kritik bir şey - tam olarak kendini kanıtlamadıkları sürece her zaman çok kuşkucu davranıyoruz. Zor koşullar altında veri tabanı performansı analizleri sunan Jepsen birçok NoSQL veri tabanında çok sayıda hata buldu. Veri tabanı teknolojilerini değerlendirirken sağlıklı bir dozda kuşkuculuğun korunmasını ve bir gözünüzün Jepsen gibi sitelerde olmasını tavsiye ediyoruz

Giderek daha fazla kuruluş uygulamalarını bulutta kurmayı tercih ederken, biz sürekli olarak, mevcut veri merkezi yönetimlerini ve güvenlik yaklaşımlarını bulutta tekrarlamak için boş yere çalışan BT grupları buluyoruz. Bu, çoğu zaman, üzerinde çok fazla yeniden düşünülmeden buluta genişletilen firewall'lar, yük dengeleyiciler, ağ proxy'leri, erişim kontrolleri, güvenlik cihazları ve servisleri şeklinde ortaya çıkar. Bulut sağlayıcılar karşısında, ekiplerin kullanabileceği servileri sınırlamak için kendi orkestrasyon API'lerini kuran kuruluşlar görüyoruz. Çoğu durumda bu katmanlar sadece kapasiteyi azaltmaya hizmet ediyor ve buluta geçmekten beklenen faydaların çoğunu ortadan kaldırıyor. Radar'ın bu sayısında buluta taşımayı (**cloud lift and shift**) kaçınılması gereken bir teknik olarak yeniden vurgulamayı tercih ettik. Kuruluşlar bunun yerine kendi mevcut güvenlik ve operasyonel kontrollerini daha derin bir bakışla incelemeli, gereksiz engellemeler yaratmadan bulutta çalışan alternatif kontroller aramalıdırlar. Bu kontrollerin çoğu gelişmiş bulut sağlayıcılar için zaten mevcut olacaktır ve bulutu benimseyen ekipler kendi kendine kullanım ve işletim için yerli API'leri kullanabilirler.

PLATFORMLAR

HTTP Strict Transport Security (**HSTS**) şu anda çok yaygın destek gören ve web sitelerinin kendilerini sürüm düşürme saldırılarından korumalarını sağlayan bir politikadır. HTTPS bağlamında, sürüm düşürme saldırısı, sitenizin kullanıcılarının HTTPS yerine HTTP'ye düşmelerine neden olan ve örneğin iki bağlantı noktası arasındaki bağlantının izinsiz izlenmesi gibi başka saldırıları mümkün hale getiren bir saldırıdır. HSTS İLE sunucu, tarayıcıya web sitesine erişmek için sadece HTTPS kullanması gerektiği bilgisini veren bir başlık gönderiyor. Tarayıcı desteği, artık bu uygulaması kolay özelliğin HTTPS kullanan her site için düşünülmesine yetecek kadar yaygınlaştı. Mozilla'nın Observatory yazılımı, güvenlik ve gizliliği artıran bu ve başka başlık ve konfigürasyon seçeneklerini tanımlamaya yardımcı olabilir. HSTS'yi uygularken tüm kaynakların HTTPS üzerine düzgün bir şekilde yüklendiğini kontrol etmek önemlidir çünkü HSTS bir defa açıldığı zaman, süre sonuna kadar (hemen hemen) hiçbir geri dönüş yoktur. Alt alanları dahil etme direktifi de eklenmelidir ancak yine tüm alt alanların güvenli taşımayı desteklediği konusunda tam bir kontrol yapılmalıdır.



Uygulama beyaz listesinin siber sızma saldırılarını hafifletmek için en etkili yollardan biri olduğu kanıtlanmıştır ve yaygın bir şekilde tavsiye edilen bu yöntemi uygulamak için kullanışlı yollardan biri **Linux güvenlik modülleridir**. Linux versiyonlarının çoğunda fabrika ayarı olarak bulunan SELinux veya AppArmour ile birlikte ve kolaylıkla bulunabilen Grsecurity gibi daha geniş kapsamlı araçlar ile birlikte bu teknolojiyi bu sayıda Benimse halkasına geçirdik. Bu araçlar, ekiplere, paylaşılan ana bilgisayar üzerinde içerdiği hizmetler de dahil olmak üzere kimin hangi kaynaklara erişim hakkının olacağı konusundaki soruları değerlendirmekte yardımcı oluyor. Erişim yönetimine bu muhafazakar yaklaşım ekiplerin SDLC prosesleri içinde güvenliği geliştirmelerine yardımcı olacaktır.

Yüksek derecede dağıtık sistemler için küme kaynaklarını yönetmek amacıyla **Apache Mesos** platformunu kullanmak konusunda olumlu deneyimler yaşamaya devam ettik. Mesos, etkin kullanım sağlarken izolasyonu da sürdürmek amacıyla CPU ve depolama gibi alt yapıdaki bilgi işlem kaynaklarını sanallaştırır. Mesas kapsamında, programlı işlerin dağıtık ve hata toleranslı uygulaması için Chronos ve uzun vadeli proseslerin konteynerler içinde orkestrasyonunun yapılması için Marathon yer almaktadır.

Senaryoların çoğunda kendi kimlik doğrulama kodunuzu kullanmaya nadiren değdiğine olan inancımız giderek artıyor. Dış kaynaktan sağlanan kimlik yönetimi teslimi hızlandırır, hataları azaltır ve yeni keşfedilen zaaflara daha hızlı karşılık verme olanağı sağlayabilir.

Auth0, entegrasyon kolaylığı, desteklenen protokol ve bağlantı elemanları yelpazesinin genişliği ve zengin yönetim API'si ile bu alanda bizde özellikle olumlu izlenim bıraktı.

BENİMSE
23. Docker
24. HSTS
25. Linux güvenlik modülleri

PİLOT KULLANIM
26. Apache Mesos
27. Auth0 – yeni
28. AWS Lambda
29. Kubernetes
30. Pivotal Cloud Foundry
31. Rancher
32. Realm
33. Oyun Ötesi Unity - yeni

DEĞERLENDİR
34. NET Core
35. Amazon API Gateway
36. Apache Flink
37. AWS Application Load Balancer – yeni
38. Cassandra: dikkatli kullanın – yeni
39. Electron – yeni
40. Ethereum - yeni
41. HoloLens – yeni
42. IndiaStack – yeni
43. Nomad
44. Nuance Mix – yeni
45. OpenVR – yeni
46. Tarantool – yeni
47. wit.ai – yeni

DURDUR
48. Platform olarak CMS
49. Aşırı iddialı API gateway
50. Yüzeysel özel bulut

PLATFORMLAR *devamı*

Ekiplerimiz **AWS Lambda**'yı keyifle kullanıyorlar ve sunucusuz mimarilerle deneyimlemek için kullanmaya başlıyor ve Lambda'yı API Gateway ile birleştiriyorlar. Lambda fonksiyonlarının sadece makul miktarda kod içermesini tavsiye ediyoruz. Birçok büyük lambda fonksiyonlarından oluşan karışık bir gruba dayalı olarak kalitenin garanti edilmesi zordur ve böyle bir çözüm çok uygun maliyetli olmayacaktır. Daha karmaşık ihtiyaçlar için, konteynerlere veya VM'lere dayalı kurulumlar hâlâ tercih edilebilir durumdadır. Ayrıca, Java'yı Lambda fonksiyonları için kullanırken önemli sorunlar yaşadık ve Lambda konteyneri çalıştırıldığında birkaç saniyeye kadar çıkabilen değişken gecikmeler oldu. Tabii ki JavaScript veya Python kullanılarak bunlardan kaçınabilirsiniz ve Lambda fonksiyonları çok fazla kod içermiyorsa programlama dili tercihi çok önemli olmayacaktır.

