ThoughtWorks®

TECHNOLOGY RADAR *NOV '16*

Els nostres pensaments sobre la tecnologia i les tendències que estan modelant el futur

thoughtworks.com/radar

#TWTechRadar

QUÈ HI HA DE NOU?

Aquests són els temes principals d'aquesta edició:

"DOCKER COM A PROCÉS, PAAS COM A MÀQUINA, ARQUITECTURA DE MICROSERVEIS COM A MODEL DE PROGRAMACIÓ."

L'estil d'arquitectura en microserveis remarca l'augment d'abstraccions en el món dels desenvolupadors a causa de la proliferació de contenidors i de l'èmfasi en l'acoblament baix, el qual ofereix un gran nivell d'aïllament operacional. Els desenvolupadors poden veure els contenidors com a processos independents i el PaaS (plataforma com a servei) com l'objectiu de desplegament comú i utilitzant l'arquitectura de microserveis com a estil comú. Desacoblar l'arquitectura permet el mateix pels equips, facilitant la coordinació entre ells. El seu atractiu tant per a desenvolupadors com per a DevOps ha fet que s'hagi convertit en el nou estàndard de desenvolupament a moltes organitzacions.

ENFORTIMENT INTEL•LIGENT

Temes relacionats amb I+D llargament discutits com l'aprenentatge automàtic o la intel•ligència artificial tenen, de sobte, aplicacions pràctiques gràcies a entorns de treball com Nuance Mix i TensorFlow. Els desenvolupadors poden descarregar entorns de treball que van des de la programació neurolingüística fins a llibreries d'aprenentatge automàtic. Ens alegra veure com empreses posen a disposició d'un gran nombre de desenvolupadors per mitjà del codi obert eines i llibreries sofisticades que, sinó, serien increïblement cares i, per tant, de difícil accés fa només una dècada. Molts factors han evolucionat i s'han combinat per fer possibles noves eines: la computació col•laborativa que té com objectiu components específics com ara les GPU o els recursos al núvol. És possible que la recollida massiva de dades comenci a donar els seus fruits...

L'EFECTE HOLÍSTIC DE L'ESTRUCTURA D'EQUIP

Team structure has always had a large impact on a wide variety of software development subjects, and has become an area of increased focus given foundations such as self-service PaaS and microservices. Companies now favor product thinking over projects; tech companies are popularizing the "you build it, you run it" style of team autonomy, and we're seeing the same product thinking applied to enterprise projects. When restructuring teams yields better results, it illustrates once again that software development is still mostly a communication problem. Building cross-functional teams increases the beneficial surface area of communication across traditionally segregated job roles, which in turn removes friction imposed by artificial structures like silos.

RA I RV A L'ABAST DE TOTHOM

Estem veient com la realitat augmentada (RA) i la realitat virtual (RV), antigament considerades només per a jocs o com a articles curiosos, atrauen cada vegada més l'interès de les empreses. Mentre que l'atenció del públic per la RA va venir gràcies a equips de desenvolupament de programari (SDK per les seves sigles en anglès) mòbils que permetien caçar dibuixos virtuals, components com Oculus Rift, HTC Vive i Microsoft HoloLens estan madurant fins al punt que els primers usuaris poden gaudir d'ells sense haver de jugar amb una tecnologia immadura. Tot i que plataformes com OpenVR i Unity fa temps que existeixen, noves eines de processament de llenguatge (NLP per les seves sigles en anglès) com Nuance Mix i components que proporcionen interaccions naturals tindran un impacte enorme en l'adopció de RA i RV. Nosaltres ja estem experimentant amb RA i RV a les nostres oficines per trobar-hi futures aplicacions com ara la interacció a distància o el wayfinding. Els nostres experiments demostren que RV és una eina sorprenentment poderosa per a una narració i col•laboració a distància més empàtica gràcies a la seva habilitat per eliminar l'abstracció i submergir l'usuari en una experiència. De tota manera, però, haurem de fer front a molts reptes per a poder crear i entregar contingut de RA i RV ja que els coneixements i la tècnica estan molt endarrerits comparats amb la maquinaria, especialment en el món empresarial.

COL•LABORADORS

El Radar Tecnològic ha sigut creat pel Gabinet d'Assessors de ThoughtWorks Technology, format per:

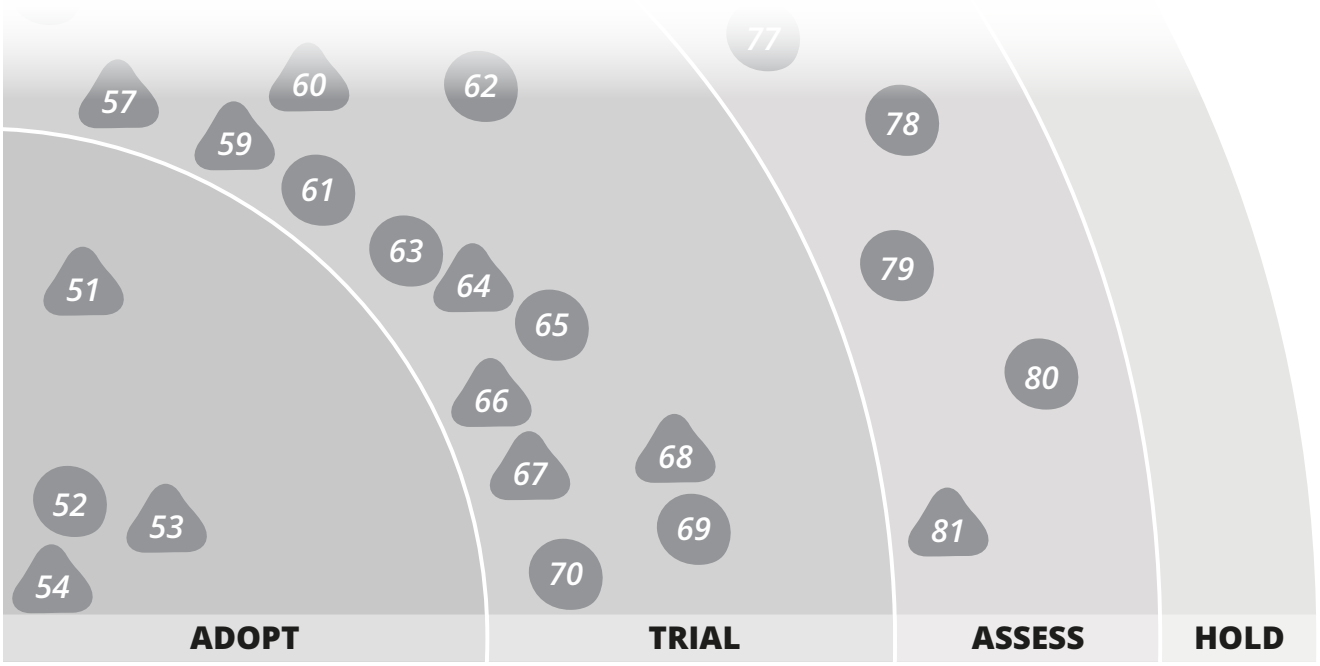
Rebecca Parsons (CTO)
Martin Fowler (Chief Scientist)
Anne J Simmons
Badri Janakiraman
Bharani Subramaniam
Camilla Falconi Crispim
Erik Doernenburg
Evan Bottcher
Fausto de la Torre
Hao Xu
Ian Cartwright
James Lewis
Jiaxing Chen
Jonny LeRoy
Marco Valtas
Mike Mason
Neal Ford
Rachel Laycock
Scott Shaw
Srihari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

