ThoughtWorks®

TECHNOLOGY RADAR *VOL.16*

Geleceği şekillendiren teknoloji ve trendler üzerine görüşlerimiz

thoughtworks.com/radar

#TWTechRadar

KATKIDA BULUNANLAR

Teknoloji Radarı aşağıdaki ThoughtWorks Teknoloji Danışma Kurulu tarafından hazırlanmıştır:

Rebecca Parsons (CTO) | Martin Fowler (Chief Scientist) | Badri Janakiraman | Bharani Subramaniam | Camilla Crispim
Erik Doernenburg | Evan Bottcher | Fausto de la Torre | Hao Xu | Ian Cartwright
James Lewis | Jonny LeRoy | Marco Valtas | Mike Mason | Neal Ford
Rachel Laycock | Scott Shaw | Srihari Srinivasan | Zhamak Dehghani

YENİLİKLER

Bu sayıda öne çıkan konular şunlar:

SOHBET TABANLI KULLANICI ARAYÜZÜ VE DOĞAL DİL İŞLEME

videoyu izle (thght.works/ConUI)

Uygulamalarla etkileşimin yeni bir yöntemi olan konuşma, Siri, Cortana ve Allo gibi araçlar üzerinde ekosistemde fırtınalar estirdi ve Amazon Echo ve Google Home gibi cihazlar üzerinden evlere girdi.

Sohbet tabanlı ve doğal dil kullanıcı arayüzleri geliştirmek, yeni zorluklar ortaya çıkarmakla birlikte çok açık yararlar sağlıyor. Echo'nun arkasındaki ekip bilerek ekranı aradan çıkararak kendilerini birçok insan-makine etkileşimini yeniden düşünmeye zorladı.

Sohbet tabanlı trend sadece sesle sınırlı değil; mesajlaşma uygulamaları büyüyerek hem telefonlara hem de işyerlerine hakim olurken diğer insanlarla konuşmaların akıllı sohbet robotları ile desteklendiğini görüyoruz. Bu platformlar geliştikçe konuşmaların bağlam ve amacını anlamayı öğrenerek etkileşimleri gerçek hayata daha uygun ve böylece daha cazip hale getirecekler.

Pazarda ve ana akım medyada ilginin patlaması, bu yeni kişisel dışkorteks etkileşim moduna geliştiricilerin ilgisinin de paralel olarak artmasına yol açtı.

SERVİS OLARAK ZEKA

videoyu izle (thght.works/IntSer)

Son dönemde, bizim servis olarak zeka ismini verdiğimiz bir platformlar ailesi sahneye çıktı. Bu platformlar, ses işlemeden doğal dil anlama, resim tanıma ve derin öğrenmeye kadar uzanan çok çeşitli ve şaşırtıcı derecede güçlü araçları kapsıyor. Birkaç yıl önce yüksek maliyetli kaynakları tüketecek olan olanaklar, artık açık kaynak veya SaaS platformları olarak ortaya çıkıyor.

«Bulut savaşlarının» depolama ve bilgi işlemden, bilişsel kapasitelere kaydığı görülüyor ve Kubernetes ve Mesos gibi daha önce fark yaratan araçları açık kaynak hale getirmeye razı olunması da bu durumu kanıtlıyor.

Sektördeki bütün büyük oyuncular bu alanda ürünlerini sunarken ilginç niş oyuncular da değerlendirmeye değer. Bu hizmetlerin etik ve gizlilikle ilgili getirileri konusunda bazı çekincelerimizi korumakla birlikte bu güçlü araçları yeni şekillerde kullanmanın büyük faydalar vaat ettiğini düşünüyoruz. Müşterilerimiz, olağan kavrayış ile kendi işleriyle ilgili zekayı birleştirerek hangi yeni ufuklara ulaşabileceklerini araştırmaya başlamış durumdalar.

FARK YARATAN YENİ UNSUR OLARAK GELİŞTİRİCİ DENEYİMİ

videoyu izle (thght.works/DevExp)

Kullanıcı deneyimi tasarımı, teknoloji ürünleri şirketleri için uzun yıllardır fark yaratan önemli bir unsur oldu. Şu anda geliştiriciye yönelik araç ve ürünlerin hızla çoğalması ve mühendislik yeteneklerinin nadir olması, benzer bir odaklanmayı geliştirici deneyimine yönlendiriyor.

Kuruluşlar giderek artan bir şekilde, bulutta sunulan hizmetleri, mühendislikle ilgili engelleri ne ölçüde ortadan kaldırdığına, API'leri ne ölçüde ürün olarak değerlendirdiğine ve mühendislik verimliliğine odaklanmış ekipleri ne ölçüde hızlandırdığına göre değerlendiriyor. ThoughtWorks olarak verimli mühendislik uygulamalarına her zaman saplantımız oldu ve geliştiricilerin hayatını kolaylaştıran araç ve ürünleri destekledik. Bu yüzden sektörün bu anlayışı benimsemeye başladığını görmek bize heyecan veriyor.

En önemli teknikler arasında şunlar bulunuyor: dahili altyapıyı, başka kuruluşların sunduğu ürünlerle rekabet edebilecek kadar cazip hale getirilmesi gereken bir ürün olarak ele almak, self servise odaklanma, ürettiğiniz API'lerin geliştirici ergonomisini anlamak, "legacy sistemlerinin çevrelenmesi" ve hizmetlerinizi kullanan geliştiricilere yönelik olarak sürekli empatik araştırmalar yapmak.

PLATFORMLARIN YÜKSELİŞİ

videoyu izle (thght.works/RiseOTP)

Radar'ın konuları gözlemler ve güvenlik kontrolü süreçlerindeki konuşmalardan ortaya çıkıyor; son dönemde Radar'ı derlerken Platformlar çeyrek daire-sindeki yeni girişlerin sayısı dikkatimizi çekti. Bunun, yazılım geliştirme ekosisteminde daha geniş çaplı bir trendin göstergesi olduğunu düşünüyoruz.

Silikon Vadisi'nin önde gelen şirketleri, uygun bir platform kurmanın ne kadar önemli yararlar sağlayabi-leceğini gözler önüne serdiler. Bunların başarısı kısmen kapsülleme ve yetenekler konusunda kullanışlı bir seviye bulmaktan kaynaklanıyor. Giderek artan bir şekilde «platform düşünme biçimi», Radar'da dikkat çekilen doğal dil gibi ileri olanaklardan Amazon gibi altyapı plat-formlarına kadar ekosistemin her yerinde ortaya çıkıyor.

Şirketler, ürünlerden esinlenen API'ler üzerinden seçkin olanakları açıklarken platformları düşünmeye başlıyorlar. Geliştirme ekipleri, entegrasyon ve geliştirici deney-imini iyileştirmek için platformlar kurmayı daha fazla düşünüyorlar. Sektör nihayet, paketleme, kullanışlılık ve yararlılık açısından makul bir bileşimi benimsemiş gibi görünüyor.

Bizim hoşumuza giden bir tanım platformların bir self servis API ortaya çıkarması ve bir ekip ortamında kurul-ması ve öngörülmesi gerekmesidir - bu da, hoş bir şekil-de, yeni ortaya çıkan bir tema olan, fark yaratan yeni bir unsur olarak Geliştirici deneyimi konusuyla kesişir. Yakın gelecekte platformların hem tanımında hem de olanak-larında daha da fazla ilerlemeler görmeyi umuyoruz.

YAYGINLAŞAN PYTHON

videoyu izle (thght.works/PerPyt)

Python sürekli ilginç yerlerde ortaya çıkan bir dildir. Bir genel programlama dili olarak kullanımının kolay olmasının yanı sıra matematiksel ve bilimsel bilgi işlemdeki sağlam temelleri, tarihsel olarak akademi ve araştırma dünyası tarafından kökleşmiş bir şekilde benimsenmesine yol açtı. Daha yakın dönemde, AI'nin metalaşması ve uygulamalarıyla ilgili sektör trendlerinin yanı sıra Python 3'ün, olgunluk döneminin gelmesi Python sürüsüne yeni toplulukların katılmasını sağladı.

Radar'ın bu sayısında ekosistemin güçlenmesine katkıda bulunan birkaç Python kütüphanesi yer alıyor: makine öğrenmesi alanında Scikit-learn; akıllı veri akışı grafikleri için TensorFlow, Keras, ve Airflow ve sohbet farkındalığı olan API'lere güç vermek için doğal dil işlemeyi kullanan spaCy. Python'un giderek kuruluş içindeki bilim insanları ve mühendisler arasındaki mesafeyi kapattığını ve tercih ettikleri araçlarla ilgili geçmişteki önyargıları yumuşattığını görüyoruz.

Mikroservisler ve Konteynırlar gibi mimari yaklaşımlar, Python'un üretim ortamlarında uygulanmasını kolaylaştırdı. Mühendisler artık bilim insanlarının dil bağımsız ve teknoloji bağımsız API'ler üzerinden yarattıkları özelleştirilmiş Python kodunu kurup entegre edebiliyorlar. Bu akıcılık, fiilen kullanılan R gibi özelleştirilmiş dillerin üretim ortamlarına tercüme edilmesi yönteminin aksine araştırmacı ve mühendisler arasında tutarlı bir ekosistem kurulması yönünde büyük bir adımdır.

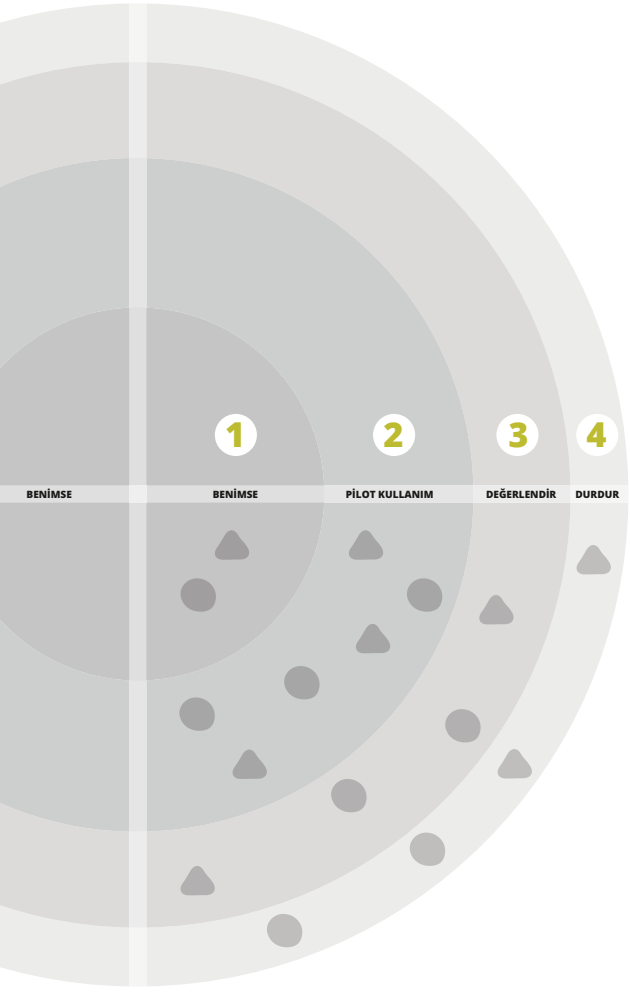
RADAR HAKKINDA

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. Misyonumuz, 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 düzenli toplantılar yaparak, ThoughtWorks'ün küresel teknoloji stratejisini ve sektörümüzü ciddi olarak etkileyen teknoloji trendlerini tartışıyor.