Realm mobil cihazlarda kullanılmak üzere tasarlanmış bir veri tabanıdır ve kendi kaydetme mekanizmasıyla yüksek performans sağlar. Realm SQLite ve Core Data yerine geçecek bir ürün olarak pazarlanıyor. Taşımaların, Realm belgelerinin size düşündürdüğü kadar düz ve kolay olmadığını bilmelisiniz. Ancak giderek daha fazla ekip, mobil uygulamalar için üretim ortamlarında kaydetme mekanizması olarak Realm'i tercih ediyor.

Oyun geliştirme platformu olarak şimdiden yıllardır süren bir büyüme deneyimi yaşamış durumda olan **Unity**, yakın geçmişte VR ve AR uygulamaları geliştirmek için tercih edilen platform oldu. İster Oculus ya da HTC Vive başlıkları için tamamen insanı içine alan, sürükleyici bir dünya, ister yeni mekansal kurumsal uygulamanız için bir holografik katman ya da isterseniz mobil uygulamanız için bir AR özellik dizisi yaratıyor olun, Unity onun hem prototipini geliştirmeniz, hem de kullanımın yoğun olduğu zamana hazır hale getirmeniz için neye ihtiyacınız varsa sağlamaktadır. ThoughtWorks'teki birçoğumuz VR ve AR'nin bilişim platformunda sıradaki önemli sıçramayı temsil ettiklerine inanıyoruz ve şu an için Unity bu değişimi geliştirmek için kullandığımız, araç gereç kutusundaki en önemli araçtır. Unity'yi başlıklar ve telefon/tablet uygulamaları için AR işlevselliğinde kullandığımız gibi, bütün VR prototiplerimizi geliştirmekte de kullandık.

.NET Core; Windows, macOS ve Linux içinde kolayca kurulabilen uygulamalar geliştirmek için açık kaynaklı bir modüler üründür. .NET Core, ASP.NET Core'u araçlar, kütüphaneler ve framework'ler kümesi ile birlikte kullanarak web uygulamaları oluşturmayı mümkün kılar, mikro servis mimarisi için başka bir seçenektir. .Net Core ve diğer ilgili projeler çevresindeki topluluk giderek büyümekte ve de Visual Studio Code gibi yeni araçlar ortaya çıktı ve hızla dönüşüm geçirdi.

Hem Linux bazlı hem de mikro servis mimarisini uygulamayı basitleştiren .Net Core'un altında çalıştığı Windows (Nano Sunucu) bazlı Docker imajlar vardır. CoreCLR ve CoreFX geçmişte Radar'da yer aldı, ama birkaç ay önce Microsoft ilk stabil sürüm olan .Net Core 1.0'i çıkardığını duyurdu. Çok iyi yeni fırsatlar, değişimler ve canlı bir topluluk gördüğümüz için bu ürünü değerlendirmeye devam edeceğiz.

Amazon'un, geliştiricilerin API hizmetlerini İnternet kullanıcılarına açabilmelerini sağlayan **Amazon API Gateway** sunumu, trafik yönetimi, izleme, kimlik doğrulama ve izin verme gibi alışılmış API Gateway özelliklerini içinde barındırıyor. Bizim ekiplerimiz sunucusuz mimarilerin parçası olarak AWS Lambda gibi diğer AWS yeteneklerinin karşısına çıkmak için bu hizmeti kullanageldiler. Biz aşırı iddialı API Gateway'lerin ortaya çıkardığı zorluklar için izlemede kalmaya devam edeceğiz, ama bu aşamada Amazon'un sunumu o problemleri yok etmek için yeteri kadar hafif gözüküyor.

Yeni nesil ölçeklendirilebilir dağıtık yığın ve akan veri işleme platformu olan **Apache Flink** için ilgi oluşmaya devam ediyor. Apache Flink'in çekirdeğinde akan bir veri-akımı motoru bulunuyor ve buna sekmeli yapıdaki işlemler (SQL benzeri), görüntü işleme ve makine öğrenimi işlemleri desteği eşlik ediyor. Apache Flink akan veri işlemeyle ilgili özellikler açısından zengin yeteneği ile göze çarpıyor: olay zamanı, zengin akımlı pencere operasyonları, hata toleransı ve tam olarak bir defalık semantikler. Kayda değer miktarda devam etmekte olan aktivite gösteren proje en son çıkan 1.1 sürümü ile iyileştirilmiş akıtma özelliklerinin yanı sıra yeni veri kaynağı/alıcısı entegrasyonlarını getiriyor.

Amazon kısa bir süre önce **AWS Application Load Balancer**'ı (ALB) piyasaya sürdü. 2009'da sunulmuş olan Elastic Load Balancers için doğrudan bir yenileme bu. Katman 7 trafik denetlemeyi destekleyen ALB, modern bulut mimarisini desteklemek için oluşturuldu. Eğer ECS kullanan mikro servis tabanlı bir sistem oluşturuyorsanız, yeni Load Balancer'lar her EC2 örneği için çoklu konteyner ve kapıların bulunduğu konteyner barındırma ve ölçeklendirmeyi doğrudan kabul edecektir. İçerik tabanlı yönlendirme, hedef sunucu gruplarının bağımsız olarak ölçeklendirilmesinin yanı sıra, taleplerin bu gruplar üzerinde segmentasyona izin veriyor. Nihayetinde Load Balancer'lar tarafından yapılan sistem durumu denetimleri çok iyileştirildi ve uygulama performansı hakkında detaylı ölçümler elde etme yeteneği eklendi. Burada gördüğümüz her şeyi beğendik ve ekipler ALB'nin başarılı kullanımını raporlamaya başladılar.