© November 2016, ThoughtWorks, Inc. All Rights Reserved.

SOBRE EL RADAR TECNOLÒGIC

Als treballadors de ThoughtWorks els apassiona la tecnologia. La creem, l'investiguem, la posem a prova, la compartim gràcies al codi obert, escrivim sobre ella i sempre la volem millorar per a beneficiar a tothom. La nostra missió és arribar a l'excel•lència i revolucionar la informàtica. Amb aquesta missió en ment, creem i compartim el Radar Tecnològic. El Radar el crea el gabinet d'assessors de ThoughtWorks Technology, un grup de líders sèniors de ThoughtWorks. Es reuneixen periòdicament per a parlar sobre l'estratègia global de ThoughtWorks i les tendències tecnològiques que tenen un impacte significatiu sobre la nostra industria.

Aquest Radar captura l'essència de les xerrades del gabinet d'assessor i la comparteix en un format adequat per a una gran varietat d'actors, des de gerents de sistemes (CIO) fins a desenvolupadors. Com que entenem el Radar com un resum, recomanem que s'explorin més detalladament aquestes tecnologies. El Radar té una naturalesa més aviat gràfica i agrupa els elements en tècniques, eines, plataformes i llenguatges i entorns de treball. Si algun element pot aparèixer en més d'un quadrant, escollim el que ens sembla més adient. Agrupem aquests elements en 4 anells per a reflectir el que en pensem. Els anells són:



Creiem fermament que la industria hauria d'adoptar aquests elements. Nosaltres els utilitzem quan ho creiem oportú en els nostres projectes.

Val la pena seguir investigant. És important entendre com desenvolupar aquesta habilitat. Les empreses haurien de provar aquesta tecnologia en un projecte en el que es pogués permetre el risc.

Val la pena explorar-la per a poder entendre com afectaria l'empresa.

Continuar amb precaució.

Els elements nous o que han canviat de manera significativa des de l'últim Radar es representen amb un triangle mentre que els elements que no s'han mogut es representen amb un cercle. Ens interessen molts més temes dels que podem posar en un document d'aquesta mida per la qual cosa hem deixat esvair molts elements del Radar anterior per a fer lloc als nous elements. Que ja no hi sigui no vol dir que ja no ens interessi.

Per a més informació sobre el Radar, visiteu thoughtworks.com/radar/faq

© November 2016, ThoughtWorks, Inc. All Rights Reserved.

EL RADAR

TÈCNIQUES

ADOPT
1. Consumer-driven contract testing
2. Pipelines as code new
3. Threat Modeling

TRIAL
4. APIs as a product new
5. Bug bounties
6. Data Lake
7. Hosting PII data in the EU
8. Lightweight Architecture Decision Records new
9. Reactive architectures
10. Serverless architecture

ASSESS
11. Client-directed query new
12. Container security scanning new
13. Content Security Policies
14. Differential privacy new
15. Micro frontends new
16. OWASP ASVS
17. Unikernels
18. VR beyond gaming

HOLD
19. A single CI instance for all teams
20. Anemic REST new
21. Big Data envy
22. Cloud lift and shift