Radar, Teknoloji Danışma Kurulunun bu tartışmalarını CTO'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 framework'ler bazında gruplandırıyor. Radarın birden fazla çeyrek dairede yer alabilecek maddeleri için en uygun göründükleri çeyrek daireyi seçiyoruz. Bu maddeleri ayrıca şu anda onlar hakkındaki tavrımızı yansıtan dört halka halinde gruplandırıyoruz.

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



BİR BAKIŞTA RADAR

1 BENİMSE

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

2 PİLOT KULLANIM

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

3 DEĞERLENDİR

Kurumunuzu nasıl etkileyeceğini anlamak amacıyla takip etmekte fayda var..

4 DURDUR

Temkinli adım atın.

YENİ VEYA DEĞİŞTİRİLMİŞ

DEĞİŞİKLİK YOK

Yeni veya son radardan beri büyük değişim geçiren maddeler üçgen olarak, aynı kalan maddeler ise yuvarlak olarak verilir.

Radarımız ileriye dönüktür. Yeni maddelere yer açmak için son zamanlarda hareket olmayan maddeleri kaldırıyoruz. Bu onların değersiz olduğu anlamına gelmiyor.

RADAR

TEKNİKLER

BENİMSE
1. Kod olarak ardışık düzenler

PİLOT KULLANIM
2. Ürün olarak API
3. Güvenlik bilgilerinin kaynak kodundan ayrıştırılması *YENİ*
4. AB'de PII verilerinin barındırılması
5. Legacy sistemlerinin çevrelenmesi *YENİ*
6. Hafif Tasarımlı Mimari Karar Kayıtları
7. İleri Web Uygulamaları *YENİ*
8. InVision ve Sketch ile prototip oluşturmak *YENİ*
9. Sunucusuz mimari

DEĞERLENDİR
10. İstemci yönlendirmeli sorgu
11. Konteynır güvenlik taraması
12. Sohbet farkındalığı olan API'ler *YENİ*
13. Kademeli Gizlilik
14. Micro ön yüz
15. Platform mühendisliği ürün takımları *YENİ*
16. Sosyal kod analizi *YENİ*
17. Oyunun ötesinde Sanal Gerçeklik

DURDUR
18. Tüm ekipler için tek bir CI kurulumu
19. Anemik REST
20. Büyük Veri özentisi
21. CI tiyatrosu *YENİ*
22. Kurumsal çapta entegrasyon test ortamları *YENİ*
23. Spesifikasyon üzerinden otomatik kod üretimi *YENİ*

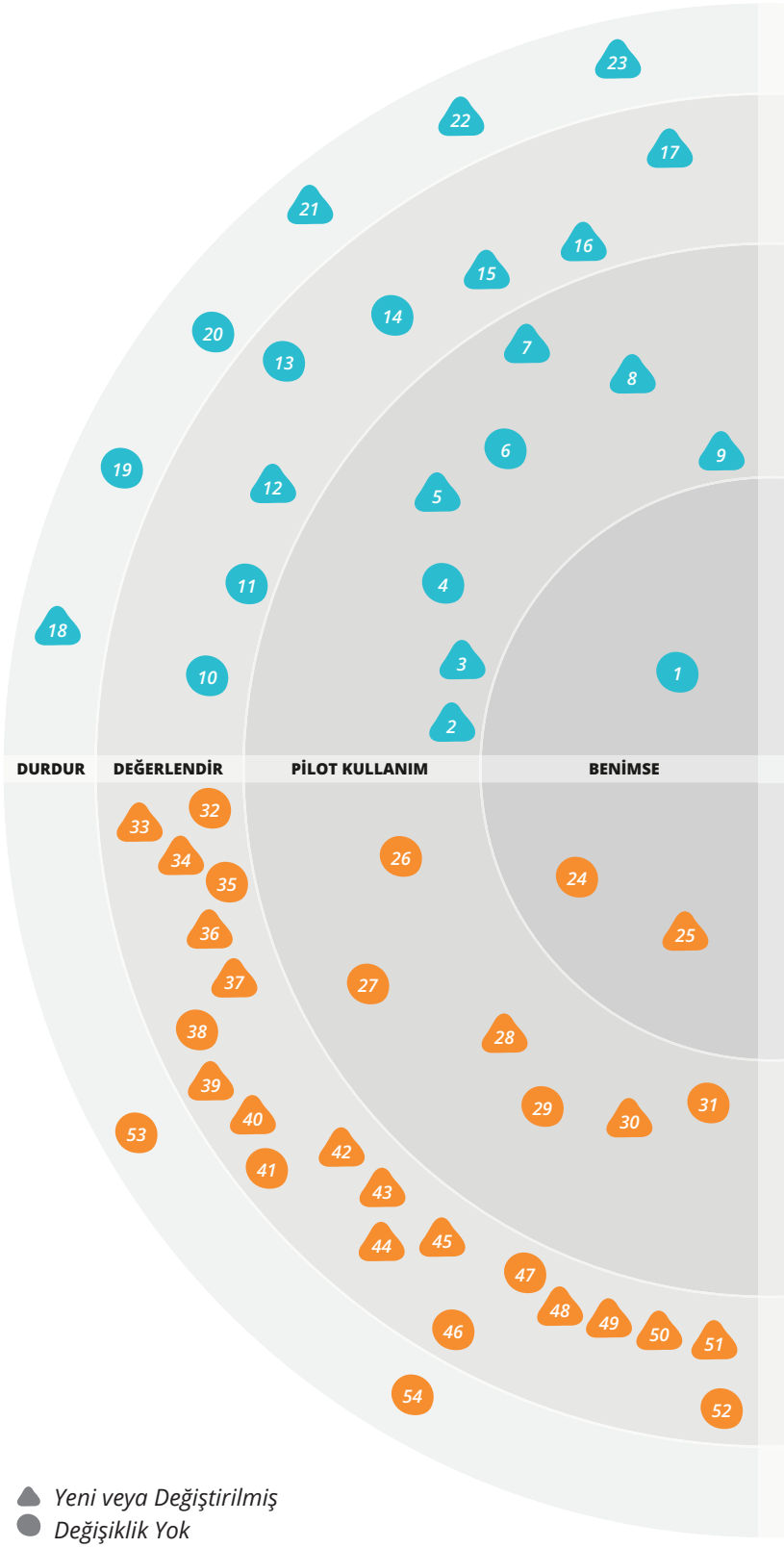
PLATFORMLAR

BENİMSE
24. HSTS
25. Linux güvenlik modülleri

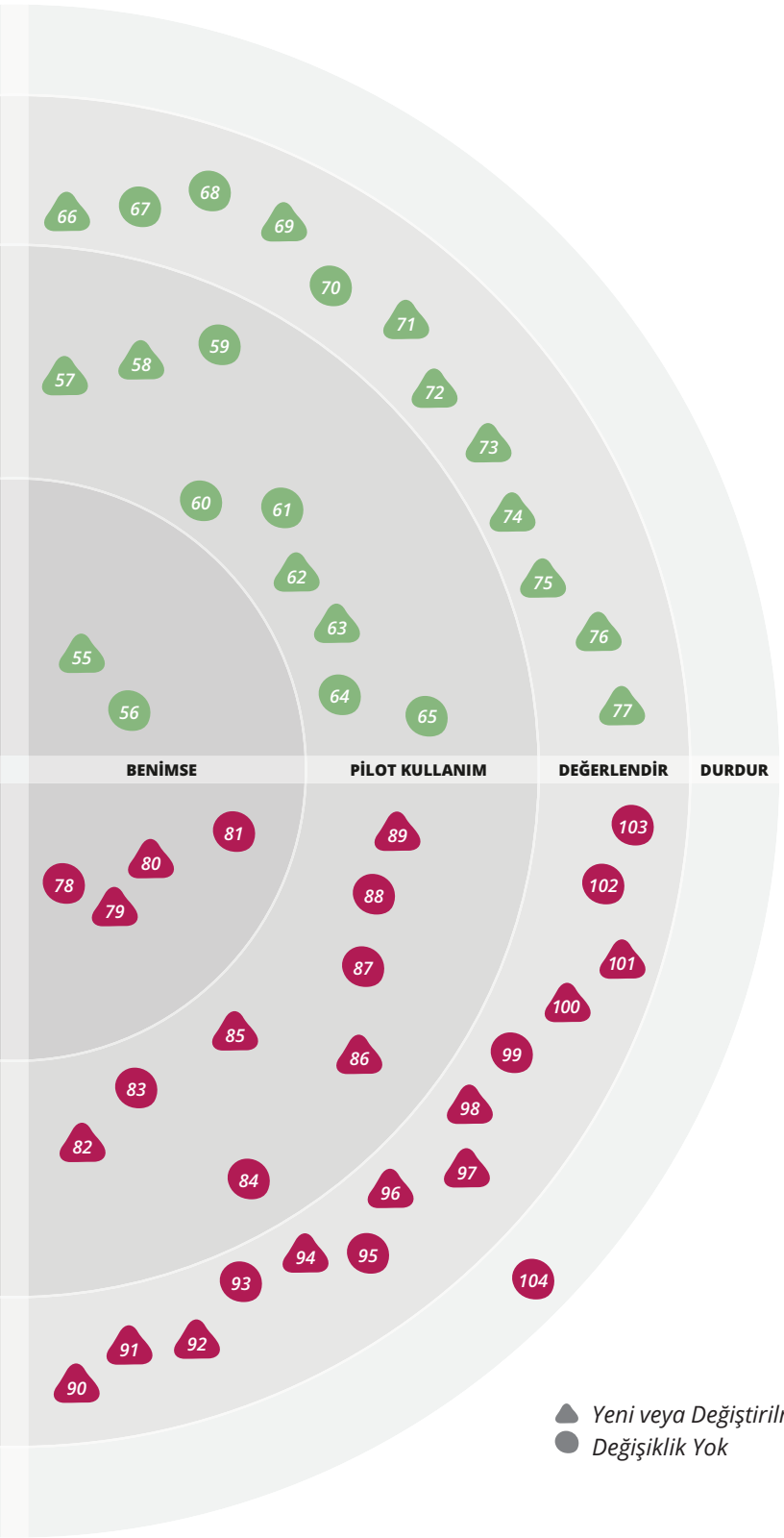
PİLOT KULLANIM
26. Apache Mesos
27. Auth0
28. AWS Device Farm *YENİ*
29. AWS Lambda
30. OpenTracing *YENİ*
31. Oyun Ötesi Unity

DURDUR
32. .NET Core
33. Amazon API Gateway
34. api.ai *YENİ*
35. Cassandra: dikkatli kullanın
36. Bulut tabanlı imaj çözümleme *YENİ*
37. DataStax Enterprise Graph *YENİ*
38. Electron
39. Ethereum
40. Hyperledger *YENİ*
41. IndiaStack
42. Kafka Streams *YENİ*
43. Keycloak *YENİ*
44. Mesosphere DCOS
45. Mosquitto *YENİ*
46. Nuance Mix
47. OpenVR
48. PlatformIO *YENİ*
49. Tango *YENİ*
50. Ses etkileşim platformları *YENİ*
51. Tango *YENİ*
52. wit.ai

DURDUR
53. Bir platform olarak CMS
54. Aşırı iddialı API Gateway



RADAR



ARAÇLAR

BENİMSE
55. fastlane
56. Grafana

PİLOT KULLANIM
57. Airflow YENİ
58. Cake ve Fake *YENİ*
59. Galen
60. HashiCorp Vault
61. Pa11y
62. Scikit-learn
63. Sunucusuz Mimari *YENİ*
64. Talisman
65. Terraform

DEĞERLENDİR
66. Amazon Rekognition *YENİ*
67. Android-x86
68. Bottled Water
69. Claudia *YENİ*
70. Clojure.spec
71. InSpec *YENİ*
72. Molecule *YENİ*
73. Spacemacs *YENİ*
74. spaCy *YENİ*
75. Spinnaker *YENİ*
76. Testinfra *YENİ*
77. Yarn *YENİ*