PLATFORMLAR *devamı*

Apache'nin Cassandra veri bankası büyük miktarda veriyi saklamak ve işlemek için güçlü, ölçeklendirilebilir büyük veri çözümüdür, çoğunlukla dünya üzerinde birçok lokasyona dağılmış yüzlerce düğümü kullanır. Biz bu harika aracı seviyoruz ama kullanan ekiplerin fazlasıyla sık sorunlar yaşadığını görüyoruz. **Cassandra'yı dikkatlice** kullanmanızı öneririz. Ekipler çoğunlukla Cassandra'nın kullanılacağı durumları yanlış anlıyor ve onu genel-amaçlı veri saklamak için kullanmaya kalkıyorlar, oysa aslında önceden tanımlanmış anahtarlar ya da endeksleri esas alarak geniş veri kümelerinde hızlı okumalar yapmak için optimize edilmiştir. Saklama şemasına bağımlılığı, zaman içinde dönüştürülmesini zorlaştırabilir de. Cassandra ayrıca kayda değer operasyonel karmaşıklığa ve bazı zorlu sınırlamalara sahiptir, dolayısıyla onun sağladığı büyük ölçeklere kesinlikle ihtiyacınız olduğunu düşünmüyorsanız, daha basit bir çözüm genellikle daha iyidir. Eğer Cassandra'nın özel kullanım durumuna ve ölçeklendirme özelliklerine ihtiyacınız yoksa demek ki onu sadece büyük veri özentisi yüzünden seçiyor olabilirsiniz. Cassandra'nın dikkatli kullanımı geniş çaplı otomatik testleri de kapsayacaktır ve test etme stratejinizin bir parçası olarak CassandraUnit'i kullanmanızı memnuniyetle öneriyoruz.

Electron; HTML, CSS ve JavaScript gibi web teknolojilerini kullanan yerel masaüstü istemciler oluşturmak için sağlam ve dayanıklı bir framework'tür. Ekipler kendi web know-how'larını verimli kullanarak başka bir teknoloji grubu daha öğrenmeye zaman harcamadan gösterişli platformlar arası masaüstü istemciler oluşturabilirler.

Blockchain ve şifrelenmiş para birimi için heyecanın zirvesi geride kalmış gözüküyor, bunun kanıtı da bu konuda önceki yangın hortumu misali açıklamalardan damlayan musluğa dönülmüş olmasıdır. Daha spekülatif çabaların bazılarının zaman içinde söneceğini tahmin ediyoruz. Blockchain'lerden biri olan **Ethereum** iyi aşama kaydediyor ve izlenmeyi hak ediyor. Ethereum halka açık bir blockchain'dir, dahili programlama dili sayesinde kendi içinde "akıllı kontratlar" oluşturulmasına imkan tanır. Bunlar "ether"in (Ethereum şifreli para birimi) blockchain'ler üzerinde oluşan aktiviteye cevaben algoritmik hamleleridir. Bankalar için blockchain oluşturma konsorsiyumu olan R3Cev, Ethereum üzerinde ilk kavram kanıtlarını oluşturdu. Ethereum, ilk "algoritmik şirket"lerden olan bir DAO (Distributed Autonomous Organization - Dağıtık Otonom Organizasyon) oluşturmak için kullanıldı, diğer yandan son zamanlardaki 150 milyon dolarlık Ether soygunu blockchain ve şifreli para birimlerinin hâlâ teknoloji dünyasının Vahşi Batı'sı olduğunu kanıtlıyor.

HoloLens'te Microsoft ilk defa gerçekten kullanılabilir AR başlığını piyasaya sürdü. Bu başlık, sadece endüstriyel tasarımın güzel bir eseri ve giyilmesi fazlasıyla konforlu bir cihaz değil, aynı zamanda muhteşem optik özellikleri ve derin Windows 10 entegrasyonu aracılığıyla, kurumlar için AR'nin vaat ettiklerinin açık bir kanıtıdır. HoloLens'in yakın zaman içinde müşterilerimize üzerinde esaslı bir uygulama işlevselliği temin ettiğimiz ilk AR platformu olmasını bekliyoruz ve giderek daha fazla çekim gücü kazanırken dönüşmesini dört gözle bekliyoruz.

IndiaStack Hindistan'ın veri fakiri bir ülke olmaktan çıkıp, veri zengini bir ülke haline gelmesi amacıyla tasarlanmış bir Açık API'ler kümesidir. Bu yığın, minimal API'ler kümesini belirterek katmanlı yenilikçilik anlayışına vurgu yapıyor, beraberinde bu API'lerin tepesinde müşteriye özel uygulamalar oluşturmak için ekosistemin geri kalan kısmını cesaretlendiriyor. Aadhaar bir milyardan fazla Hint vatandaşına kimlik doğrulama hizmetleri sağlayan temel katmanlardan biri olarak hizmet veriyor. Buna ek olarak, servis sağlayıcılara Aadhaar detaylarını güvenli bir şekilde sunmak için dijital imzalar (eSign), birleştirilmiş çevrimiçi ödeme (unified online payment, UPI) ve elektronik onay katmanı (electronic consent layer, e-KYC) aracılığıyla kağıtsız işlemler sağlayan servisler var. İnanıyoruz ki, bu Açık API güdümlü girişim, dijital inovasyonu getirecek ve IndiaStack'ın arka planındaki tasarım prensipleri diğer bölgeler/ülkeler için bir değişim etkeni olarak kullanılabilecektir.

Nuance Mix, Dragon Speaking'in ve Siri'nin bu yeni piyasaya ilk sürülen halinin arka planındaki konuşmadan-metne teknolojisini yaratan şirketin ürünü bir doğal dil işleme framework'üdür. Bu framework ses aracılığıyla serbest şekilde kullanıcı etkileşimini mümkün kılan gramer kurallarının yaratılmasını destekliyor. Geliştirici, anlamak üzere kendini eğitebilen alana özel bir gramer tanımlıyor. Elde edilen sonuçlar, kullanıcının amaçları ve etkileşim kavramlarını tanımlayan kullanıcının girdisine yanıtlar niteliğinde ortaya çıkıyor. Başlangıçta onu eğitmek için kullanılanlara yakın tümceler ile sınırlı kalıyor, ama zaman geçtikçe birbirinden çok ayrı cümle yapılarından anlam saptamaya başlıyor. Hâlâ beta safhasında olmasına rağmen, ilk araştırmaların kesinliği çok ikna ediciyken nihai ürün, el kullanmadan gerçekleştirilen kullanıcı etkileşimiyle mobil, IoT, AR, VR ve etkileşimli sahalar da dahil olmak üzere, uygulama formlarının fayda görmesi açısından izlenecekler arasında yerini alıyor.

VR Başa Takılan Monitörlerin (HMD'lerin) birçoğunun yapımında altyapıyı oluşturan SDK olan ve Unity ile çalışan **OpenVR**, muhtemelen önem kazanmaya devam edecektir. ThoughtWorks'teki VR işlerinin çoğu OpenVR üzerine kuruludur çünkü diğer SDK'lardan farklı olarak herhangi bir HMD üstünde çalışabilir. Açık kaynak olmamasına rağmen, lisans üzerinden ücretsizdir. Oculus SDK lisanslamada daha kısıtlayıcıdır ve sadece Oculus cihazlarda çalışır. OSVR gerçekten açık kaynak olmakla birlikte henüz bu kadar benimsenmiş gibi gözükmüyor. Eğer bir VR uygulaması geliştireceksiniz ve mümkün olduğu kadar çok cihazı hedefliyorsanız — ve de o halde geliştirmek için Unity veya Unreal kullanmıyorsanız — OpenVR tam şu anda en somut ve pragmatik çözümdür.