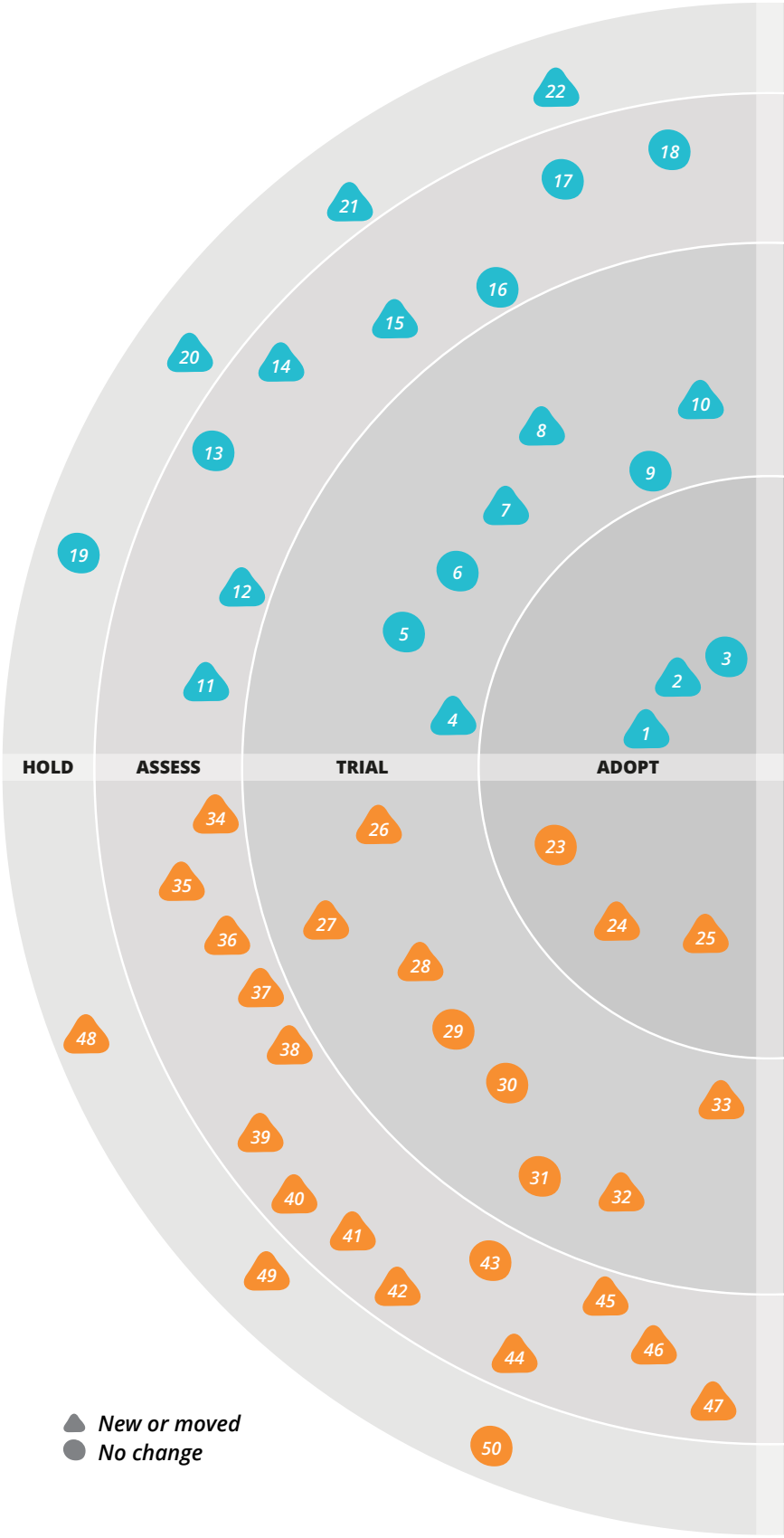
PLATAFORMES

ADOPT
23. Docker
24. HSTS
25. Linux security modules

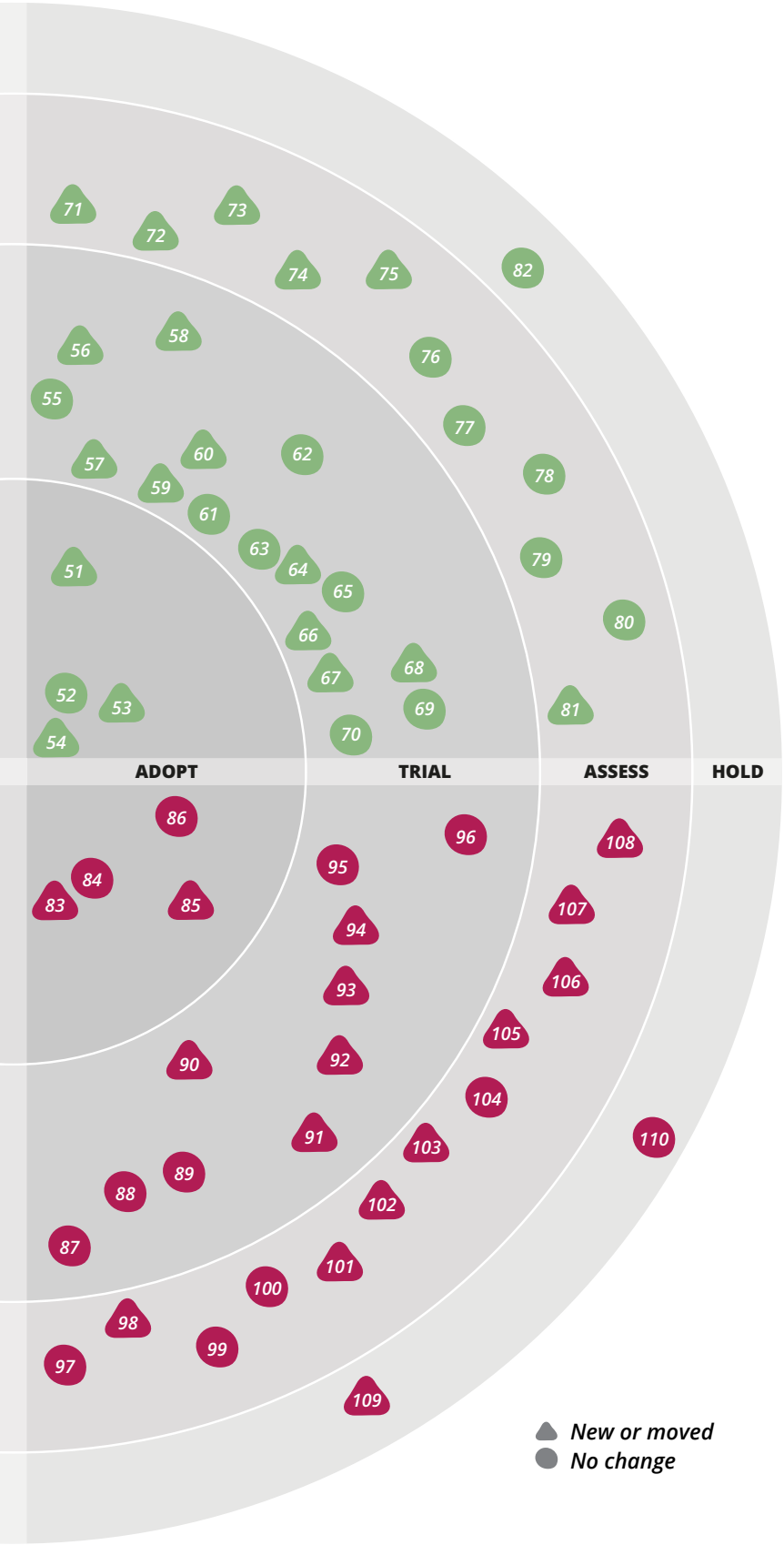
TRIAL
26. Apache Mesos
27. Auth0 new
28. AWS Lambda
29. Kubernetes
30. Pivotal Cloud Foundry
31. Rancher
32. Realm
33. Unity beyond gaming new

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

HOLD
48. CMS as a platform
49. Overambitious API gateway
50. Superficial private cloud



© November 2016, ThoughtWorks, Inc. All Rights Reserved.