DURDUR

DİLLER VE FRAMEWORK'LER

BENİMSE
78. Ember.js
79. Python 3
80. ReactiveX
81. Redux

PİLOT KULLANIM
82. Avro *YENİ*
83. Elixir
84. Enzyme
85. Hangfire *YENİ*
86. Nightwatch *YENİ*
87. Phoenix
88. Quick ve Nimble
89. Vue.js

DEĞERLENDİR
90. Angular 2 *YENİ*
91. Caffe *YENİ*
92. DeepLearning.scala *YENİ*
93. ECMAScript 2017
94. Instana *YENİ*
95. JuMP
96. Keras *YENİ*
97. Knet.jl *YENİ*
98. Kotlin *YENİ*
99. Physical Web
100. PostCSS *YENİ*
101. Spring Cloud *YENİ*
102. Three.js
103. WebRTC

DURDUR
104. AngularJS

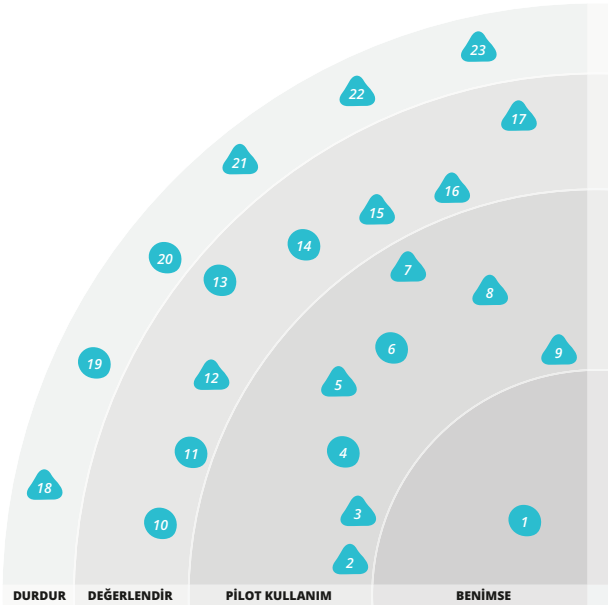
TEKNİKLER

BENİMSE
1. Pipelines as code

PİLOT KULLANIM
2. Bir ürün olarak API
3. Güvenlik bilgilerinin kaynak kodundan ayrıştırılması *YENİ*
4. AB'de PII verilerinin barındırılması
5. Legacy sistemlerinin çevrelenmesi *YENİ*
6. Hafif Tasarımlı Mimari Karar Kayıtları
7. İleri Web Uygulamaları *YENİ*
8. InVision ve Sketch ile prototip oluşturmak *YENİ*
9. Sunucusuz mimari

DEĞERLENDİR
10. İstemci yönlendirmeli sorgu
11. Konteynır güvenlik taraması
12. Sohbet farkındalığı olan API'ler *YENİ*
13. Kademeli Gizlilik
14. Micro ön yüz
15. Platform mühendisliği ürün takımları *YENİ*
16. Sosyal kod analizi *YENİ*
17. Oyunun ötesinde Sanal Gerçeklik

DURDUR
18. Tüm ekipler için tek bir CI kurulumu
19. Anemik REST
20. Büyük Veri özentisi
21. CI tiyatrosu *YENİ*
22. Kurumsal çapta entegrasyon test ortamları *YENİ*
23. Spesifikasyon üzerinden otomatik kod üretimi *YENİ*



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 kurumsal entegrasyon servisinden ayıran nedir? **ÜRÜN OLARAK API** yaklaşımının farkı, API'leri, dahili sistemlerin yada uygulama geliştiricilerin dahi kullanıyor olmasıdır. API'leri tasarlayan ekiplerin müşterilerinin ihtiyaçlarını anlaması ve ürünün onlar için cazip hale gelmesini sağlaması gerekir. Kullanılabilirlik testi ve kullanıcı deneyimi araştırmaları API kullanım kalıplarının daha iyi tasarlanması ve anlaşılmasını sağlayabilir ve API'lere bir ürün zihniyeti kazandırılmasına yardım edebilir. API'ler, ürünler gibi, aktif bir şekilde bakımdan geçirilmeli ve desteklenmeli, ve kullanımları kolay olmalıdır. Bunların, müşterinin haklarını savunan ve sürekli iyileştirme için gayret gösteren sahiplenenleri olmalıdır. Bizim deneyimlerimize göre sıradan kurumsal entegrasyon ile API'lerden oluşan bir platform üzerine kurulmuş çevik bir şirket arasındaki farkı oluşturan eksik bileşen, ürün oryantasyonudur.
Önceki Radar sayılarında, gizli bilgileri kaynak kodunun içinde güvenli bir şekilde saklayan git-crypt ve Blackbox gibi araçlara değinmiştik. **GÜVENLİK BİLGİLERİNİN KAYNAK KODUNDAN AYRIŞTIRILMASI**, bizim teknoloji uzmanlarına sırları saklamanın başka seçenekleri de olduğunu hatırlatma yöntemimiz. Örneğin, HashiCorp vault CI sunucuları ve yapılandırma yönetimi araçları bir uygulamanın kaynak koduyla bağlantısı olmayan sırları saklamak için mekanizmalar sağlıyor. Her ikisi de kullanılabilecek olan bu iki yaklaşımdan, en az birini projelerinizde kullanmanızı tavsiye ediyoruz.

API'leri tasarlayan ekiplerin müşterilerinin ihtiyaçlarını anlaması ve ürünün onlar için cazip hale gelmesini sağlaması gerekir. Kullanılabilirlik testi ve kullanıcı deneyimi araştırmaları API kullanım kalıplarının daha iyi tasarlanması ve anlaşılmasını sağlayabilir ve API'lere bir ürün zihniyeti kazandırılmasına yardım edebilir.

— Ürün olarak API

Uzun süredir geliştirilmiş ve karmaşıklığı artmış kodlarla ve özellikle tek parçadan oluşan devasa sistemlerle çalışmak geliştiriciler için en az tatmin eden ve anlaşmazlık yaşatan deneyimlerden biridir. Karmaşık ve tek parça halindeki sistemleri yaygınlaştırmak ve elde tutmak konusunda temkinli olmanızı tavsiye etmekle birlikte, bunlar bizim ortamlarımızda bağımlılık olmaya devam ediyor ve geliştiriciler çoğu zaman bu bağımlılıkların maliyetlerini ve bunlara yönelik geliştirme çalışması için gerekli zamanı çoğu zaman hafife alıyorlar. Sürtünmeyi azaltmaya katkıda bulunmak için geliştiriciler sanallaştırılmış makine imajları veya Docker konteynırlar ile konteynır imajları kullanarak uzun süredir geliştirilen ve tek parça halindeki sistemlerin (Legacy sistem) ve bunların yapılandırmalarının değiştirilemez kopyalarını yaratmaya çalıştılar. Amaç, geliştiricilerin yerel olarak çalışıp ortamları yeniden kurma, yapılandırma ve paylaşma gerekliliğini ortadan kaldırmaları için **LEGACY SİSTEMLERİNİ ÇERÇEVELEMEKTİR.** İdeal bir senaryoda, legacy sistemlerinin sahibi olan ekipler kendi inşa ettikleri ardışık düzenleri üzerinden kutuya yerleştirilmiş legacy sistem imajlarını oluştururlar ve geliştiriciler bu imajları kendilerine tahsis edilmiş kum havuzlarında daha güvenilir bir şekilde organize edebilirler. Bu yaklaşım her geliştiricinin harcadığı toplam zamanı azaltmış olmakla birlikte, akış yönünde bağlımlılıklara sahip ekiplerin başkalarının kullanımı için konteynır imajları oluşturmak istememeleri durumlarında sınırlı bir başarı elde edebildiler.

İLERİ WEB UYGULAMALARINDAKİ (PWA) artış, kullanıcıların «uygulama tükenmişliğine» karşılık olarak mobil web'i geri getirmeye yönelik en yeni girişim oldu. İlk olarak Google tarafından 2015'te önerilen PWA'lar en ileri teknolojilerden yararlanarak web'in ve mobile özgü uygulamaların en iyi özelliklerini bir araya getiriyor. Service workers, app manifest ve cash ve push API'ler gibi açık standart teknolojilerden bir grubu kullanarak platformlardan bağımsız olan ve uygulama benzeri kullanıcı deneyimi sağlayabilen uygulamalar yaratabiliriz. Bu, web uygulamaları ve mobile özgü uygulamalar arasında denklik oluşturacak ve mobil geliştiricilerin, uygulama mağazalarının kısıtlı ortamının dışındaki kullanıcılara ulaşmasına yardımcı olacaktır. PWA'ları, mobile özgü uygulamalar gibi davranan ve algılanan web siteleri olarak düşünebilirsiniz.

InVision ve Sketch'in birlikte kullanılması bazı kişilerin web uygulaması geliştirmeye bakışını değiştirdi. Bunlar araç olmakla birlikte, bu sinyale önem kazandıran **INVISION VE SKETCH İLE PROTOTİP OLUŞTURULMASIDIR**. Ön yüz ve arka yüz davranışını uygulamak için zengin, tıklanabilir prototipler yaratmak geliştirmeyi hızlandırır ve uygulama detaylarında karışıklıkları ortadan kaldırır. Bu araçların birlikte kullanımı görsel özelliklerin fazla erken detaylandırılması ile etkileşimli deneyim konusunda ilk kullanıcı geri beslemelerinin yakalanması arasında doğru dengeyi kurar.

SUNUCUSUZ MİMARİ uzun zamandır kullanılan uzun soluklu sanal makineler yerine, talep üzerine ortaya çıkan ve kullanıldıktan sonra hemen ortadan kalkan kısa vadeli bilişim gücünü getiriyor. Ekiplerimizin beğendiği bu sunucusuz yaklaşım bizim açımızdan 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. AWS Lambda sunucusuz kelimesiyle neredeyse eş anlamlı olmakla birlikte başka büyük bulut sağlayıcılar da benzer hizmetler sunuyorlar ve biz webtask gibi niş aktörlerin de değerlendirilmesini tavsiye ediyoruz.

Amazon Alexa, Google Voice ve Siri gibi teknolojiler, yazılımla ses tabanlı etkileşimde çıtayı hızla aşağı çektiler. Ancak daha fazla sohbete dayalı bir girdi tarzının (sesli veya yazılı) mevcut birçok API'nin üzerine inşa edilmesi zor olabilir

— Sohbet farkındalığı olan API'ler

Amazon Alexa, Google Voice ve Siri gibi teknolojiler, yazılımla ses tabanlı etkileşimde çıtayı hızla aşağı çektiler. Ancak, özellikle, takip etkileşiminin tüm sohbet bağlamının farkında olması gereken durum bilgisi içeren bir etkileşim tarzında, daha fazla sohbete dayalı bir girdi tarzının (sesli veya yazılı) mevcut birçok API'nin üzerine inşa edilmesi zor olabilir. Bu etkileşim tarzında, örneğin, Manchester'dan Glasgow'a giden trenler hakkında bilgi isteyip sonra da sohbetin bağlamını yeniden belirtmemiz

gerekmeksizin, «İlk kalkış saati kaç?» diye sorabilmek isteyebiliriz. Normal olarak bu bağlam, bir tarayıcıya geri gönderdiğimiz ilk cevapta bulunacaktır ancak sesli arayüzlerde bu bağlamı başka yerde tutmamız gerekir. **SOHBET FARKINDALIĞI OLAN API'LER** backend for front-end kalıbının frontend'in bir sohbet platformunun sesi olduğu bir örneği olabilir. Bu API türü, bu etkileşim tarzının özelliklerini, sohbet durumlarını yönetirken ses ön yüzü adına temel hizmetleri çağırarak ele alabiliyor.