PLATFORMLAR *devamı*

Tarantool veri bankasını ve önbelleği tek başlıkta birleştiren ve Lua dilinde uygulama mantığı yazmak için API'ler sağlayan açık kaynak bir NoSQL çözümüdür. Hem hafızada gömülü, hem de disk tabanlı motorlar desteklenir, kullanıcılar kendi kullanım durumlarına dayalı çoklu endeksler (HASH, TREE, RTREE, BITSET) yaratabilir. Verinin kendisi de MessagePack formatında saklanır ve istemci ve sunucu arasında iletişim kurmak için aynı protokolü kullanır. Tarantool kaydı önden tutmayı, işlemleri ve asenkron master-master kopyaları destekler. Eşzamanlı bağlantıları halletmek için tek yazıcı politikasını (single writer policy) ve kooperatif çok görevliliği tercih eden mimari kararı memnuniyetle karşılıyoruz

Makine zekası konusundaki heyecan dalgası zirvesine ulaştı ancak tüm boş sözler arasında «büyük veri» ile ilgili olarak yararlı framework ve araçlar keşfedilmeyi bekliyor. Bu tür araçlardan biri geliştiricilere Doğal Dil İşlemeyi (NLP - Natural Language Processing) kullanarak, karşılıklı konuşmalı arayüzler yaratma imkanı veren bir SaaS platformu olan **wit.ai**'dir. Wit, ya metin ya da konuşma girdileriyle çalışıyor, geliştiricilere karşılıklı konuşmanın amaçlarını yönetmekte yardımcı oluyor ve JavaScript kullanarak, geleneksel iş katmanı algoritması (kullanıcı arayüzü ile veri tabanı arasındaki veri alışverişini sağlayan algoritma) uygulanmasına imkan veriyor. Ticari ve ticari olmayan kullanımlar için ücretsiz olan sistem, açık uygulamaların oluşturulmasını teşvik ediyor. Wit'e, hizmet kalitesini artırmaları ve kendi analizlerinde kullanmaları için sizin verilerinizi kullanma izni vermeniz gerektiği konusunda bilginiz olsun. Bu yüzden kullanım şartlarını dikkatle okuyun. Bu alandaki iddia sahiplerinden biri de Microsoft Bot Framework, ancak bu yazı yayına hazırlandığı sırada sadece sınırlı ön izleme biçiminde sunuluyordu. Microsoft'un birçok şeyinde olduğu gibi Bot Framework'ün hızla gelişeceğini tahmin ettiğimiz için gözünüzün üzerinde olmasında fayda var.

Büyük ve karmaşık dijital uygulamalarını yapmak için **CMS'lerini bir platform olarak** kullanmaya teşebbüs eden birçok kuruluşun sorun yaşadığını görüyoruz. Kuruluşları buna yönlendiren, çoğu zaman, satıcıların duyarsız BT kuruluşlarını devre dışı bırakarak, değişiklikleri üretime doğrudan sürükleyip bırakma olanağı sağlama umutlarını körüklemesi oluyor. İçerik üreticilerine doğru araçların ve iş akışlarının sağlanmasına çok destek vermemize rağmen, iş mantığı karmaşık olan uygulamalar için, işlevselliğinizin tümünü CM'nin kendi içinde uygulamaya teşebbüs etmekten ziyade, sizin CMS'nize platformunuzun bir bileşeni olarak muamele etmenizi (çoğu zaman hibrit veya headless modunda) ve başka servislerle temiz bir işbirliği yapmanızı tavsiye etme eğilimindeyiz.

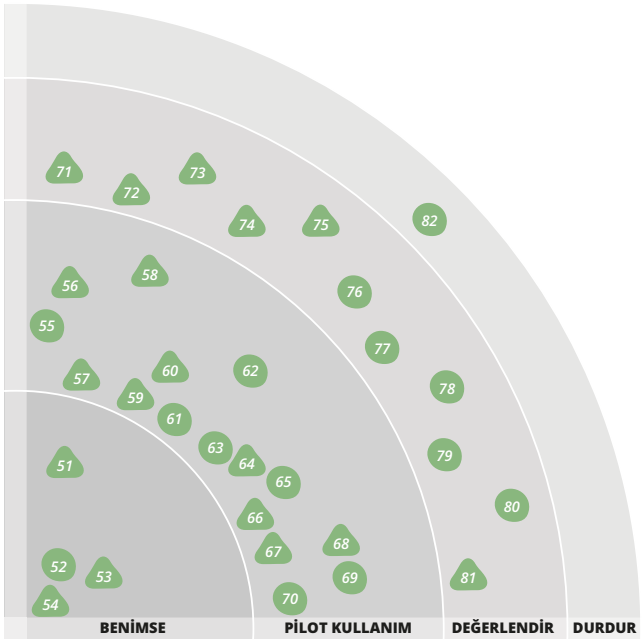
Sürekli yakınmalarımızdan biri de aracı yazılımın (middleware) içinde uygulamaya geçirilmiş olan iş zekası uygulamalarının, kritik uygulama mantığı çalıştırma hırsları eşliğinde transport yazılımına dönüşmekle sonuçlanması hakkındadır. Rekabetin çok yoğun olduğu API Gateway pazarındaki satıcılar, ürünlerinde fark yaratan özellikler eklemeye devam ediyorlar. Bu da **aşırı iddialı API Gateway** ürünleri olarak sonuçlanıyor ve bu ürünlerin işlevsellikleri — esasen bir ters proksi olan şeyin üzerine — test edilmesi ve kurulması zor tasarımları teşvik ediyor. API Gateway'ler bazı jenerik kaygılarla baş etmek için yararlı olabilir – örneğin kimlik kontrolü ve hız sınırlama – ancak örneğin veri dönüşümü veya kural işleme gibi alan zekası tekniklerinin (domain smarts), destekledikleri alanlarla yakın ilişki içinde çalışan üretim ekipleri tarafından kontrol edilebilecekleri uygulama ve hizmetlerin içinde kalmaları gerekir.

ARAÇLAR

Babel.js, yeni nesil JavaScript kodu yazmak için varsayılan derleyici haline geldi. Yeniden yapılandırılmış eklenti sistemi sayesinde, ekosistemi gerçekten kalkışa geçiyor. Geliştiricilerin, daha eski tarayıcılar için geriye dönük uyumu feda etmeden ve çok az yapılandırma ile tarayıcıda veya sunucuda çalışan ES6 (ve hatta ES7) kodu yazmalarına imkan sağlıyor. Farklı oluştur-test et sistemleri için birinci sınıf desteğe sahip, bu da onun mevcut herhangi bir iş akışıyla entegrasyonunun çok basit olmasını sağlıyor. ES6'nın (ve ES7'nin) benimsenmesinin ve yenilikçiliğinin ana sürücüsü haline gelmiş olan müthiş bir yazılım eseri.