EL RADAR

EINES

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

TRIAL
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

ASSESS
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

HOLD
82. Jenkins as a deployment pipeline

LLENGUATGES & ENTORNS DE TREBALL

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

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

ASSESS
97. Aurelia
98. ECMAScript 2017 new
99. Elm
100. GraphQL new
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

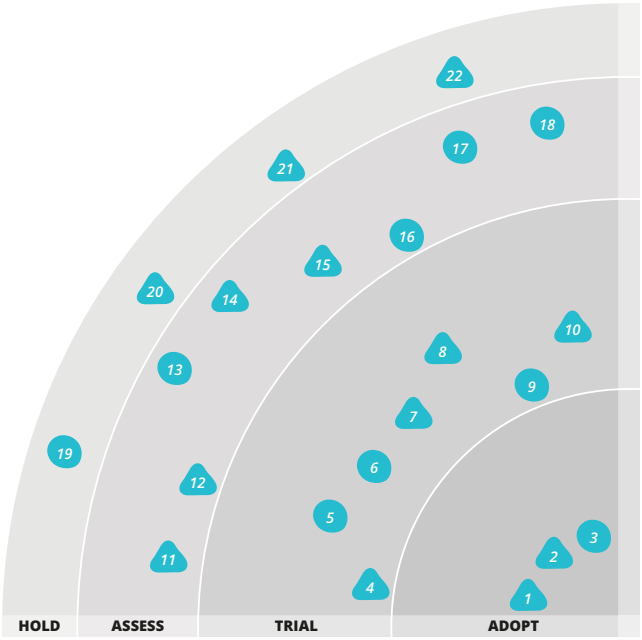
HOLD
109. AngularJS

© November 2016, ThoughtWorks, Inc. All Rights Reserved.

TÈCNIQUES

Per aquesta edició hem decidit recuperar les **proves de contracte guiats pel consumidor** tot i que anteriorment vam deixar que s'esvaís. El concepte no és nou però, ara que els microserveis estan sent acceptats per la majoria, hem de recordar que els contractes guiats pels consumidors són una part essencial d'un portafoli madur de proves de microserveis que permeti els desplegaments independents de serveis. Volem remarcar, a més, que les proves de contractes guiats pels consumidors són una tècnica i una actitud que no requereixen que s'implementi cap eina especial. Ens encanten els entorns de treball com el Pact perquè permeten que es puguin implementar amb més facilitat les proves de contracte en certs contextos. Hem vist, però, una tendència per la qual els equips es centren més en l'entorn de treball que en la pràctica general. Dur a terme proves Pact no garanteix que s'estiguin creant contractes guiats pels consumidors. A més, en moltes situacions s'haurien de crear bons contractes guiats pels consumidors fins i tot quan no hi ha cap eina de proves implementada.

Els equips estan buscant la automatització en els seus entorns i, fins i tot, en la seva estructura de desenvolupament. **Pipelines as code** defineix el desplegament de canals via codi en lloc de configurar una eina CI/CD en funcionament. LambdaCD, Drone, GoCD i Concourse són exemples d'eines que permeten l'ús d'aquesta tècnica. A més, eines d'automatització de la configuració per a sistemes CI/CD, com GoMatic, es poden utilitzar per tractar el canal de desplegament com a codi (adaptat i provat).



Les empreses han adoptat massivament les API com a la manera de demostrar les capacitats de l'empresa tant per a desenvolupadors interns com externs. Les API permeten experimentar sense esforços amb noves idees de l'empresa recombinant els elements essencials. Aleshores, però, què diferencia una API d'un servei ordinari d'integració d'empresa (EIS)? Una de les diferències és que es tracta les **API com un producte**, fins i tot quan el consumidor és un sistema intern. Els equips que creen APIs haurien d'entendre les necessitats dels seus clients i fer que el producte sigui adequat per a ells. Els productes, a més, es milloren, es mantenen i es dóna suport durant molt de temps. Haurien de tenir un propietari que defensi el consumidor i té com objectiu la millora constant. Els productes es mantenen i suporten activament i són fàcils de trobar i de fer servir. La nostra experiència ens diu que la orientació del producte és el que diferencia un servei ordinari d'integració d'empresa (EIS) i una empresa flexible construïda sobre una plataforma de APIs.

En alguns països veiem com empreses governamentals busquen tenir accés total a informació privada i personal (PII per les seves sigles en anglès). L'increment en l'ús de serveis al núvol públics fa que les organitzacions tinguin dificultats per a protegir les dades que els seus usuaris els han confiat a la vegada que compleixen totes les lleis pertinents. La Unió Europea té algunes de les lleis sobre privacitat més progressistes i tots els principals proveïdors de serveis al núvol (Amazon, Google i Microsfot) tenen diversos centres de dades i regions a la Unió Europea. Així doncs, recomanem a les companyies, especialment les que tinguin una base d'usuaris mundial, que assegurin un refugi segur per a les dades dels seus usuaris **posant els seus servidors a la UE**. Des de que vam escriure sobre aquesta tècnica a l'últim Radar hem desenvolupat un nou sistema intern de tractament de dades sensibles dels nostres empleats i hem decidit guardar-les en un centre de dades situat a la Unió Europea.

ADOPT
1. Consumer-driven contract testing
2. Pipelines as code
3. Threat Modeling

TRIAL
4. APIs as a product
5. Bug bounties
6. Data Lake
7. Hosting PII data in the EU
8. Lightweight Architecture Decision Records
9. Reactive architectures
10. Serverless architecture

ASSESS
11. Client-directed query
12. Container security scanning
13. Content Security Policies
14. Differential privacy
15. Micro frontends
16. OWASP ASVS
17. Unikernels
18. VR beyond gaming

HOLD
19. A single CI instance for all teams
20. Anemic REST
21. Big Data envy
22. Cloud lift and shift

© November 2016, ThoughtWorks, Inc. All Rights Reserved.

TÈCNIQUES *continuació*

Tot i que molta documentació es pot reemplaçar amb codis i proves fàcils de llegir, en el món de l'arquitectura evolutiva és important tenir en compte algunes decisions de disseny que beneficiïn futurs membres de l'equip i la supervisió externa. El **registre de decisions d'arquitectura de baix pes** és una tècnica per la qual es capturen decisions arquitecturals importants juntament amb el seu context i les seves conseqüències. Tot i que normalment aquests elements es troben en una wiki o una eina col•laborativa, nosaltres preferim afegir-los al control de versions amb un simple comentari.

L'**arquitectura sense servidor** és un enfocament pel qual es canvien les màquines virtuals en funcionament constant per un mode de processament efímer que apareix quan se'l demana i desapareix tant bon punt s'ha deixat d'utilitzar. Des de l'últim Radar hem tingut diversos equips que han decidit crear aplicacions utilitzant aquest estil "sense servidor". Als nostres equips els atrau la idea: funciona bé i considerem que és una opció d'arquitectura vàlida. S'ha de tenir en compte, a més, que aquest enfocament no té perquè ser blanc o negre: alguns dels nostres equips han desplegat alguns elements del seu sistema sense servidor mentre que continuen utilitzant una arquitectura més tradicional per altres parts.