Bulut ve DevOps anlayışının benimsenmesi, artık daha hızlı hareket edebilen ve merkezi operasyon ekiplerine bağımlılığı azalan ekiplerin verimliliğini artırırken, bir uygulamanın tamamını ve operasyon yükünü kendi kendine yönetme becerisi olmayan ekipleri de kısıtladı. Bazı kurumlar bu sorunu çözebilmek için **PLATFORM MÜHENDİSLİĞİ ÜRÜN TAKIMLARI** oluşturma yoluna gittiler. Bu ekiplerin işlettiği dahili platform, teslimat ekiplerinin, sistemlerin kurulum ve işletimini self-servis olarak yapabilmelerini ve teslimat süresi ve operasyon yığın karmaşıklığının azalmasını sağlıyor. Buradaki vurgu API tabanlı self-servis ve destek araçlarındadır ve teslimat ekipleri platform üzerine kurdukları her şeyin desteklenmesinden sorumlu olmaya devam etmektedir. Böyle bir platform takımı kurmaya çalışırken yanlışlıkla ayrı bir DevOps ekibi kurmak ya da sadece mevcut barındırma ve operasyon yapılarını platform takımı olarak yeniden adlandırmaktan kaçınılmalıdır.

SOSYAL KOD ANALİZİ bir geliştiricinin davranışını kodun yapısal analiziyle üst üste bindirerek kod kalitesi anlayışımızı zenginleştirir. Versiyon kontrol sistemlerinden gelen, değişimin sıklık ve zamanı ile değişimi yapan kişi gibi verileri kullanır. Bu tür verileri analiz etmek için kendi kodunuzu yazmayı veya CodeScene gibi araçları kullanmayı tercih edebilirsiniz. CodeScene kablosuz bağlantı alanları ve karmaşık, bakımı zor alt sistemleri dağıtık alt sistemler arasında geçici eşleştirme ve kuruluşunuzda Conway kuralının görünüşünü tanımlayarak yazılım sistemleriniz hakkında daha iyi bir kavrayış kazanmanıza yardımcı olabilir. Dağıtık sistemler, mikro servisler ve dağıtık ekipler gibi teknoloji trendleriyle birlikte kodlarımızın sosyal boyutunun sistemimizin sağlığının bütüncül olarak anlaşılmasında çok önemli hale geldiğine inanıyoruz.

Sanal gerçeklik fikri 50 yıldan fazla bir süredir gündemde ve bilişim teknolojisinde art arda gelen ilerlemelerle birlikte birçok fikir heyecan yarattı ve keşfedildi. Bir dönüm noktasına ulaştığımıza inanıyoruz. Oldukça makul fiyatlı tüketiciye yönelik sanal gerçeklik gözlükleri geçen sene pazara çıktı ve modern grafik kartları bu gözlüklerle birlikte insanı sarmalayan deneyimler yaratmak için yeterli gücü sağlıyor. Gözlükler esas olarak video oyunu tutkunlarını hedefliyor ancak bunların **OYUN ÖTESİ SANAL GERÇEKLİK** için birçok olanağa kapı açacaklarına inanıyoruz. Video oyunları hazırlamakta deneyimli olmayan ekipler iyi 3D modeller ve ikna edici dokular yaratmak için gerekli çaba ve beceriyi hafife almamalıdır.

Sanal gerçeklik fikri 50 yıldan fazla bir süredir gündemde ve bilişim teknolojisinde art arda gelen ilerlemelerle birlikte birçok fikir heyecan yarattı ve keşfedildi. Bir dönüm noktasına ulaştığımıza inanıyoruz.

— Oyunun ötesinde sanal gerçeklik

Tüm ekipler için **TEK BİR CI DURUMU** yaratılmasına karşı uyarıda bulunmak zorundayız. Teoride CI altyapısını konsolide etmek merkezileştirmek hoş bir fikir olsa da bu alanda istenen sonuçlara ulaşmak için araçların da yeterince olgunlaştığını düşünmüyoruz. Merkezi CI sunumunu kullanmak zorunda olan yazılım teslimat ekipleri, merkezi bir ekibin küçük konfigürasyon işlerini yapmasına veya paylaşılan altyapı veya araçlarda küçük sorunları gidermesine bağımlı olmaları nedeniyle sürekli olarak uzun gecikmeler yaşıyor. Bu aşamada, kuruluşların kalıplar, kılavuzlar hazırlamak için merkezi yatırımlarını sınırlamalarını tavsiye ediyoruz ve teslimat ekiplerinin kendi CI altyapılarını işletmelerini destekliyoruz.

Uzun zamandır sürekli entegrasyon (CI) savunucusu olduk ve check-in'ler üzerine otomatik olarak projeler yapacak CI sunu programları yapmakta öncülük ettik. İyi kullanıldığı takdirde bu programlar geliştiricilerin her gün katkıda bulunduğu ortak bir proje ana hattı üzerindeki bir uyuyan kod prosesi gibi çalışır. CI sunucusu projeyi yapar ve geniş kapsamlı testler yaparak tüm yazılım sisteminin entegre edildiğinden ve her zaman yayınlanabilir durumda olduğundan

emin olarak Sürekli teslimat ilkelerini yerine getirmeye çalışır. Ne yazık ki, bazı geliştiriciler sadece CI sunucusunu kuruyor ve yanlış bir şekilde «CI yaptıklarını» varsayıyorlar ancak aslında bunun hiçbir faydasını kullanamıyorlar. Yaygın başarısızlık şekilleri şunlar: CI'yi bir ortak ana hat karşısında çalıştırmak ancak katkıların sık olmaması ve aslında entegrasyonun sürekli olmaması, yapımı zayıf bir test kapsamıyla yürütmek; yapımın uzun süreler boyunca kırmızıda kalmasına izin vermek veya CI'yi özellik dalları karşısında yürütmek ve bunun sürekli izolasyona neden olması. Ortaya çıkan "**CI TİYATROSU**" insanların iyi hissetmesini sağlayabilir ancak hiçbir güvenilir CI sertifikasyon testinden geçemeyecektir.

Ne yazık ki, bazı geliştiriciler sadece CI sunucusunu kuruyor ve yanlış bir şekilde «CI yaptıklarını» varsayıyorlar ancak aslında bunun hiçbir faydasını kullanamıyorlar. Ortaya çıkan "CI TİYATROSU" insanların iyi hissetmesini sağlayabilir ancak hiçbir güvenilir CI sertifikasyon testinden geçemeyecektir.

— CI tiyatrosu

İşletme genelindeki üç aylık veya aylık sürümlerin en iyi uygulama kabul edildiği dönemde, üretime yönelik kurulum öncesinde test döngülerini gerçekleştirmek için komple bir ortam bulundurmak gerekiyordu. Bu **İŞLETME GENELİNDE ENTEGRASYON TEST ORTAMLARI** (çoğu zaman SIT veya canlıya hazırlama ortamı olarak anılır) bugün sürekli teslimat için yaygın bir dar boğaz oluşturuyor. Bu ortamların kendisi kırılgandır ve bakımları masraflıdır, bileşenlerine ise çoğu zaman ayrı bir ortam yönetim ekibi tarafından manüel konfigürasyon yapılması gerekir. Canlıya hazırlama ortamında test yapılması güvenilmez ve yavaş geri besleme temin ederken, bileşenlere yalıtık halde test uygulanmasına kıyasla test için harcanan zahmet iki katına çıkar. Kuruluşların, ana bileşenler için üretime geçişte bağımsız bir yolu kademeli oluşturmalarını tavsiye ediyoruz. Önemli teknikler arasında, kontrat testi, kurulumu sürümden ayrıştırma, düzeltme için gerekli ortalama süreye odaklanma ve üretimde test etmedir.

SOAP'ın işletme yazılım sektörüne egemen olduğu günlerde WSDL şartnamelerinden müşteri kodu oluşturma yöntemi kabul edilen - hatta teşvik edilen - bir uygulamaydı. Ne yazık ki, sonuçta ortaya çıkan kod çoğu zaman karmaşık, test edilemez, değiştirilmesi zor oluyordu ve uygulama platformlarının hepsinde çalışmıyordu. REST'in çıkmasıyla birlikte, sadece ihtiyaç duyulan alanları elde etmek ve işlemek için toleranslı okuyucu kalıbını kullanan API istemcilerini dönüştürmenin daha iyi olduğu kanaatine vardık. Son dönemde, eski alışkanlıklara rahatsız edici bir dönüş olduğunu ve Swagger veya RAML araçlarıyla yazılmış API şartnamelerinden kod üreten geliştiricilerin olduğunu gözlemledik ki biz bu işleme **ŞARTNAME ÜZERİNDEN OTOMATİK KOD ÜRETİMİ** diyoruz. Bu tür araçlar API'lerin tasarımını yönlendirmek ve dokümantasyon çıkarmak için çok yararlı olmakla birlikte istemci kodları doğrudan bu şartnamelerden üretmek gibi cazip bir kısa yola sapmaya karşı uyarıyoruz. Bu tür bir kodun test edilmesi ve bakımının yapılması büyük ihtimalle zor olacaktır.

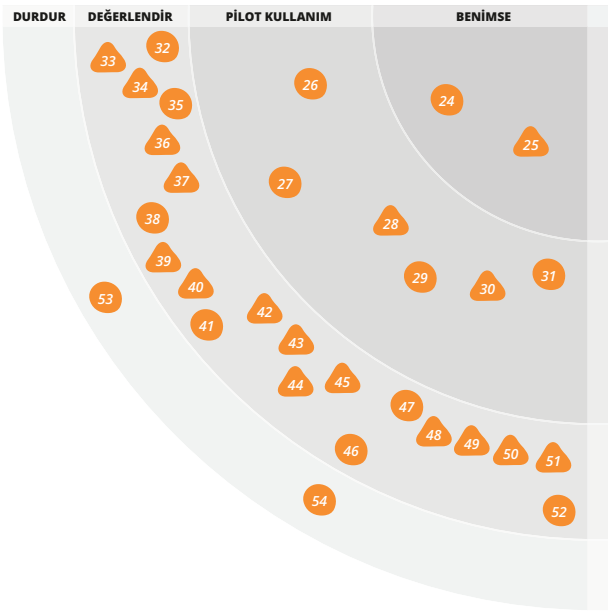
PLATFORMLAR

BENİMSE
24. HSTS
25. Linux güvenlik modülleri

PİLOT KULLANIM
26. Apache Mesos
27. Auth0
28. AWS Cihaz Farm *YENİ*
29. AWS Lambda
30. OpenTracing *YENİ*
31. Oyun Ötesi Unity

DEĞERLENDİR
32. .NET Core
33. Amazon API Gateway
34. api.ai *YENİ*
35. Cassandra: dikkatli kullanın
36. Bulut tabanlı imaj çözümleme *YENİ*
37. DataStax Enterprise Graph *YENİ*
38. Electron
39. Ethereum
40. Hyperledger *YENİ*
41. IndiaStack
42. Kafka Streams *YENİ*
43. Keycloak *YENİ*
44. Mesosphere DCOS
45. Mosquitto *YENİ*
46. Nuance Mix
47. OpenVR
48. PlatformIO *YENİ*
49. TangoYENİ
50. Ses platformları *YENİ*
51. Tango *YENİ*
52. wit.ai

HOLD
53. Bir platform olarak CMS
54. Aşırı iddialı API Gateway



En Az Ayrıcalık İlkesi bizi yazılım bileşenlerini sadece ihtiyaçları olan kaynaklara erişebilecekleri şekilde sınırlamaya teşvik ediyor. Ancak varsayım olarak, bir Linux prosesi - rasgele portlara bağlanmaktan, yeni dallara ayrılmaya kadar- kullanıcısının yapabildiği her şeyi yapabilir. Güvenlik uzantılarının çekirdeğe eklenebilmesini sağlayan **LINUX GÜVENLİK MODÜLLERİ** (LSM) framework'ü MAC ion Linux'u uygulamak için kullanılageliyor. SELinux ve AppArmor çekirdek ile birlikte gelen hakim ve en çok tanınan LSM uyumlu uygulamalardır. Ekiplerin bu güvenlik framework'lerinden birini kullanmayı öğrenmelerini tavsiye ediyoruz (bu yüzden bunu Benimse halkasına koyduk). Bunlar, ekiplere, paylaşılan ana bilgisayar üzerinde konteynırlı hizmetler dahil olmak üzere kimin hangi kaynaklara erişim hakkının olacağı konusundaki soruları değerlendirmekte yardımcı oluyor. Erişim yönetimine bu muhafazakar yaklaşım ekiplerin SDLC prosesi içinde güvenliği geliştirmelerine yardımcı olacaktır.