Geliştirme sırasında Mikro Servisler, DevOps ve Canlı ortamda QA gibi modern teknikler ve mimari stillerini birleştirirken, geliştirme ekipleri giderek artan miktarda sofistike bir şekilde izlemeye ihtiyaç duyarlar. Artık basit bir şekilde disk kullanımı grafikleri ve CPU kullanımına bakmak yeterli değildir, birçok ekip uygulama ve iş alanına özel ölçümleri Graphite ve Kibana kullanarak topluyor. **Grafana** birçok kaynaktan gelen veriler için faydalı ve derli toplu kontrol panellerini yaratmayı kolay hale getiriyor. Özellikle faydalı bir özellik, farklı grafiklerin zaman ölçeklerini senkronize etmeye imkan tanıyor, bu da altta yatan veride korelasyonlar yakalamaya yardım ediyor. Eklenmiş olan modelleme sistemi çok şey vaat ediyor, muhtemelen benzer servislerin kümelerinin yönetilmesini aynı şekilde kolaylaştıracaktır. Güçlü yanlarına dayanarak, Grafana bizim bu kategoride varsayılan seçeneğimiz oldu.

Makine görüntüleri, modern kurulum ardışık düzenlerinin bir unsuru oldu. Bu görüntüleri yaratmak için de birçok araç ve teknik mevcuttur. Beraberinde kapsamlı özellik kümesi olduğu için ve olumlu deneyimlerimize dayanarak, alternatiflerine kıyasla **Packer**'ı tavsiye ediyoruz. Packer'ın yaptığı yaratıcılığı yapmak için geleneksel kod yazmaya çalışmayı da önermiyoruz.



Android uygulaması geliştirmek için test etme piramidinin en tepesinde, ekiplerimiz işlevsel test aracı olarak giderek artan bir şekilde **Espresso**yu kullanıyorlar. Küçük çekirdekli API'si karmakarışık uygulamaya geçirme detaylarını saklıyor, az ve öz test yapmaya yardımcı oluyor, daha hızlı ve güvenilir testler yaptırıyor. Espresso ile kullanıcı etkileşimlerinin simülasyonunu yapan otomatik UI testlerini, hem emülatörler hem de gerçek cihazlar üzerinde farklı Android versiyonlarının tamamında tek bir hedef uygulama içinde çalıştırabilirsiniz.

fastlane, iOS ve Android mobil uygulamalarında oluşturma, test etme, belge hazırlama ve kullanıma sunma gibi eylemlerde geçen sıkıcı aktivitelerin çoğunluğunu otomatikleştirmek için güvendiğimiz aracımızdır. Basit yapılandırma, bir araç yelpazesi ve çoklu ardışık düzenler bunu mobil cihazlar için Sürekli Dağıtım yapmada anahtar bir bileşen haline getiriyor.

BENİMSE
51. Babel
52. Consul
53. Grafana
54. Packer

PİLOT KULLANIM
55. Apache Kafka
56. Espresso
57. fastlane
58. Galen
59. HashiCorp Vault
60. JSONassert
61. Let's Encrypt
62. Load Impact
63. OWASP Dependency-Check
64. Pa11y
65. Serverspec
66. Talisman
67. Terraform
68. tmate
69. Webpack
70. Zipkin

DEĞERLENDİR
71. Android-x86
72. axios
73. Bottled Water
74. Clojure.spec
75. FBSnapshotTestcase
76. Grasp
77. LambdaCD
78. Pinpoint
79. Pitest
80. Repsheet
81. Scikit-learn

DURDUR
82. Kurulum ardışık düzeni olarak Jenkins

ARAÇLAR *devamı*

Bu planı test etmek ve farklı boyut ve çözünürlükteki cihazlara uyumlu web siteleri stilize etmek, tahmin edildiği gibi çeşitli form faktörleri baştan sona düşünüldüğünde yavaş ve sıklıkla manuel bir süreçtir. **Galen** basit bir dil sunup, Selenium üzerinde çalışarak bu sorunu kolaylaştırmaya yardım eder, bu sayede çeşitli ekran boyutlarında web sitenizin görünümü için size beklentilerinizi belirtebilme imkanı tanır. Galen, tipik kırılganlık açısından ve herhangi bir uçtan uca test yaklaşımı sırasında hız konusunda sorunlar yaşamasına rağmen; tasarım konularında erken geri besleme yapmakta fayda olduğunu gördük.

Sırları güvenli bir şekilde yönetmek, projelerde giderek önem kazanan bir konu haline dönüşüyor. Sırları ya da ortam değişkenlerini içeren bir dosya tutmayı öngören eski fikirler, özellikle uygulamaların sır katmanlarına erişiminin gerektiği durumlara sahip mikro servis ya da mikro konteyner ortamları gibi çoklu uygulamalara sahip ortamlarda yönetmesi zor bir yönteme dönüşüyor. **HashiCorp Vault**, birleşik bir arayüz üzerinden sırlara güvenli erişim için mekanizmalar sunarak bu sorunu çözmeye çalışıyor. Şifreleme ve bilinen araçlar için otomatik olarak sırlar üretme gibi hayatı kolaylaştıran bazı özelliklere sahip. Sırları depolamak ve güncellemek, bir komut satırı aracına ve ekibin oldukça disiplinli olmasına bağlı olduğu için biraz külfetlidir.

Giderek daha fazla proje, bilgiyi JSON formatlı olarak yayıyor ve tüketiyor. JSON için Java'da testler yazmak çok uğraş gerektirebilir. JSON ile uğraşarak daha küçük testler yazmaya yardımcı olan küçük bir kütüphane olan **JSONassert**, savlar basitleştirir ve daha iyi hata mesajları sağlar.

Pa11y, komut satırından çalışabilen ve bir inşa ardışık düzenine iliştirilebilen otomatik bir erişilebilirlik test aracıdır. Ekiplerimiz, çok dinamik bir site üzerinde, önce bir statik HTML versiyonu yaratıp daha sonra erişilebilirlik testini bunun üzerinde deneyerek Pa11y'yi başarıyla kullandılar. Birçok sistem için - özellikle devlet web siteleri için - erişilebilirlik testi şarttır ve Pa11y bunu çok daha kolay hale getiriyor.

Vault gibi araçların olgunlaşmasıyla sırları veri havuzunda saklamanın mazereti kalmadı çünkü bu durum çoğu zaman önemli sistemlerin yumuşak karnı haline geliyordu. Daha önce Gitrob gibi veri havuzu tarama araçlarına değinmiştik ancak şu anda öntanımlı örüntülere denk gelen sırlar için kod güncellemelerini tarayan git için piyasaya sürme öncesi bir ön yoklama kancası olan (ThoughtWork'ün yarattığı) **Talisman** gibi proaktif araçları ön plana çıkarıyoruz.