Tot i que molts dels problemes que la gent experimenta amb les APIs quan les enfoca des del servei RESTful es poden atribuir a anti-patrons REST anèmics, alguns casos justifiquen l'exploració d'altres enfocaments. Poden arribar al límit de l'arquitectura RESTful, sobretot, organitzacions que han de mantenir un gran nombre d'aplicacions de clients (per la qual cosa poden tenir una proliferació de versions de APIs fins i tot si utilitzen contractes guiats pels consumidors) i que tenen una gran part de les seves API donant suport a un llistat interminable d'estils de canals d'activitat. Aquests problemes es poden mitigar, a vegades, enfocant la **interacció client-servidor** des del punt de vista de la recerca guiada pel client. Hem vist com aquest enfocament s'ha aplicat satisfactòriament tant a GraphQL com a Falcor, on els clients tenen més control sobre el contingut i la definició de la informació que se'ls retorna. Això pot posar més responsabilitat sobre la capa de servei i és possible que porti a un alt grau d'acoblament al model de dades subjacent però els beneficis poden ser prou grans com per explorar aquesta opció si no funcionen bons models de APIs RESTful.

La revolució del contenidor instigada per Docker ha reduït enormement els problemes de moure una aplicació a d'altres entorns però, a la vegada, s'ha malmès considerablement els controls tradicionals sobre el que es pot dur a terme. La tècnica d'**escaneig de seguretat de contenidors** és una resposta necessària per aquest vector d'amenaça. Docker inclou, actualment, les seves pròpies eines d'escaneig de seguretat, tal com fa CoreOS, i també hem tingut èxit amb els benchmarks de seguretat de CIS. Creiem que, sigui quin sigui l'enfocament escollit, el tema de la validació automatitzada de seguretat de contenidor és molt important i necessària en el pensament PaaS.

Fa temps que se sap que les col•leccions de dades "anònimes" poden revelar informació sobre individus, especialment si es tracta diferents col•leccions de dades a la vegada. Amb l'augment de la preocupació per la privacitat, algunes companyies, incloent Apple i Google, estan començant a utilitzar tècniques de privacitat diferencial per a millorar la privacitat a la vegada que conserven la capacitat d'analitzar grans quantitats de dades d'usuaris. **La privacitat diferencial** és una tècnica criptogràfica que intenta maximitzar la precisió de les recerques estadístiques des d'una base de dades tot minimitzant les possibilitats d'identificar les seves entrades. Aquests resultats es poden aconseguir introduint poc "soroll" a les dades però és important apuntar que es tracta d'una àrea que encara s'està investigant. Apple ha anunciat els seus plans per incorporar la privacitat diferencial en els seus productes (i podem dir que aplaudim el seu compromís amb la privacitat dels seus consumidors) però el secretisme típic de Apple ha fet que alguns experts en seguretat estiguin una mica confosos. Nosaltres seguim recomanant l'alternativa Datensparsamkeit: emmagatzemar el mínim de dades possible reportarà, en la majoria de casos, els millors resultats en privacitat.

Hem vist que la introducció d'arquitectures de microserveis té grans beneficis, la qual cosa ha permès a alguns equips d'escalonar l'entrega de serveis desplegats i mantinguts independentment. No obstant això, els equips sovint tenen problemes per evitar la creació d'interfícies monolítiques, és a dir, enormes aplicacions web que són tan difícils de mantenir i evolucionar com les aplicacions de servidor que hem abandonat. El que estem veient aparèixer és un enfocament que els nostres equips anomenen **micro-interfícies**. D'aquesta manera, una aplicació web es divideix tenint en compte

© November 2016, ThoughtWorks, Inc. All Rights Reserved.

TÈCNIQUES *continuació*

les seves pàgines i característiques i cada una d'aquestes divisions és responsabilitat d'un sol equip. Hi ha diverses tècniques (algunes noves i d'altres velles) per aconseguir que les característiques de l'aplicació es complementin per crear una millor experiència d'usuari però l'objectiu segueix sent el de permetre que es pugui desenvolupar, provar i desplegar cada característica independentment de les altres. L'enfocament de motor d'interfície (BFF per les seves sigles en anglès) funciona molt bé aquí on cada equip desenvolupa un BFF per a donar suport a la seva col•lecció de característiques de l'aplicació.

Amb l'augment de la popularitat del patró de motor d'interfície (BFF) i l'ús d'entorns de treball d'associació de dades com React.js, hem vist una resposta negativa cap a les arquitectures REST. Els crítics acusen REST de crear interaccions ineficients entre sistemes i de no adaptar-se a l'evolució de les necessitats dels clients. Proposen com a mecanismes d'obtenció de dades alternatius entorns de treball com GraphQL o Falcor, ja que permeten que el client especifiqui el format de les dades retornades. En la nostra experiència, però, no és REST el que causa aquests problemes sinó que més aviat apareixen perquè no s'ha sabut modelar el domini com una col•lecció de recursos. Desenvolupar nativament serveis que només exposen models de dades estàtics i jeràrquics a través de URLs basades en plantilles té com a resultat una **implementació anèmica de REST**. En un domini amb un model ric, el REST hauria de permetre més que una simple recol•lecció de dades repetitiva. En una arquitectura RESTful totalment evolucionada, els esdeveniments d'empresa i els conceptes abstractes es modelen també com a recursos i la implementació hauria d'utilitzar correctament l'hipertext, la relació d'enllaços i els tipus de contingut multimèdia per a maximitzar l'acoblament entre els serveis. Aquest anti-patró està estretament relacionat amb el patró de Model de Domini Anèmic i té com a resultat serveis que ocupen llocs baixos en el Model de Maduresa de Richardson. Disposem de més recomanacions sobre el disseny de APIs de RESt efectives en el nostre article sobre troballes.