Amazon'un, geliştiricilerin API hizmetlerini İnternet kullanıcılarına açabilmelerini sağlayan **AMAZON API GATEWAY** sunumu, trafik yönetimi, izleme, kimlik doğrulama ve izin verme gibi alışılmış API kapısı özelliklerini içinde barındırıyor. Bizim ekiplerimiz sunucusuz mimarilerin parçası olarak AWS Lambda karşısında bu hizmeti kullandıklarında olumlu deneyimler edindiler. Diğer yandan EC2 üzerinde çalışan HTTP/ HTTPS uç noktaları karşısında daha genel amaçlı bir kapı olarak kullanırken daha fazla zorlukla karşılaştık - burada VPC'lerle birlikte çalışabilirlik olmaması ve kapı ile müşteri sertifikası doğrulamasının yapılmasındaki zorluklar bize engel çıkardı. Deneyimimizin bu şekilde hem olumlu hem olumsuz yönleri olması nedeniyle ekiplere AWS API Gateway'i Lambda ile birlikte deneme olarak kullanmalarını ancak daha genel koşullar altında kullanmak için uygunluğunu değerlendirmelerini tavsiye etmek istiyoruz.

Mobil cihaz çeşitlerinin devasa bir sayıya ulaşmış olması, şirketlerin mobil uygulamalarını bütün cihazlar üzerinde test etmelerini neredeyse imkansız hale getiriyor. Android, iOS ve web uygulamalarınızı, bulutta eş zamanlı olarak barındırılan çok çeşitli fiziksel cihaz üzerinde çalıştırabilmenizi ve bunlarla etkileşime girebilmenizi sağlayan bir uygulama test hizmeti olan **AWS DEVICE FARM**'a girin. Her çalıştırmada, ayrıntılı log kayıtları, performans grafikleri ve ekran görüntüleri oluşturularak genel ve cihaza özgü geri beslemeler veriliyor. Bu hizmet, çok spesifik test senaryolarını yeniden üretmek için her cihazın durumu ve konfigürasyonunun değiştirilmesine izin vererek büyük bir esneklik sağlıyor. Bizim ekiplerimiz, cihazlar üzerinde uygulamaları için en geniş yükleme alanı ile uçtan uca testler yapmak için AWS Device Farm kullanıyorlar.

Yekpare uygulamalar yerini daha karmaşık mikroservis ekosistemlerine bırakırken çeşitli servislerin hepsinde taleplerin iz kaydını yapmak norm haline geliyor.

— OpenTracing

Yekpare uygulamalar yerini daha karmaşık (mikro)servis ekosistemlerine bırakırken çeşitli servislerin hepsinde taleplerin iz kaydını yapmak norm haline geliyor. Neyse ki **OPENTRACING** dağıtık iz kaydı için hızla fiili standart haline geliyor. Uber, Apple, Yelp ve diğer büyük oyuncular tarafından geliştirilen bu araç Zipkin, Instana ve Jaeger gibi çok sayıda iz sürücüyü destekliyor. OpenTracing şu anda altı dilde satıcı farkı gözetmeyen bir uygulama yapma imkanı sağlıyor: Go, JavaScript, Java, Python, Objective-C ve C++.

Sohbet robotları ve ses etkileşim platformlarının, son zamanlardaki yükselişine paralel olarak, metinden niyeti çıkaran ve katılabileceğiniz konuşma akışının yönetimini yapan bir hizmet sunan **API.AI** gibi araç ve platformlarının yaygınlaştığını gördük. Kısa bir süre önce Google tarafından satın alınan bu «servis olarak doğal dil anlama» sunumu, bu alandaki wit.ai ve Amazon'a ait Lex gibi diğer oyuncularla rekabet ediyor.

İmaj çözümleme eskiden bir karanlık sanat olarak, sahada çalışan verilerle ilgili bilim insanlarından oluşan bir ekip gerektiriyordu. Ancak son yıllarda, imaj ve yüz sınıflandırma/kategorize etme, yüz karşılaştırma, yüz noktaları tanımlama ve yüz tanıma gibi sorunları çözmeye daha yaklaştık. **BULUT TABANLI İMAJ ÇÖZÜMLEME** AR uygulamalarının ve fotoğraf etiketleme ve sınıflandırma içeren her şeyin yerine geçebilen Amazon Rekognition, Microsoft Computer Vision API ve Google Cloud Vision API gibi servisler üzerinden makine öğrenmesi olanaklarına erişim sağlıyor.

DATASTAX ENTERPRISE GRAPH (DSE Graph) ile büyük grafik veri tabanlarını işlemekte ilk aşamada bazı başarılar elde ettik. Cassandra üzerine kurulu DSE Graph uzun zamandır favorimiz olan Neo4j'nin bazı sınırlamalar göstermeye başladığı büyük veri seti tipini hedefliyor. Bu ölçeğin kendi getirileri ve götürüleri vardır; örneğin ACID işlemlerini ve Neo4j'in çalışma süresi boyunca şemasız çalışma niteliğini kaybediyorsunuz ancak altyapıdaki Cassandra tablolarına erişim, Spark'ın analitik iş yükleri için entegre olması ve güçlü TinkerPop/Gremlin sorgulama dili bunun değerlendirilmeye değer bir seçenek olmasını sağlıyor.

Blockchain'ler ve şifrelenmiş para birimleri için heyecanın zirvesi geride kalmış gözüküyor, bunun kanıtı da bu konuda önceki yangın hortumu misali akan açıklamaların yavaşlamış olmasıdır ve daha spekülatif çabaların bazılarının zaman içinde söneceğini tahmin ediyoruz. Blockchain'lerden biri olan **ETHEREUM** blockchain meraklıları arasında evrensel bir popülerliği olmamakla birlikte yeni girişimlerde giderek daha fazla görülüyor. Etherum, geliştiricilerin, ether'in (Ethereum şifreli para birimi) blockchain'ler üzerinde oluşan aktiviteye cevaben yaptığı algoritmik hamleler olan "akıllı kontratlar" geliştirmesini sağlayan dahili bir programlama diline sahip ve kamuya açık bir blockchain'dir. Bankalar için blockchain oluşturma konsorsiyumu olan R3Cev, Ethereum üzerinde ilk konsept 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.

HYPERLEDGER blockchain teknolojileri çerçevesinde inşa edilen bir platformdur. Doku adı verilen bir blockchain uygulaması ve diğer ilgili araçlardan oluşur. Blockchain konusunu saran heyecanı dikkate almayan ekiplerimiz bu araçlarla işe başlamanın kolay olduğunu düşünüyor. Linux Foundation tarafından desteklenen bir açık kaynak platformu olması da bizim Hyperledger konusundaki heyecanımızı artırıyor.

KAFKA STREAMS akış uygulamaları inşa etmek için kullanılan bir hafif kütüphanedir. Akış işlemeyi, eşzamansız hizmetler için bir ana akım uygulama programlama modeli olarak kolaylıkla erişilebilir hale getirebilecek kadar sadeleştirmek amacıyla tasarlandı.

Bir küme çalıştırmanın (genellikle tam donanımlı akış işleme framework'lerinin getirdiği) karmaşıklığına katlanmaksızın sorununuza bir akış işleme modeli uygulamak istediğiniz senaryolarda iyi bir alternatif olabilir

Bir mikroservisteveya herhangi bir başka dağıtık mimaride, en yaygın ihtiyaçlardan biri hizmetleri veya API'leri kimlik doğrulama veya izin verme özellikleri üzerinden güvenceye almaktır. Keycloak işte bu noktada gündeme geliyor. **KEYCLOAK** açık kaynaklı bir kimlik ve erişim yönetimi çözümü olarak uygulamaların ve mikro servislerin çok az kod kullanılarak veya hiç kod kullanılmadan güvence altına alınmasını kolaylaştırıyor. Temel özellik olarak, tek oturum açma, sosyal oturum açma ve OpenID Connect, OAuth2 ve SAML gibi standart protokolleri destekliyor.

MESOSPHERE DCOS, konteynırlı uygulamalar ve Docker'ın içinde çalıştırmak istemediğiniz uygulamalar için temel altyapınızı soyutlayan Mesos üzerine inşa edilen bir platformdur. Mütevazi kurulumlar için bu, aşırı güç kullanmak anlamına gelebilir ancak hem ticari hem de açık kaynak versiyonlarında başarı elde edilmeye başladığını gördük. Özellikle, farklı bulut hizmeti sağlayıcıları ve tahsis edilmiş donanım arasında taşınabilirliği kolaylaştırmasını ve konteynırlı iş yükleri için tek bir konteynır organizasyon framework'ünü kullanmak zorunda olunmamasını beğendik. Versiyon yükseltmeler bizim istediğimizden biraz daha karmaşık olabilmekle birlikte genel yığın iyi bir şekilde dengeleniyor.

Bizim deneyimimizde - birçok cihazın birbiriyle ve/veya bir veri merkeziyle iletişim kurduğu eşyaların interneti için - MQTT bağlanabilirlik protokolü kendini kanıtladı. **MOSQUITTO**'nun kullandığı MQTT aracısını beğenmeye başladık. Özellikle ölçeklenebilirlikle ilgili olarak tüm talepleri karşılayamayabilir ancak kompakt niteliği ve kolay kurulumu, geliştirme ve test amacıyla kullanımda ideal olmasını sağlıyor.

PLATFORMIO çapraz-platform yapımları, kütüphane yönetimi ve mevcut IDE'lerle iyi entegrasyon sağlayarak Eşyaların İnternetine yönelik geliştirmeler için zengin bir ekosistem oluşturuyor. Dahili terminal ve seri port monitörü ile akıllı kod tamamlama ve akıllı kod ayıklama geliştirici deneyimini büyük ölçüde iyileştiriyor. Aynı zamanda binlerce kütüphaneyi organize ve muhafaza ederek, eşyalarının internetinin gelişimini kolaylaştırmak amacıyla semantik versiyonlama kullanan bir temiz bağımlılık yöneticisi sağlıyor. PlatformIO'yu birkaç eşyaların interneti projesinde kullanmaya başladık ve sadeliğini ve geniş bir platformlar ve tablolar yelpazesini desteklemesini gerçekten beğendik.