Terraform ile bulut altyapısını bildirimsel tanımlar yazarak yönetebilirsiniz. Terraform tarafından başlatılan sunucuların konfigürasyonları, genellikle Puppet, Chef veya Ansible gibi araçlara bırakılır. Terraform'u, dosyalarının sözdizimi oldukça okunaklı olduğu için ve çok sayıda bulut sunucuyu desteklerken bu sunucular arasında yapay bir soyutlama yapmaya çalışmadığı için beğeniyoruz. Yaklaşık iki yıl önce ilk defa ve daha temkinli bir şekilde Terraform'a değinmemizin ardından sürekli gelişme gösterdi ve bizim projelerimizde değerini kanıtlayan stabil bir ürüne dönüştü. Durum dosyası yönetimi artık Terraform'un «uzak durum yönetim altyapısı» olarak adlandırdığı yöntem kullanılarak bir kenara bırakılabilir. Bu amaca yönelik olarak Consul'u başarıyla kullandık.

İkili programlama (pair programming) bizim için temel bir tekniktir ve üyeleri çeşitli mekanlara dağılmış ekipleri giderek daha fazla görmeye başladığımız için birçok araçla uzaktan ikili programlamayı desteklemeyi denedik. ScreenHero'yu kesinlikle beğendik ama geleceği konusunda kaygılıyız. Grafiksel bir IDE'ye dayanmayan ekiplerin ikili programlama için **tmate** kullanmasının harika bir çözüm olduğu ortaya çıktı. Popüler tmux aracının bir dallanması olan tmate'in kurulumu', uzaktan ikili programlama için Tmux'a oranla çok kolaydır. Grafik ekranı paylaşımı çözümlerine kıyasla, bant genişliği ve kaynak gereksinimleri daha mütevazidir ve bulanık ekran sorununu hiç yaşamadığı çok açıktır. Ekipler kendi sunucularını da kurabilir ve böylece çözümün gizlilik ve bütünlüğü üzerinde tam kontrolü ellerinde tutabilirler.

Android-x86, x86 platformlarının kullanımına sunulmuş olan, Android açık kaynak projesinin bir port'udur. Proje x86'ya destek veren topluluktan kaynaklanarak çeşitli yamaları barındırma suretiyle başladı, ama daha sonrasında farklı x86 platformları için destek sağlamak amacıyla kendi kod tabanını oluşturdu. CI sunucularımızdaki hermetik UI testler için emülatörler yerine Android-x86 kullanımıyla kayda değer zaman tasarrufları sağlandığını gördük. Ancak özel bir cihaz çözünürlüğünü hedefleyen —düşük bellek, bant genişliği ve pil gücüyle simülasyon yaparken—, UI'ye özel testler için emülatörlerle çalışmak daha iyidir.

Ekiplerimiz **axios** ile başarı kaydetti. Bu, "Fetch"ten daha iyi" diye tanımladıkları vaatler üzerine kurulu, JavaScript dilinde bir HTTP istemcisidir. GitHub üzerinde birçok onaylanma ve aktiviteye sahip olan proje, bizden geçer not aldı.

ARAÇLAR *devamı*

Akım konusundaki veri mimarilerine ve besledikleri akıntı yönündeki veri göllerine artan ilgiyle beraber, işlem tabanlı veri ambarlarını akış-işleme sistemlerine bağlayan "değişen veri yakalama" araçlarına güven artışı gördük. **Bottled Water** ; bu alana, memnuniyetle karşılanan bir katkıdır; PostgreSQL'in kaydı önden tutma sistemi, değişiklikleri Kafka olaylarına dönüştürüyor. Ama bu yaklaşımın bir dezavantajı, olay odaklı mimarinin temeli olarak tavsiye ettiğimiz yüksek seviyeli iş olaylarından daha çok düşük seviyeli veri tabanı olaylarına bağlı olmanızdır.

Geliştiriciler arasında sürüp giden tartışma konularından biri, dillerin tip sistemleridir: Bunun ne kadarı tam olarak doğrudur? JVM üzerinde dinamik olarak yazılan işlevsel Lisp olan Clojure, bu tartışmaya sınırları belirsizleştiren yeni bir başlık ekledi. **Clojure.spec**; Clojure'ün içinde oluşturulmuş yeni bir kolaylıktır; geliştiricilerin, örneğin izin verilen değer aralıklarını kontrol etmek gibi, veri yapıları çevresindeki yazım ve diğer doğrulama kriterlerini uygulayabilmesine imkan tanır. Bir kere yerleşik hale geldiklerinde, Clojure bu belirlemeleri yığınla fayda sağlamak için kullanıyor: gerçekleştirilen testler, validasyon, veri yapılarının içinden veriyi çekip çıkarma ve diğerleri. Clojure.spec her yerde değil ama geliştiricilerin onlara ihtiyacı olan yerde veri tipleri ve değer aralıklarının faydalarını görmek için geleceği parlak bir yoldur.

iOS uygulamalarının görsel kısmını test etmek zahmetli, yavaş ve güvenilmez olabilir. Bu yüzden **FBSnapshotTestcase**'i araç kitimize dahil etmekten memnuniyet duyuyoruz. Kullanıcı arayüzü bileşenlerinin anlık görüntülerinin alınması, depolanması ve farkını alma işlemlerini otomatikleştirerek, arayüzlerinizi piksel mükemmelliğinde tutmanızı sağlıyor. Birim testi olarak çalıştığı için (simülatörün içinde) işlevsel test yaklaşımlarından daha hızlı ve güvenilirdir.

Scikit-learn, Python'da yazılmış ve giderek popülerliği artan bir makine öğrenimi kütüphanesidir. Kümeleme, sınıflandırma, regresyon, boyut azaltma (değişken sayısını azaltma) gibi makine öğrenimi modellerinden oluşan güçlü bir set ve model seçimi, model değerlendirme ve veri hazırlama gibi yan görevler için zengin bir işlevler grubu sağlar. Basit olması, çeşitli bağlamlarda yeniden kullanılabilmesi ve açıklaması iyi yapılmış olması amacıyla tasarlandığı için bu aracın uzman olmayanlar için bile makine öğrenimi dünyasını keşfederken kullanılabileceğini düşünüyoruz.