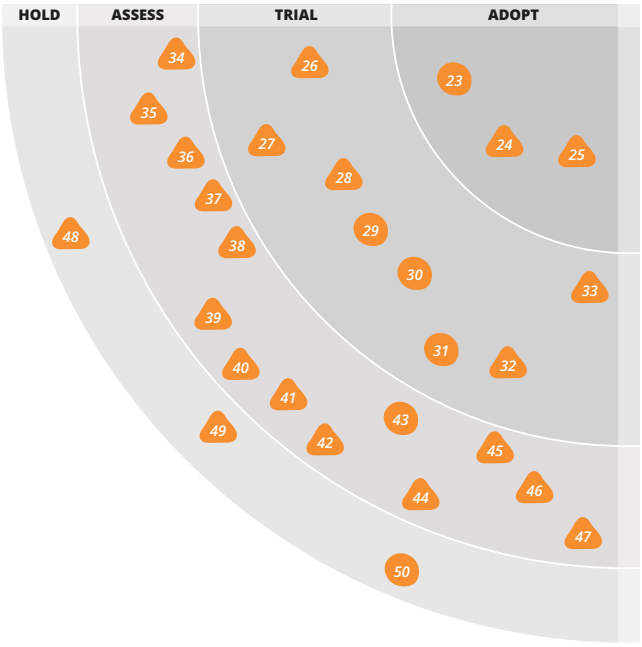
Seguim veient com organitzacions busquen tecnologies "cool" les quals els fan prendre riscos i complicar-se la vida quan existeix una opció més simple. Un tema en concret és el de l'ús de sistemes distribuïts de dades massives per a col•leccions de dades relativament petites. Aquest comportament, juntament amb noves dades extretes d'experiències recents, ens porta, una vegada més, a evitar aquest tipus de **sistemes de dades massives**. La base de dades Apache Cassandra promet una escalabilitat enorme en sistemes de consum però hem vist a molts equips aclaparats per la seva complexitat arquitectural i operacional. A menys que es tinguin volums de dades que requereixin un clúster de 100 o més nodes, no recomanem l'ús de Cassandra. No val la pena si es té en compte l'equip necessari per a fer-lo funcionar. A l'hora de crear aquesta edició del Radar vam parlar sobre diverses tecnologies de bases de dades, moltes de les quals ofereixen un rendiment "10 cops" millor que els sistemes existents. Sempre som escèptics davant noves tecnologies, especialment amb tecnologies tan importants com les bases de dades, fins que no han estat degudament provades. Jepsen proporciona anàlisis del rendiment de base de dades sota condicions difícils i ha trobat molts errors a diverses bases de dades NoSQL. Recomanem mantenir una dosi adequada d'escepticisme i donar un cop d'ull a pàgines com Jepsen quan s'avaluï sistemes de bases de dades.

A mida que cada vegada més organitzacions decideixen desplegar aplicacions al núvol, estem veient regularment grups informàtics que intenten replicar al núvol, amb grans esforços inútils, els seus enfocaments sobre el tractament de dades i la seguretat. Això ho podem trobar, sovint, en forma de tallafocs, administradors de càrregues, xarxes de proxys, control d'accés, eines de seguretat i serveis que es transporten al núvol sense pensar. Hem vist organitzacions construir les seves pròpies APIs d'orquestració davant dels proveïdors de serveis al núvol per a limitar els serveis que els equips poden utilitzar. En la majoria de casos, aquestes capes només serveixen per reduir el potencial, eliminant la majoria dels beneficis de canviar-se al núvol. En aquesta edició del Radar, hem decidit tornar a destacar el **cloud lift and shift** com una tècnica a evitar. En lloc d'això, les organitzacions haurien de revisar detingudament l'objectiu dels seus controls de seguretat i operacionals existents i buscar controls que funcionin al núvol sense crear limitacions innecessàries. Molts d'aquest controls ja es poden trobar a proveïdors madurs de serveis al núvol i els equips que adoptin el núvol poden utilitzar APIs natives per al seu propi aprovisionament i les seves operacions.

© November 2016, ThoughtWorks, Inc. All Rights Reserved.

PLATAFORMES

Actualment, la seguretat de transport HTTP estricta (**HSTS**) és una política vastament suportada que permet que les pàgines web es protegeixin contra atacs degradants. Un atac degradant, en el context de HTTPS, és el que comporta que els usuaris del servei acabin a HTTP en lloc de HTTPS, el que permet nous atacs com ara els atacs man-in-the-middle. Amb l'HSTS, el servidor informa al navegador, per mitjà d'una capçalera, que només hauria d'utilitzar HTTPS per accedir a la pàgina web. El suport per part dels navegadors ja és prou generalitzat com per què aquesta funció de fàcil implementació s'afegeixi a qualsevol pàgina web que utilitzi HTTPS. L'Observatori de Mozilla pot ajudar a identificar aquesta i d'altres capçaleres útils i opcions de configuració que millorin la privacitat i la seguretat. Quan s'implementi HSTS, és de vital importància verificar que tots els recursos es carreguen correctament en HTTPS ja que un cop s'ha activat el HSTS ja no es pot tornar enrere, gairebé, fins que no caduqui. Seria necessari afegir la instrucció per a incloure els subdominis però, una vegada més, és necessari assegurar-se de que els subdominis accepten el transport segur.



Ha quedat demostrat que la llista blanca d'aplicacions és una de les maneres més efectives de mitigar els atacs cibernètics intrusius. Una manera convenient d'implementar aquesta pràctica altament recomanada és a través dels **mòduls de seguretat de Linux** (LSM). Amb mòduls com SELinux i AppArmor inclosos per defecte en la majoria de distribucions de Linux i amb altres eines més completes com Grsecurity a l'abast, hem decidit moure aquesta tecnologia a l'anell d'Adopta per a aquesta edició. Aquestes eines permeten als equips d'avaluar qui té accés a quins recursos en els servidors compartits, incloent els serveis continguts. Aquest enfocament més conservador de gestió ajudarà als equips a incorporar la seguretat en els seus processos SDLC.

Seguim tenint experiències molt positives desplegant la plataforma **Apache Mesos** per a gestionar recursos de clúster per a sistemes altament distribuïts. Mesos abstreu recursos computacionals subjacents com la CPU i l'emmagatzematge amb l'objectiu de proporcionar un ús eficient mentre es manté l'aïllament. Mesos inclou Chronos per a l'execució distribuïda i a prova de fallades de tasques programades i Marathon per a orquestrar processos de llarga durada a contenidors.

Estem cada vegada més convençuts de que en la majoria de casos no val la pena escriure el seu propi codi d'autenticació. La gestió de la identitat feta externament accelera l'entrega, redueix els errors i tendeix a permetre una resposta més ràpida contra noves vulnerabilitats. Ens ha impressionat especialment **Auth0** per la seva facilitat

ADOPT
23. Docker
24. HSTS
25. Linux security modules

TRIAL
26. Apache Mesos
27. Auth0
28. AWS Lambda
29. Kubernetes
30. Pivotal Cloud Foundry
31. Rancher
32. Realm
33. Unity beyond gaming

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

HOLD
48. CMS as a platform
49. Overambitious API gateway
50. Superficial private cloud

© November 2016, ThoughtWorks, Inc. All Rights Reserved.