Donanım gereksinimleri ve sanal dünyaları yaratmak için gerekli çaba nedeniyle giriş çıtası görece daha yüksek olan sanal gerçekliğin (VR) yanı sıra, alternatif gerçeklik (AR) ve karma gerçeklik (MR) de geçen sene ana akıma katıldı. Pokémon Go, sıradan akıllı telefonların cazip AR/MR deneyimleri yaratmak için yeterliği olduğuna ilişkin yeterince kanıt sağladı. **TANGO** cep telefonları için, AR/MR olanaklarını daha da zenginleştiren yeni bir donanım sensörü teknolojisidir. Kullanıcının çevresinden ayrıntılı 3-D ölçümler alınmasını sağlayarak sanal nesnelerin daha inandırıcı hale getirilmesini ve kamera beslemesine daha inandırıcı bir şekilde yerleştirilebilmesini mümkün kılıyor. Tango teknolojisine sahip ilk telefonlar şu anda satışa sunulmuş durumda.

Amazon Alexa ve Google Home gibi **SES İŞLEME PLATFORMLARI** gündemde çok yer alıyor ve hatta bazıları sohbet tabanlı ses arayüzünün her yerde kullanılabileceğini müjdeliyor. Sohbet tabanlı kullanıcı arayüzlerini ürünlere entegre etmeye ve bu yeni etkileşimin arayüzleri tasarlama şeklimizi nasıl etkilediğini görmeye başlamış durumdayız. Alexa özel olarak temelden itibaren ekransız olarak inşa edildi ve sohbet tabanlı kullanıcı arayüzlerini birinci sınıf olarak ele alıyor. Ancak söylenenlere inanmak için henüz erken ve oyuna katılan büyük oyuncuların sayısının artmasını bekliyoruz.

Sohbet tabanlı kullanıcı arayüzlerini ürünlere entegre etmeye ve bu yeni etkileşimin arayüzleri tasarlama şeklimizi nasıl etkilediğini görmeye başlamış durumdayız.

— Ses işleme platformları

WEBVR tarayıcınız üstünden VR cihazlarına erişebilmenizi sağlayan deneysel bir JavaScript API'sidir. Sektörden destek aldı ve bir gecede yapılan işlerin yanı sıra bazı sürümlerde de bulunabiliyor. Tarayıcınızda VR deneyimleri oluşturmak istiyorsanız başlamak için burası çok iyi bir yerdir. Bu teknoloji ile birlikte, Three.js, A-Frame, ReactVR, Argon.js ve Awe.js gibi araçlar AR deneyimlerini tarayıcıya taşıyor. Bu alandaki araç sağanağı, internet komisyon standartlarıyla birlikte AR ve VR'nin benimsenmesini güçlendirmeye yardımcı olabilir.

ARAÇLAR

BENİMSE
55. fastlane
56. Grafana

PİLOT KULLANIM
57. Airflow YENİ
58. Cake ve Fake *YENİ*
59. Galen
60. HashiCorp Vault
61. Pa11y
62. Scikit-learn
63. Sunucusuz Mimari *YENİ*
64. Talisman
65. Terraform

DEĞERLENDİR
66. Amazon Rekognition *YENİ*
67. Android-x86
68. Bottled Water
69. Claudia *YENİ*
70. Clojure.spec
71. InSpec *YENİ*
72. Molecule *YENİ*
73. Spacemacs *YENİ*
74. spaCy *YENİ*
75. Spinnaker *YENİ*
76. Testinfra *YENİ*
77. Yarn *YENİ*

DURDUR

Web uygulama geliştiricileri, farklı uygulama iş akışlarını sadeleştirmek ve otomatikleştirmekte zorluk çekmiyorlar; sürüm süreçlerini otomatikleştirmelerine yardımcı olmak için geliştirilmiş, aralarında tercih yapabilecekleri çok çeşitli çözümler bulunuyor. Ancak, mobil için geliştirme yaparken, yapım, test, dağıtım, ekran görüntüleri oluşturma, imzalama ve uygulamaları dağıtma işleri için iki farklı yöntemi olan iki farklı işletim sistemiyle uğraşıyoruz. Bu zorlukları azaltmak için ekiplerimiz iOS ve Android uygulamalarının sürüm sürecini otomatikleştirmek için başvuru aracı olarak **FASTLANE**'i benimsediler. Sade konfigürasyonlar ve çoklu ardışık düzenler kullanarak mobil geliştirme için sürekli teslimat gerçekleştirmeyi başarabiliyorlar.

AIRFLOW veri ardışık düzenlerini programatik olarak yaratmak, programlamak ve izlemek için bir araçtır. DAG'lere (Directed Acyclic Graphs) kod muamelesi yaparak, bakımı yapılabilir, versiyonlanabilir ve test edilebilir veri ardışık düzenlerini teşvik ediyor. Bu konfigürasyonu projelerimizde kullandık ve sade ve açık veri iş akışları ile sonuçlanan dinamik ardışık düzenler yaratmayı başardık. Airflow operatör ve uygulayıcılarınızı tanımlamanızı ve kütüphaneyi sizin ortamınıza uyan soyutlama düzeyine denk gelecek şekilde genişletmenizi kolaylaştırıyor.

Ekiplerimiz iOS ve Android uygulamalarının sürüm sürecini otomatikleştirmek için başvuru aracı olarak FASTLANE'i benimsediler.

— fastlane

MSBuild, 2005'te kullanıma girdiğinden beri .NET ekosisteminin önde gelen oluşturma sistemi oldu ancak daha önce Maven için vurguladığımız aynı zayıflıklara sahip. .NET topluluğu MSBuild'e, bakımı daha kolay ve daha esnek olan aynı zamanda proje büyürken daha akıcı olan alternatifler geliştirmeye başladı. Bu alternatiflerden ikisi **CAKE** ve **FAKE**'tir. Cake, C# içinde oluşturulmuş bir DSL kullanırken Fake F# kullanıyor. Bunların her bir son bir yıl içinde önemli büyüme

gösterdiler ve .NET projeleri içindeki ortak oluşturma görevlerinin organize edilmesinde MSBuild'e ciddi bir alternatif olabileceklerini kanıtladılar.

.NET topluluğu MSBuild'e, bakımı daha kolay ve daha esnek olan aynı zamanda proje büyürken daha akıcı olan alternatifler geliştirmeye başladı.

— Cake ve Fake

SCIKIT-LEARN yeni bir araç değil (10. yıldönümüne yaklaşıyor); yeni olan makine öğrenimi araç ve tekniklerinin akademi ve büyük teknoloji şirketleri dışındaki benimsenme oranıdır. Güçlü bir modeller seti ve zengin bir işlevsellik seti sağlayan Scikit-learn makine öğrenimi konsept ve olanaklarının daha geniş (ve çoğu zaman uzman olmayan) bir alıcı kitle için daha erişilebilir hale gelmesinde önemli bir rol oynuyor.

Popüler **SUNUCUSUZ FRAMEWORK** esas olarak AWS Lambda ve diğer AWS sunumlarını kullanarak sunucusuz uygulamaların iskelesini oluşturmak ve kurulumunu yapmak için araç sağlıyor. JavaScript, Python, Java ve C# için şablon desteği sağlayan Sunucusuz Framework'ün sahip olduğu aktif topluluk, framework'ü genişleten eklentilerle katkıda bulunuyor. Framework aynı zamanda, AWS Lambda'nın bir alternatifi olarak Apache kuluçka projesini destekliyor.

AMAZON REKOGNITION bu Radar'da söz ettiğimiz bulut bilişim ile imaj tanımlama araçlarından biridir. Bu konuda beğendiğimiz özellik, Amazon'un yüz tanıma ile ortaya çıkan bazı gizlilik endişelerini yatıştırmak için (GUIDS kullanarak) yüzlerin AWS için anonim olmasını sağladı.

AWS Lambda ile Amazon API Gateway'in birleşimi hizmetler ve API'lerin kurulumunu yapma şeklimizi çok etkiledi. Ancak sunucusuz konfigürasyonda bile parçaları bir araya getirmek için gerekli konfigürasyon miktarı önemsiz değil. **CLAUDIA** JavaScript ile yazılan AWS Lambda fonksiyonlarının ve API Gateway konfigürasyonlarının kurulumunu otomatikleştiren bir araçtır. Ekiplerimiz makul sayıda varsayılan değer ve ayarlar sağlayan Claudia'nın Lambda tabanlı mikro servislerle çalışmaya hızla başlamalarına izin verdiğini tespit ettiler.

Bir şirket teslimat ekiplerine özerklik verirken yine de kurulu çözümlerinin güvenli ve uyumlu kalmasını nasıl sağlayabilir? Sunucular kurulduktan sonra, işletim ömürleri boyunca güvenli ve uyumlu kalacaklarından nasıl emin olabilirsiniz? InSpec bu sorunları ele almaya çalışıyor. **INSPEC**, ilhamını Serverspec'in verdiği bir altyapı test aracıdır ancak yapılan modifikasyonlar, bu aracı binlerce sunucuda uyumluluk sağlamaları gereken güvenlik profesyonelleri için daha yararlı hale getiriyor. Münferit testler bir araya getirilerek bütünlüklü güvenlik profilleri oluşturulabilir ve bir komut satırı üzerinden uzaktan yürütülebilir. InSpec geliştiriciler için yararlıdır ancak kurulu üretim altyapısını sürekli test etme yönünde genişlemekte ve üretimde QA doğrultusunda ilerlemektedir.

MOLECULE Ansible görevlerinin geliştirilmesi ve test edilmesi için tasarlandı. Sanal bir makine veya konteynır üzerinde Ansible görevini yürütmek için iskeleyi kurduğumuz zaman, test ortamımızı manüel olarak kurmamıza gerek kalmıyor. Molecule, Vagrant, Docker ve OpenStack'ten yararlanarak sanal makineler/konteynırları yönetiyor ve testleri yürütmek için Serverspec, Testinfra veya Goss'u destekliyor. Sıralı tesis modelinde kendiliğinden var olan adımlara şunlar dahildir: sanal makine yönetimi, Ansible ile harmanlama, eş zamanlılık testi ve yakınsama testi. Oldukça genç bir proje olmakla birlikte büyük kullanım potansiyeli olduğunu düşünüyoruz.

Herhangi bir Emacs hayranının size söyleyeceği gibi Emacs bir metin editörünün ötesinde yazı tipinden bağımsız karakteri tanıyan uygulamalar için bir platformdur. Son birkaç yıl içinde bu platformda yeni geliştirme patlaması oldu ancak biz **SPACEMACS**'in özel ilgiyi hak ettiğini düşünüyoruz. Spacemacs, yeni bir klavye kullanıcı arayüzü, basitleştirilmiş özelleştirme katmanları ve düzenlenmiş bir Emacs paketleri dağıtımıyla Emacs platformuna giriş yapmanızı sağlıyor. Projenin amaçlarından biri, Vim kullanıcı arayüzünü Emacs'ın dahili yeniden programlanabilirlik özelliği ile birleştirerek her iki dünyanın en iyisi olmaktır. Geliştirici

verimlilik araçlarını, etkili yazılım geliştirmenin hayati bir parçası olarak görüyoruz ve eğer bir süredir Emacs'ı düşünmediyseniz, Spacemacs'ın bu klasik geliştirme platformunu nasıl yeniden değerlendirdiğine bir göz atmanızı öneririz.

SPACY Python'da yazılmış bir Doğal Dil İşleme (NLP) kütüphanesidir. Geliştiriciler tarafından üretimde kullanılmak üzere tasarlanmış yüksek performanslı bir kütüphane olan spaCy sıklıkla duygu ifadeleriyle ve tutarsız noktalama işaretleriyle karışan metinleri işlemek için uygun olan NLP modellerini uygular. Diğer NLP framework'lerinden farklı olarak spaCy, eklenebilir bir kitaplıktır ve bir platform değildir; Araştırmaya yönelik veri modelinin eğitilmesinden ziyade üretim ortamindaki uygulamalara yöneliktir. TensorFlow Python AI ekosisteminin diğer unsurlarıyla çok uyumludur. Biz spaCy'yi insan dilini girdi olarak alıp, kullanıcıların iş kararları almasına yardımcı olan bir arama motoru oluşturmak için kurumsal bağlamda kullandık.