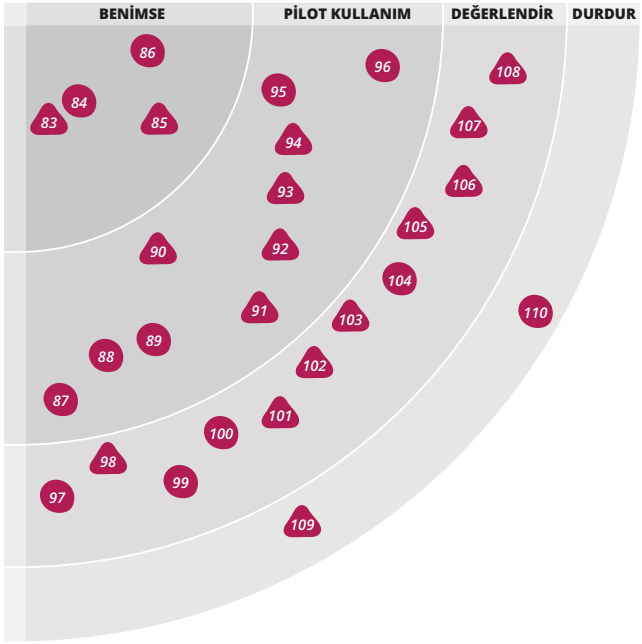
DİLLER ve FRAMEWORK'LER

Tek sayfalık bir uygulama (SPA) yapmanız gerekiyorsa ve bunun için bir framework seçmeye çalışıyorsanız **Ember.js** lider bir seçenek olarak ön plana çıktı. Ekiplerimiz; Ember'i, AngularJS gibi diğer framework'lere kıyasla çok daha az sürpriz içermesi nedeniyle, çok verimli geliştirici deneyimi açısından övüyor. Ember CLI yapım araçları, JavaScript yapım araçları fırtınası içinde bir limandır ve Ember çekirdek ekibi ve topluluğu son derece aktif ve duyarlıdır.

Tek sayfalık JavaScript uygulamalarının artan karmaşıklığı içinde, müşteri tarafı durum yönetiminin daha öngörülebilir hale getirilmesi ihtiyacının giderek daha baskılı hale geldiğini gördük. Durum güncelleme konusunda üç ilkesi olan **Redux**; bizim uygulamaya geçirdiğimiz çok sayıda projede, paha biçilmez değerde olduğunu kanıtladı. Redux ve idyomatik Redux ile çalışmaya başlarken; eğitimler, yeni ve deneyimsiz kullanıcılar için iyi bir başlangıç noktasıdır. Minimal kütüphane tasarımı zengin bir araç seti ortaya çıkmasını sağladı ve size örnekler, aracı yazılımlar ve yardımcı kütüphaneler için redux-ekosistemi-linkleri projesini denemenizi tavsiye ederiz. Ayrıca test edilebilmesi hikayesini de özellikle seviyoruz: Sevk etme eylemleri (dispatching actions), durum geçişleri (state transitions) ve yürütme birbirinden ayrı olarak ünite ünite ve asgari miktarda taklitle test edilebilirler.

Elixir programlama diline yönelik ilgi artmaya devam ediyor. Ciddi projelerde kullanıldığını giderek daha çok görüyoruz ve Actor modelinin sağlam ve çok hızlı olduğunu düşünen geliştiricilerden giderek daha fazla geri beslemeler almaya devam ediyoruz. Erlang sanal makinenin üzerine yapılandırılan Elixir, eş zamanlı ve hata toleranslı sistemler yaratmak konusunda umut vaat ediyor. Elixir, UNIX komut satırında yapabildiğiniz gibi fonksiyon ardışık düzenleri geliştirilmesine olanak tanıyan Pipe operatörü gibi farklı özelliklere sahip. Paylaşılan bayt kodu, Elixir'in Erlang ile birlikte çalışmasına ve mevcut kütüphaneleri kullanırken, Mix oluşturma aracı, lex interaktif kabuk ve ExUnit birim test framework'u gibi araçların desteklenmesine olanak tanıyor.

Enzyme'in React.js uygulamaları için sağladığı hızlı, bileşen seviyesinde UI test etme özelliğinden memnun kaldık.



Diğer birçok anlık tabanlı test framework'lerinden farklı olarak, Enzyme daha hızlı ve daha detaylandırılmış test etme özelliğiyle sonuçlanan, cihaz üzerinde yürütmeden test etmeye imkan tanır. Bu, React uygulamalarında yapmamız gerektiğini bulduğumuz işlevsel test etme toplamımızı, yüklü miktarda azaltma yeteneğimize katkı sağlayan bir faktördür.

Değişmezlik, fonksiyonel programlama paradigmasında sıklıkla vurgulanıyor ve birçok dilde bir kere yaratıldıktan sonra değiştirilemeyen, değişmez objeler yaratılması özelliği bulunuyor. **Immutable.js** modern JavaScript sanal makineler üzerinde çok verimli olan, kalıcı olarak değiştirilemez veri yapıları sağlayan JavaScript için geliştirilmiş bir kütüphane. Ancak, Immutable.js objeleri normal objeler değildir. Dolayısıyla değiştirilemez objelerden JavaScript objelerine referanslar verilmesinden kaçınılmalıdır. Giderek daha fazla ekip bu kütüphaneyi mutasyonun takibi ve üretimde durumun korunması için kullanıyor. Bu kütüphanenin özellikle Facebook yığınının geri kalanıyla birleştirildiği zaman, geliştiriciler için incelenmesi gereken bir kütüphane olduğunu düşünüyoruz.

BENİMSE
83. Ember.js
84. React.js
85. Redux
86. Spring Boot

PİLOT KULLANIM
87. Butterknife
88. Dagger
89. Dapper
90. Elixir
91. Enzyme
92. Immutable.js
93. Phoenix
94. Quick and Nimble
95. React Native
96. Robolectric

DEĞERLENDİR
97. Aurelia
98. ECMAScript 2017
99. Elm
100. GraphQL
101. JuMP
102. Physical Web
103. Rapidoid
104. Recharts
105. ReSwift
106. Three.js
107. Vue.js
108. WebRTC

DURDUR
109. AngularJS
110. JSPatch

DİLLER ve FRAMEWORK'LER *devamı*

ThoughtWorks ekibimizden bazıları Elixir dilinde yazılmış sunucu-taraflı web MVC framework'ü olan Phoenix ile çok olumlu deneyimler yaşadılar. Akış özellikli ve kullanımı kolay olmasına ek olarak **Phoenix**, Elixir'in son derece hızlı olmasının avantajını kullanıyor. Bazı geliştiriciler için Phoenix, Ruby ve Rails'i ilk keşfettikleri zaman deneyimledikleri keyfi çağrıştırıyor. Phoenix için kütüphaneler ekosistemi, daha olgunlaşmış framework'ler için olduğu kadar kapsamlı olmamasına rağmen, Elixir'in sürmekte olan başarısının ve Elixir'e verilen desteğin artmasının faydasını görüyor olsa gerek.

iOs ekibimizin çoğunluğu şu anda **Quick and Nimble** eşleştirmesini, kendi birim testleri için kullanıyor. Davranış güdümlü geliştirme (BDD, behavior-driven development) testi araçlarının RSpec ailesinde, Swift ve Objective-C dilleriyle geliştirme süreci boyunca çok okunabilir testler (tarif blokları ile beraber) sağlıyor ve asenkron test etmek için iyi destek veriyor.

ECMAScript 2017—ES7 ile karışmasın (diğer adı ECMAScript 2016)— dile başka birçok kayda değer iyileştirme getiriyor. Tarayıcıların bu standardı 2017 yazı itibariyle tamamıyla uygulamaya geçirmesi bekleniyor, ama Babel JavaScript derleyici bugün daha şimdiden çok sayıda özelliği destekliyor. Eğer JavaScript'i kapsamlı bir şekilde kullanırsanız ve kod bankanız aktif geliştirme sürecindeyse, Babel'i yapım ardışık düzeninize eklemenizi ve desteklenen özellikleri kullanmaya başlamanızı tavsiye ediyoruz.