PLATAFORMES *continuació*

d'integració, la varietat de protocols i connectors que suporta i la gran API de gestió.

Els nostres equips segueixen gaudint de **AWS Lambda** i l'estan començant a utilitzar per a experimentar amb arquitectures sense servidor, combinant Lambda amb API Gateway. Recomanem que les funcions de Lambda continguin, només, una quantitat moderada de codi. Assegurar la qualitat d'una solució basada en una barreja de moltes i extenses funcions Lambda és difícil i pot no ser rendible. Seguim preferint, per a necessitats més complexes, el desplegament basat en contenidors o Màquines Virtuals. Hem tingut, a més, problemes significatius utilitzant Java per a funcions Lambda amb latències erràtiques de fins a diversos segons des de que s'activa el contenidor Lambda. Òbviament, però, és possible esquivar aquest problema utilitzant JavaScript o Python i, si la funció Lambda no conté massa codi, l'elecció del llenguatge de programació no hauria de ser massa important.

Realm és una base de dades dissenyada per a ser utilitzada en dispositius mòbils i que compta amb el seu propi motor de persistència per aconseguir un millor rendiment. Realm es presenta com un recanvi per a SQLite i Core Data. S'ha de tenir en compta, però, que la migració no és tan senzilla com la documentació de Realm ens vol fer creure. De tota manera, però, cada vegada més equips escullen Realm com a mecanisme de persistència en els entorns de producció per a aplicacions mòbils.

Després de créixer durant anys com a plataforma per al desenvolupament de jocs, **Unity** s'ha convertit ara en la plataforma preferida per al desenvolupament d'aplicacions de RV i RA. Ja s'estigui creant un món totalment immersiu per a l'Oculus o l'HTC Vive, una nova capa hologràfica per a una nova aplicació espacial o una funció de realitat augmentada per a una aplicació mòbil, és molt possible que Unity proporcioni tot el necessari per, no només fer el prototip, sinó també el producte final. A ThoughtWorks, molts creiem que RV i RA representa el següent canvi significatiu en el món de la informàtica i, per ara, Unity és l'eina més important per al desenvolupament d'aquest canvi. Hem utilitzat Unity per a desenvolupar tots els nostres prototips de RV i les funcions de RA d'aplicacions per a cascos de RV i mòbils/tauletes.

.NET Core és un producte modular de codi obert per crear aplicacions que es poden desplegar fàcilment tant a Windows com a macOS i Linux. .NET Core possibilita crear aplicacions web que funcionen en diversos sistemes utilitzant ASP.NET Core amb una sèrie d'eines, llibreries i entorns de treball, una altra opció per a l'arquitectura de microserveis. La comunitat al voltant de .NET Core i d'altres projectes no ha deixat de créixer i han aparegut i evolucionat noves eines com Visual Studio Code. Existeixen imatges de Docker basades tant en Linux com en Windows (Nano Server) amb .NET Core que simplifiquen l'aplicació d'una arquitectura de microserveis. CoreCLR i CoreFX ja van aparèixer al Radar anteriorment però fa uns mesos Microsoft va anunciar el llançament de .NET Core 1.0, la primera versió estable. Considerem que les oportunitats, els canvis i la vibrant comunitat són bones raons per a seguir avaluant aquest producte.

Amazon API Gateway és la oferta de Amazon per permetre que els desenvolupadors puguin exposar serveis API a clients d'internet. Ofereix les característiques típiques del API Gateway com la gestió del tràfic, la monitorització, l'autenticació i l'autorització. Els nostres equips han utilitzat aquest servei per donar protagonisme a altres AWS com AWS Lambda en arquitectures sense servidor. Ho seguim supervisant per minimitzar els reptes que comporten API gateways massa ambicioses però, en aquest punt, sembla que la oferta de Amazon és prou lleugera com per evitar aquests contratemps.

Segueix creixent l'interès per **Apache Flink**, una plataforma d'última generació per al processament distribuït i escalable de lots i transmissió. Al cor de Apache Flink hi trobem un motor de transmissió de dades en temps real que suporta operacions tabulars (de l'estil SQL), de processament de gràfics i d'aprenentatge automàtic. Apache Flink compta amb un gran nombre de funcions per al processament de transmissions en temps real: temps de l'esdeveniment, operacions sobre la finestra de transmissió, és a prova de fallades i utilitza la semàntica "exactament una vegada". El projecte té molta activitat i la última versió (1.1) introdueix noves integracions de fonts i característiques de transmissió millorades.

Amazon ha llançat recentment el **AWS Application Load Balancer** (ALB), un recanvi per als administradors de càrregues elàstics introduïts l'any 2009. ALB suporta la inspecció del trànsit a la capa 7 i s'ha construït per suportar arquitectures modernes al núvol. Si es construeix un sistema basat en microserveis utilitzant ECS, els nous administradors de càrregues entendran

© November 2016, ThoughtWorks, Inc. All Rights Reserved.

PLATAFORMES *continuació*

sense problemes l'allotjament i l'escalatge en contenidors, amb múltiples contenidors i ports per a cada instància EC2. L'encaminament basat en el contingut permet la segmentació de les peticions en grups de servidors objectiu, juntament amb l'escalatge independent d'aquells grups. Les proves fetes pels administradors de càrregues són molt millors i tenen l'habilitat de capturar mètriques detallades sobre el rendiment de l'aplicació. Ens agrada molt tot el que veiem i ja tenim equips que reporten tenir èxit utilitzant ALB.

La base de dades Cassandra, de Apache, és una solució de Big Data potent i escalable per a guardar i processar grans quantitats de dades utilitzant, sovint, centenars de nodes dispersats per tot el món. És una molt bona eina i ens agrada però veiem massa sovint equips amb problemes utilitzant-la. Nosaltres recomanen utilitzar **Cassandra amb compte**. Molts equips no entenen la utilitat de Cassandra i intenten utilitzar-la com una base de dades normal quan en realitat està optimitzada per a lectures ràpides dintre de grans col•leccions de dades basades en claus o índexs predefinits. La seva independència respecte de l'esquema d'emmagatzematge també pot dificultar la seva evolució. Cassandra és, a més, molt complexa d'operar i té alguns punts que cal polir per la qual cosa, a menys que sigui de vital importància l'escalatge que proporciona, és millor optar per una solució més simple. Si no es necessiten les característiques específiques que aporta és molt possible que s'estigui escollint per cobdícia pel Big Data. Una utilització adequada de Cassandra inclourà extenses proves automatitzades i, per això, recomanem amb gust que s'utilitzi CassandraUnit com a part de l'estratègia de proves.