Netflix mikroservis sürekli teslimat (CD) platformu olan **SPINNAKER**'ı açık kaynak haline getirdi. Diğer CI / CD platformlarına kıyasla, Spinnaker kullanıma hazır kopyaların küme yönetimi ve buluta konuşlandırılmasını birinci sınıf özellikler olarak gerçekleştirir. Google Cloud Platform, AWS ve Pivotal Cloud Foundry gibi birçok bulut hizmet sağlayıcısı için kullanıma hazır konuşlandırma işlemleri ve küme yönetimini destekler. Jenkins ile bir iş planı oluşturmak için Spinnaker'ı Jenkins ile entegre edebilirsiniz. Spinnaker'ın mikroservisleri buluta konuşlandırmak için kullandığı kendi çizgisini koruyan yaklaşımı beğeniyoruz - sadece Spinnaker'ın ardışık düzenlerinin bir kullanıcı arayüzü üzerinden oluşturulmuş olması ve kod olarak yapılandırılamaması hariç.

Bugün altyapı araçlarının geniş çapta kullanılmakta olduğu göz önüne alındığında, mevcut projelerde kod olarak altyapı kullanımının artmış olması da sürpriz sayılmamalıdır. Bu eğilim, bu kodu test etme ihtiyacını ortaya çıkarıyor. Manüel olarak veya Ansible, Puppet ve Docker gibi araçlarla yapılandırılmış sunucularınızın gerçek durumunu **TESTINFRA** kullanarak test edebilirsiniz. Python'da bir Serverspec eşdeğeri olmayı hedefleyen Testinfra Pytest test motoruna eklenti olarak yazıldı.

Bugün altyapı araçlarının geniş çapta kullanılmakta olduğu göz önüne alındığında, mevcut projelerde kod olarak altyapı kullanımının artmış olması da sürpriz sayılmamalıdır.

— Testinfra

YARN npm istemcisi için mevcut iş akışının yerine geçerken npm kayıtlarıyla uyumlu kalan yeni bir paket yöneticisidir. Npm istemcisiyle, bağımlılıkların yüklü olduğu sıraya göre devre modülleri altında farklı bir ağaç yapısına sahip olabiliriz. Bu deterministik olmayan nitelik, "benim makinemde bekleyen işler" sorunlarına yol açabilir. Kurulum adımlarını çözümleme, veriyi alma ve ilişki oluşturma kısımlarına bölen Yarn, bu sorunları deterministik algoritmalar ve erişimi düzenleyen dosyalar (lockfiles) kullanarak önler ve böylece kurulumların başka makinelerde de tekrarlanabilir olmasını garanti eder. Ayrıca, Yarn indirdiği tüm paketleri önbelleğe aldığı için sürekli entegrasyon (CI) ortamımızda oluşturma işlerimizin önemli ölçüde hızlandığını gördük.

DİLLER VE FRAMEWORK'LER

BENİMSE
78. Ember.js
79. Python 3
80. ReactiveX
81. Redux

PİLOT KULLANIM
82. Avro *YENİ*
83. Elixir
84. Enzyme
85. Hangfire *YENİ*
86. Nightwatch *YENİ*
87. Phoenix
88. Quick ve Nimble
89. Vue.js

DEĞERLENDİR
90. Angular 2 *YENİ*
91. Caffe *YENİ*
92. DeepLearning.scala *YENİ*
93. ECMAScript 2017
94. Instana *YENİ*
95. JuMP
96. Keras *YENİ*
97. Knet.jl *YENİ*
98. Kotlin *YENİ*
99. Physical Web
100. PostCSS *YENİ*
101. Spring Cloud *YENİ*
102. Three.js
103. WebRTC

DURDUR
104. AngularJS

PYTHON 3 geçmişe dönük olarak Python 2.x ile uyumlu olmayan birçok yararlı özellik getirdi. Ayrıca, daha önce geçmişe uyumluluk için korunmuş olan birçok Python 2.x özelliği kaldırılarak; Python 3, öğrenip kullanması daha kolay ve dilin diğer unsurları ile daha uyumlu hale getirildi. Python 3'ü makine öğrenimi ve web uygulaması geliştirme gibi alanlarda kullanma deneyimimiz hem dilin hem de destekleyen kütüphanelerin çoğunun benimsenebilecek kadar olgunlaştığını gösteriyor. Mevcut kütüphaneleri kullanarak ufak sorunları ayrıştırıp yamalayabildik veya uyumsuz olan ve artık desteklenmeyen Phyton 2.x kütüphanelerinin kullanımından kaçındık. Python'da geliştirme yapıyorsanız Python 3 kullanmanızı kuvvetle tavsiye ederiz.

Dağıtık sistemler genel sistem verimliliğini artırmak için, çoğu zaman, çoklu kullanım, olay tabanlı iletişim ve bloke etmeyen girdi/çıktı (I/O) kullanır. Bu programlama teknikleri, düşük düzeyli kullanım, senkronizasyon, iş parçacığı güvenliği, eşzamanlı veri yapıları ve bloke etmeyen girdi/çıktı (I/O) gibi zorluklar getirir.

Açık kaynak **REACTIVEX** kütüphanesi, bunları güzelce soyutluyor, uygulamanın gerektiği şekilde olgunlaştırılmasını sağlıyor ve akan asenkron olay kayıtları üzerinde gözlenebilir (observable) deseni genişletiyor. ReactiveX ayrıca aktif bir geliştirici topluluğuna sahip ve gittikçe büyüyen bir dil listesini destekliyor; en son eklenen dil ise RxSwift oldu. Ayrıca mobil ve masaüstü platformlarına bağlanmayı da gerçekleştiriyor.

Açık kaynak REACTIVEX Kütüphanesi, bunları güzelce soyutluyor, uygulamanın gerektiği şekilde olgunlaştırılmasını sağlıyor ve akan asenkron olay kayıtları üzerinde gözlenebilir (observable) deseni genişletiyor.

— ReactiveX

AVRO veriyi seri hale getirmek için kullanılan bir framework'tür. Şemayı mesaj içeriği ile birlikte depolayarak şemanın dönüşümünü teşvik ediyor. Üreticiler alan adlarını düzenleyebiliyor, yeni alanlar

ekleyebiliyor veya mevcut alanları silebiliyor, Avro ise müşterilerin mesajları tüketmeye devam etmesini sağlıyor. Bir şemaya sahip olmak, her bir verinin ek yük olmadan yazılmasına olanak tanıyor; bu da sıkıştırılmış veri kodlaması ve daha hızlı veri işlemeyi mümkün kılıyor. Üretici ve tüketici arasında yapısız mesaj alışverişinin esnek olmasına rağmen, ekiplerin konuşlandırmalar sırasında kuyruktaki uyumlu olmayan işlenmemiş mesajlarla sorun yaşadığını gördük. Çok sayıda projede Avro'yu kullandık ve sadece yapısız mesajlar göndermek için Avro'yu kullanmanızı öneririz.

Kullanımı kolay ve esnek olan Hangfire işlevsel bir tarz benimsiyor. Özellikle ilginç olan, bir görevin durumunu kaydetme yeteneğidir ve böylece bir uygulama, bir kilitlenme veya kapanma sonrasında yeniden başlatıldığında Hangfire da çalışmaya yeniden başlayabiliyor.

— Hangfire

Uygulama geliştirmede sık görülen bir sorun, ana proses dışında periyodik olarak veya belirli koşullar sağlandığında çalışan görevleri nasıl planlayacağımızdır. Uygulamanın kapanması gibi beklenmedik olaylar oluştuğunda sorun daha da karmaşıklaşır. Ekibimiz **HANGFIRE** framework'ünün bunu ve daha fazlasını .NET ortamında yapabildiğini keşfetti. Kullanımı kolay ve esnek olan Hangfire işlevsel bir tarz benimsiyor. Özellikle ilginç olan, bir görevin durumunu kaydetme yeteneğidir ve böylece bir uygulama, bir kilitlenme veya kapanma sonrasında yeniden başlatıldığında Hangfire da çalışmaya yeniden başlayabiliyor.

NIGHTWATCH JavaScript'te oluşturulacak ve Node.js'de çalıştırılacak tarayıcı tabanlı testler için otomatik kabul testlerini mümkün kılan bir framework'tür. Nightwatch, akıcı bir API kullanarak testlerin tanımlanmasını sağlıyor ve testler daha sonra bir Selenium / Webdriver sunucusuna uygulanabiliyor. Tek sayfalı uygulamalar veya diğer JavaScript ağırlıklı sayfalar söz konusu olduğunda, otomatik testler kodun çoğunluğuyla aynı dilde ve ortamda oluşturulup çalıştırılabiliyor.

Ön yüz JavaScript framework'lerin sürekli değişen dünyasında ortaya çıkan favorilerden biri **VUE.JS**. Vue.js, AngularJS'ye hafif tasarımlı bir alternatiftir. Çok esnek - ve kendi çizgisini daha az koruyan - bir kütüphane olması ve modülerlik, bileşenler ve reaktif veri akışı gibi kavramlar etrafında etkileşimli web arayüzleri oluşturmak için bir dizi araç sunması için tasarlandı. Düşük bir öğrenme eğrisi olması, tecrübesiz geliştiriciler ve yeni başlayanlar için ilgi çekici olmasını sağlıyor. Ancak şuna dikkat edilmelidir ki, Vue.js her katman için gelişmiş bir framework değildir - sadece görüntü katmanına odaklanır ve bu nedenle başka kütüphaneler ve mevcut projelerle entegre edilmesi kolaydır.

Önceki Radar'da, AngularJS'yi Durdur halkasına taşımıştık (bu sayıda da aynı yerde kalıyor). **ANGULAR 2**'ye gelince karışık mesajlar görüyoruz. Geçtiğimiz yıl boyunca ThoughtWorks'teki bazı ekipler Angular 2'yi başarıyla kullandı ve bunun iyi bir seçim olduğunu düşünüyorlar. Ancak Angular 2, AngularJS'nin evrim geçirmiş hali değil bir yeniden yazımıdır ve AngularJS'den Angular 2'ye geçiş, AngularJS'den başka bir framework'e geçişten çok farklı değildir. Bizim deneyimlerimize göre daha üstün olan, React.js, Ember.js ve Vue.js gibi rakiplerin varlığı göz önünde bulundurulduğunda, Angular 2 için güçlü bir tavsiyede bulunmak konusunda hâlâ tereddüt ediyoruz. Bununla birlikte, özellikle TypeScript'i satın aldıysanız, bunun kötü bir seçim olmadığını vurgulamak isteriz.

CAFFE Berkeley Vision and Learning Center tarafından yaratılmış açık kaynaklı bir derin öğrenme kütüphanesidir. Çoğunlukla bilgisayar görüntü uygulamaları için katlamalı sinir ağları üzerinde yoğunlaşmaktadır. Caffe bilgisayar görüntü görevleri için iyi ve popüler bir tercihtir ve Caffe kullanıcılarından tarafından yapılan birçok başarılı modeli kullanıma hazır bir şekilde Caffe Model Zoo'dan indirebilirsiniz. Keras gibi Caffe de Python tabanlı bir API'dir. Bununla birlikte, Keras'ta modeller ve bileşenler doğrudan Python kodunda oluşturulmuş nesnelerdir, oysa Caffe modelleri Protobuf yapılandırma dosyaları tarafından tanımlanır. Her iki yaklaşımın da artı ve eksileri vardır ve ikisi arasında dönüşüm yapmak da mümkündür.