JuMP, Julia'da matematiksel optimizasyonlar yapmak için alana-özel bir dildir. MathProgBase diye adlandırılan ortak bir API tanımlayan JuMP, kullanıcılarının da Julia'da çözümleyiciden bağımsız kod yazmasını mümkün kılar. Desteklenen güncel çözücüler arasında Artelys Knitro, Bonmin, Cbc, Clp, Couenne, CPLEX, ECOS, FICO Xpress, GLPK, Gurobi, Ipopt, MOSEK, NLopt ve SCS vardır. Diğer bir faydası da türev almak için ters modda otomatik türev alma tekniğinin uygulamaya geçirilmesidir ki böylelikle kullanıcılar sin, cos, log ve sqrt gibi standart işlemcilerle sınırlı kalmadıkları gibi, Julia'da kendilerine ait olarak tasarladıkları objektif fonksiyonlarını da uygulamaya geçirebilirler.

Google'ın yarattığı **Physical Web** standardı ilgimizi çekti. Physical Web'in temelindeki fikir basittir — gösterim takip kodu bir URL yayınlar — ama olanaklar geniştir. Temel olarak bu, fiziki dünyayı not etmek için bir yöntemdir, nesne ve lokasyonları dijital aleme bağlar. Güncel taşıma mekanizması Bluetooth LE üzerinden Eddystone URL'lerdir ve örnek istemciler mevcuttur. Rastgele keşfedilmiş linkleri takip ederken belirgin güvenlik meseleleri olmasına rağmen, en çok, gerektiğinde URL'leri filtreleyebileceğiniz ya da proksiye kaydedebileceğiniz özelleştirilmiş istemcilerin kullanım durumları ile ilgileniyoruz.

Rapidoid, Java NIO üzerinde sıfırdan hayata geçirilmiş, hızlı bir düşük seviyeli HTTP sunucu içeren, web framework modüllerinin bir derlemesidir. Öbek dışı girdi/çıktı ara belleklerinin, nesne havuzlarının ve iş parçacığında yerel veri yapılarının zekice kullanımı Rapidoid'e Netty gibi diğer NIO tabanlı sunuculara göre avantaj sağlar. Oldukça yeni bir proje olarak Rapidoid'in hâlâ dahili önbellek ve SSL desteği gibi uygulamaya geçirmesi gereken birkaç özellik vardır; biz güncellemeler için yol haritasını kontrol etmenizi öneririz.

Redux paradigmasının Swift alanına **ReSwift** biçiminde giriş yapmasından heyecan duyuyoruz. Kod tabanlarının basit ve okunabilir olmasında; durum ve durum değişimleri, merkezi bir yerde ve kendine özgü ortak bir dille yönetildiği sürece, gerçek faydalar olduğunu tespit ettik. Bu "çevrimdışı öncelikli" uygulamalar oluşturmaya da yardım eder.

Yeni başlık akınını saran heyecan dalgasına rağmen; tarayıcıda ve özellikle mobilde anlamlı olan birçok VR ve AR senaryosunun bulunduğuna inanıyoruz. Bu trende rağmen, güçlü bir JavaScript görselleştirmesi ve 3 boyutlu canlandırma framework'ü olan **Three.js** kullanımında olumlu bir yan gördük. Three.js'nin dayandığı WebGL'ye desteğin büyümesi, benimsenme sayısının artmasına ve bu açık kaynak projesini destekleyen canlı topluluğun genişlemesine destek verdi.

Ön kullanıcı JavaScript framework'lerinin sürekli değişen dünyasında **Vue.js**, AngularJS'nin hafif bir alternatifi olarak çok mesafe kat etti. Çok esnek - ve daha az kuralcı - bir kütüphane olması ve modülerlik, bileşenler ve reaktif veri akışı gibi kavramlar etrafında etkileşimli web arayüzleri oluşturmak için bir dizi araç sunmak üzere tasarlandı. Çok düşük bir öğrenme bariyeri olması, tecrübesiz geliştiriciler ve yeni başlayanlar için çok ilgi çekici olmasını sağlıyor. Vue.js'nin kendisi tam gelişmiş bir framework değildir - sadece izleme katmanına odaklanır. Bu nedenle diğer kütüphaneler veya mevcut projelerle entegre etmek çok kolaydır.

AR/VR'nin bir ortak çalışma ve iletişim ortamı olarak geniş bir kabul görmesi modern ve hemen kullanılabilir bir video akıtma (streaming) platformunun bulunmasını gerektirir. **WebRTC**, yaygın bir şekilde kullanılan web teknolojilerinin içinde video akıtmayı mümkün kılan tarayıcılar arasında gerçek zamanlı iletişim için yükselen bir standarttır. Bu standardı destekleyen tarayıcılar yelpazesi giderek genişliyor ancak Microsoft ve Apple, kendilerine ait tarayıcılarda WebRTC'yi benimsemekte yavaş davranıyorlar. İvmenin artmaya devam etmesi halinde, web üzerinde AR/VR ortak çalışmasının gelecekteki temelini oluşturabilir.

AngularJS tek sayfalı JavaScript uygulamaları dünyasında devrim yapılmasına yardımcı oldu ve biz de onunla yıllardır birçok başarılı proje gerçekleştirdik. Ancak artık, yeni bir projeye başlacak olan ekiplere tavsiye etmiyoruz (v1).

Onun yerine ekiplere daha hızlı bir başlangıç ve bakımı daha kolay bir kod yapısı sunduğunu gördüğümüz Ember ve React'ın özellikle Redux birlikte kullanılmasını tercih ediyoruz

---

ThoughtWorks, yazılım danışmanlığı, üretimi ve ürünleri konusunda uzmanlaşmış tutkulu ve hedef sahibi bireylerin oluşturduğu bir topluluk ve yazılım şirketidir. Müşterilerimize yaşadıkları en büyük zorluk ve mücadelelerde yardımcı olmak üzere alışılmışın dışında düşünürken IT sektöründe bir devrim yapmanın ve olumlu bir sosyal değişiklik yaratmanın yolunu arıyoruz.

Harika olmayı çok isteyen yazılım ekipleri için öncü araçlar yapıyoruz. Ürünlerimiz, kuruluşların sürekli gelişmelerine ve en kritik ihtiyaçları için kaliteli yazılımlar üretmelerine yardımcı oluyor. 23 Yıl önce kurulan ThoughtWorks, Chicago'daki küçük bir şirketten 14 ülkedeki (Avustralya, Brezilya, Sili, Çin, Ekvador, Almanya, Hindistan, Singapur, Italya, Ispanya, Güney Afrika, Türkiye, Birleşik Krallık ve ABD) 40 ofiste 4000'den fazla çalışanı olan bir şirkete dönüşmüştür.

ThoughtWorks®