DEEPLEARNING.SCALA ThoughtWorks'teki meslektaşlarımız tarafından Scala'da yaratılmış olan açık kaynak kodlu bir derin öğrenme araç kitidir.
Bu proje bize heyecan veriyor, çünkü sinir ağları oluşturmak için diferansiyel fonksiyon programlama kullanıyor; bir geliştirici basitçe derlenme anında tip kontrolü yaparak Scala'da kod yazıyor. DeepLearning.scala şu anda float, double, GPU ile hızlandırılmış N-boyutlu dizilerin yanı sıra cebirsel veri türleri gibi temel türleri destekliyor. Bu araç kitinin daha üst düzey işlevleri ve Spark'ta dağıtık öğretmeyi destekleyeceği söylenen gelecek sürümlerini merakla bekliyoruz.

INSTANA çok kalabalık olan uygulama performans yönetimi alanına yeni giren bir diğer araç. Baştan sona bulut kökenli mimariler için yapılmış olması Instana'yı rakiplerinin birçoğundan farklılaştırıyor. Özellikleri arasında dinamik keşif, dağıtık iz kaydı ve servis sağlığı ve altyapınızın görüntüsünün problem olayının gerçekleştiği ana kaydırılması olanağı bulunuyor. Bu ürünün Consul, Prometheus ve OpenTracing'in çeşitli uygulanma şekilleri gibi, aynı şeyi yapan açık kaynak kodlu projelerin bileşimi karşısında cazip hale gelip gelemeyeceğini zaman gösterecek; ancak kullanıma hazır bir çözüme ihtiyacınız varsa bir bakmaya değer.

KERAS Python'da sinir ağları oluşturmak için kullanılan üst düzey bir arayüzdür. Google'da çalışan bir mühendis tarafından yaratılan açık kod kaynaklı Keras TensorFlow veya Theano üzerinde çalışıyor. CPU veya GPU'lar üzerinde öğretmek üzere kullanılan güçlü derin öğretme algoritmaları yaratmak için şaşırtıcı derecede sade bir arayüz sunuyor. Keras, modülerlik, basitlik ve genişleyebilirlik göz önünde bulundurularak çok iyi tasarlanmış. Örneğin Caffe gibi kütüphanelerden farklı olarak, Keras yinelenen ağlar gibi daha genel ağ yapılarını desteklemesi nedeniyle metin analizi, NLP ve genel makine öğrenmesi için daha kullanışlı. Bilgisayar görüşü veya makine öğrenmesinin başka herhangi bir dalıyla öncelikli olarak ilgileniyorsanız, Caffe daha uygun bir seçim olabilir. Ancak basit ama güçlü bir framework öğrenmek istiyorsanız ilk tercihiniz Keras olmalıdır.

KNET.JL Koç Üniversitesi'nin Julia'da Deniz Yüret ve arkadaşları tarafından gerçekleştirilen derin öğrenme framework'üdür. Kullanıcıları sınırlı bir mini dil içinde kalmaya zorlayan Theano ve TensorFlow gibi gradyan üreten (gradient-generating) derleyicilerden farklı olarak Knet, Julia'nın tüm gücünü ve etkileyiciliğini kullanarak makine öğrenimi modellerinin tanımının ve eğitiminin yapılmasına izin veriyor. Knet hemen hemen tüm Julia kodlarının otomatik olarak ayırt edilmesi için çalışma süresince üretilen dinamik hesaplamalı grafikleri kullanıyor. Gerçekten GPU işlemlerinin KnetArray türü üzerinden desteklenmesini çok sevdik. Bir GPU makinesine erişiminiz yoksa, Knet'in arkasındaki ekip önceden yapılandırılmış bir Sanal Makine Kopyası (Amazon Machine Image - AMI) bulunduruyor; böylece Knet değerlendirmenizi bulutta yapabiliyorsunuz.

KOTLIN programlama dilini deneyip değerlendirmek, geliştiricilerimizin bu yıl 'yapilacaklar listeleri'nin birçoğunda bulunuyor, bazıları ise şimdiden başarıyla üretim sürecinde kullanmış durumdalar. Bu, JetBrains'in açık kaynak kodlu bir JVM dilidir. Söz dizimi olarak Swift'e daha yakın ve onun kadar hassas olduğu için bizim Swift mobil geliştiricilerimiz tarafından çok seviliyor. Java geliştiricilerimiz, Java dili ve araçları ile sorunsuz bir şekilde birlikte çalışabilmesinden hoşlandılar ve Scala'dan daha kolay öğrenildiğini gördüler. Kotlin, işlevsel programlama konseptlerini desteklemekle birlikte Scala'dan daha az özelliğe sahip. Ekiplerimizdeki derleme anında tip kontrolü yapılmasını seven geliştiriciler, hiçbir sorun tespit etmediler ve kendilerini daha az standart test yazarken buldular.

Mikro servislerden oluşan sistemler oluşturan ekipler, servis keşfi, yük dengeleme, devreyi kesme ve sağlık kontrolü gibi koordinasyon teknikleri üzerine düşünmelidir.

— Spring Cloud

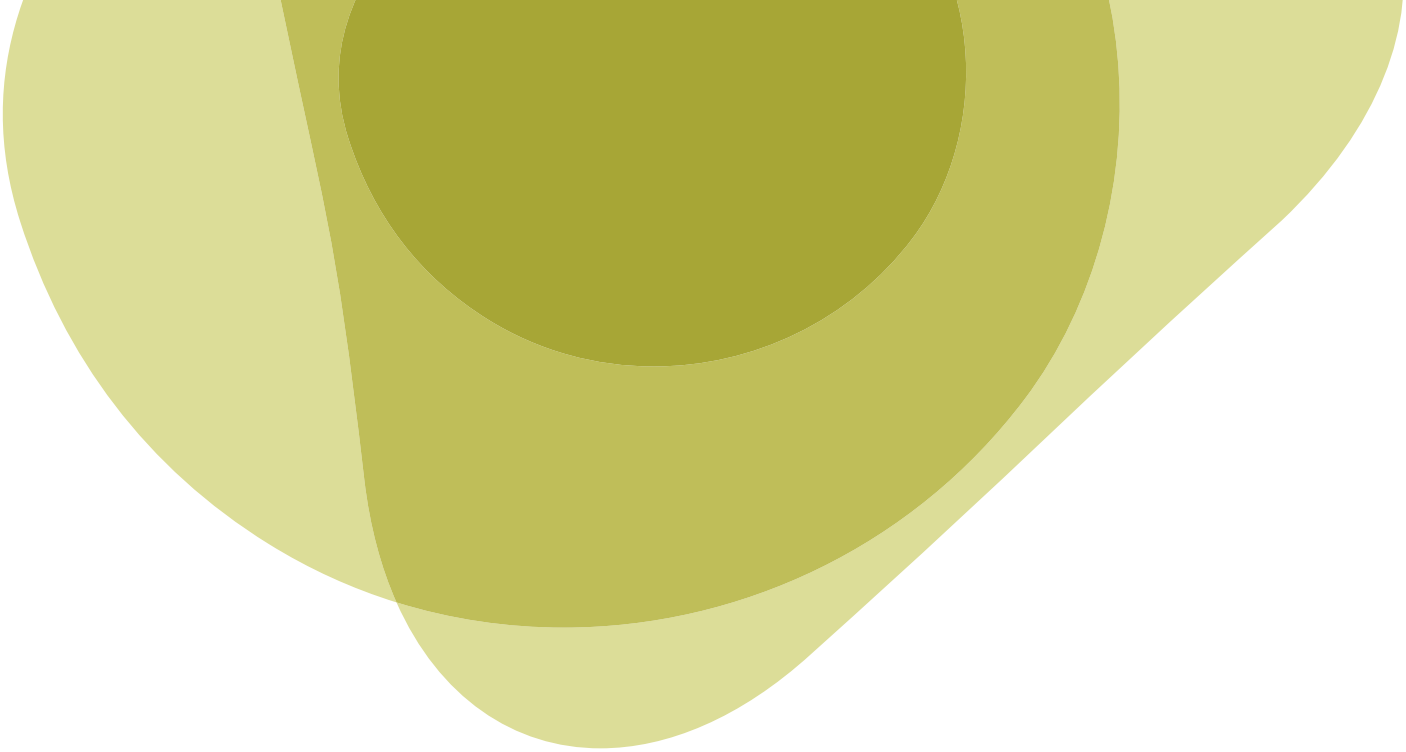
POSTCSS zengin bir eklenti ekosistemine sahip olan CSS dokümanlarının soyut bir söz dizimi ağacı (syntax tree-based) gösterimi üzerinde işletilmek üzere yapılmış Node.js tabanlı JavaScript framework'üdür. Çoğu zaman yanlış bir şekilde bir ön-işlemci (SaaS veya Less gibi) olarak düşünülse de biz, PostCSS'nin gerçek gücünün, harmanlama (stylelint eklentisi), çapraz derleme (sugarss eklentisi), modül seçici çakışmasını önlemek için isim sınıfı çözümlemesi (moduller eklentisi), şablon CSS kodu üretimi (autoprefixer eklentisi), küçültme gibi zengin bir eklenti grubu sayesinde yapılan birçok şeyden kaynaklandığını düşünüyoruz. Eklentiler farklı olgunluk seviyelerinde olmakla birlikte, PostCSS'nin kendisi, CSS'yi ön-yüz geliştirmek için kullanılan tam teşekküllü bir dil haline getiren basit ve güçlü bir framework olmaya devam ediyor.

Mikro servislerden oluşan sistemler oluşturan ekipler, servis keşfi, yük dengeleme, devre kesici ve sağlık kontrolü gibi koordinasyon teknikleri üzerine düşünmelidir. Bu tekniklerin birçoğu ekiplerin araçlarını oluşturmasını gerektiriyor ve bu da her zaman kolay bir iş olmuyor. **SPRING CLOUD** projesi geliştiricilere bu koordinasyon tekniklerini tanıdık Spring ortamında kullanabilmeleri için gerekli araçları sunuyor. Bu araçlar, Consul, ZooKeeper ve tüm Netflix OSS'yi destekliyor ve bu araçların hepsini beğeniyoruz. Basitçe ifade etmek gerekirse, bu araç setleriyle doğru işler yapmak kolaylaşıyor. Spring'le ilgili her zamanki endişelerimiz, yani karmaşıklığın çok büyük bir bölümünü saklaması devam etmekle birlikte, ekosistemin içindeyseniz ve bu sorunları çözmeniz gerekiyorsa Spring Cloud'u düşünmeniz gerekir.

Teknoloji Radarı çıktığında ilk öğrenen siz olun ve özel web seminerleri ve içerik hakkında güncel bilgilere ulaşmak için

HEMEN ABONE OLUN

thght.works/Sub-TR



ThoughtWorks®

ThoughtWorks, yazılım danışmanlığı, üretimi ve ürünleri konusunda uzmanlaşmış tutkulu ve hedef sahibi bireylerin oluşturduğu bir topluluk ve danışmanlık şirketidir. Müşterilerimizin teknolojiyi işlerinin merkezine yerleştirmelerine yardım ediyoruz. Olumlu toplumsal değişime adanmış bir topluluk olarak misyonumuz insanlığı yazılım üzerinden daha iyi anlamak ve aynı doğrultuda mücadele eden birçok kuruluşla ortak hareket etmektir.

20 yıl önce kurulan ThoughtWorks, bugün yazılım ekipleri için öncü araçlar üreten bir ürün birimini de kapsayan ve 4000'den fazla çalışanı olan bir şirkete dönüşmüştür. ThoughtWorks 14 ülkede (Avustralya, Brezilya, Şili, Çin, Ekvador, Almanya, Hindistan, Singapur, Güney Afrika, Türkiye, Birleşik Krallık ve ABD) 40 ofisi bulunmaktadır.

thoughtworks.com