Electron és un entorn de treball sòlid per construir clients natius per a ordinadors utilitzant tecnologies web com HTML, CSS i JavaScript. Els equips poden utilitzar el seu coneixement existent per crear clients multi-plataforma sense haver de perdre el temps aprenent noves tecnologies.

L'excitació per les cadenes de blocs i la moneda digital sembla que ha arribat al seu màxim com es pot veure pel descens en els anuncis sobre aquest tema i anticipem que alguns dels esforços més especulatius acabaran morint. **Ethereum**, una de les cadenes de blocs, està progressant positivament i és digne de veure. Ethereum és una cadena de blocs pública i creada amb un llenguatge de programació que permet que s'hi construeixin "contractes intel•ligents". Són moviments algorítmics de "ether" (la moneda digital de Ethereum) com a resposta a l'activitat que té lloc a la cadena de blocs. R3Cev, el consorci que construeix la tecnologia de cadena de blocs pels bancs, va crear a Ethereum la seva primera prova de concepte. Ethereum s'ha utilitzat per crear una Organització Autònoma Descentralitzada (DAO per les seves sigles en anglès), una de les primeres "corporacions algorítmiques", però un robatori de Ether per valor de 150 milions de dólars demostra que la cadena de blocs segueix sent el "Far West" del món de la tecnologia.

Amb **HoloLens**, Microsoft ha creat el primer casc de RA realment utilitzable. No només té un disseny industrial molt maco i és un dispositiu còmode de portar sinó que, a més, demostra clarament el potencial de la RA per a l'empresa gràcies a la seva imatge i a la profunda integració amb Windows 10. Creiem que HoloLens serà la primera plataforma de RA en la qual podrem oferir grans funcionalitats per aplicacions per als nostres clients i desitgem veure com evoluciona quan rebi més atenció.

IndiaStack és una col•lecció de APIs obertes dissenyades amb l'objectiu de fer que la Índia passi de ser un país pobre en dades a un de ric. Aquestes APIs emfatitzen la innovació per capes especificant una col•lecció mínima de APIs i anima a la resta de l'ecosistema a construir aplicacions a sobre d'aquestes APIs. Aadhaar funciona com a una de les capes base i proporciona serveis d'autenticació per a més de mil milions de ciutadans indis. A més d'això, hi ha serveis per proporcionar transaccions sense papers gràcies a firmes digitals (eSign), pagament online unificat (UPI per les sigles en anglès) i una capa de consentiment electrònic (e-KYC) per proporcionar amb seguretat els detalls de Aadhaar als proveïdors de serveis. Nosaltres creiem en aquest tipus d'iniciativa per portar la innovació digital i que els principis de disseny que hi ha al darrera de IndiaStack es podrien utilitzar per portar un canvi a altres regions o països.

Nuance Mix és un entorn de treball per al processament de llenguatge natural creat per la companyia que va crear la primera tecnologia de veu a text (speech-to-text) darrera Dragon Speaking i la primera versió de Siri. Aquest entorn de treball permet la creació de gramàtiques que permeten la interacció improvisada de l'usuari per mitjà de la veu. El desenvolupador defineix una gramàtica per un camp específic que l'entorn de treball pot acabar entenent. El resultat són respostes que identifiquen els conceptes d'intenció i d'interacció de l'usuari. Al principi, està limitat a frases properes a les que s'han utilitzat per a entrenar-lo però, amb el temps, pot entendre frases construïdes diferentment. Tot i que encara està en fase beta, la precisió que hem trobat és molt interessant i el producte que se'n podria crear podria tenir aplicacions

© November 2016, ThoughtWorks, Inc. All Rights Reserved.

PLATAFORMES *continuació*

que podrien beneficiar-se d'interaccions sense mans amb l'usuari, incloent mòbils, IoT, RA, RV i espais interactius.

OpenVR és el SDK subjacent que fa que moltes pantalles per RV (HMD per les seves sigles en anglès) funcionin amb Unity i és molt probable que sigui cada cop més important. Molta de la feina feta en RV a ThoughtWorks s'ha creat sobre OpenVR ja que funciona en qualsevol HMD a diferència dels altres SDKs. Tot i que no és de codi obert, és gratuït amb una llicència. El SDK de l'Oculus té un sistema de llicències més restrictiu i només funciona en dispositius Oculus. Tot i que OSVR és realment de codi obert no sembla, de moment, que tingui tanta acceptació. Si es vol desenvolupar una aplicació de RV sense utilitzar ni Unity ni Unreal i amb l'objectiu de que funcioni al màxim de dispositius possibles, OpenVR és, ara mateix, la millor solució.

Tarantool és una solució NoSQL de codi obert que combina la base de dades i la cache en una sola entitat i proporciona APIs per escriure la lògica d'aplicació a Lua. Suporta tant motors en memòria com en disc i els usuaris poden crear múltiples índexs (HASH, TREE, RTREE, BITSET) basant-se en el seu ús. Les dades s'emmagatzemen en format MessagePack i utilitzen el mateix protocol per comunicar-se entre clients i servidor. Tarantool suporta registres d'escriptura avançada, transaccions i replicació master-master asincrònica. Estem d'acord amb la decisió arquitectural d'utilitzar la política d'un sol escriptor i la multitasca cooperativa per a gestionar connexions simultànies.

L'excitació per l'aprenentatge automàtic no para de créixer però, com amb el Big Data, encara s'han de descobrir i crear entorns de treball i eines realment útils. Una d'aquestes eines és **wit.ai**, una plataforma SaaS que permet als desenvolupadors de crear interfícies conversacionals utilitzant el processament de llenguatge natural. Wit funciona amb entrades en format de text o de veu, ajuda als desenvolupadors a gestionar la intenció conversacional i permet que s'implementi una lògica de negoci personalitzada utilitzant JavaScript. El sistema és gratuït per a ús comercial i no comercial i fomenta la creació d'aplicacions de codi obert. S'ha de tenir en compte, però, que cal acceptar que Wit utilitzi les nostres dades per a millorar el servei i per les seves propies anàlisis així que recomanem llegir detingudament els seus termes i condicions. Un altre aspirant en aquest espai és el Microsoft Bot Framework però només està disponible en format de vista prèvia en el moment d'escriure això. Com passa amb tot el relacionat amb Microsoft, però, esperem que aquest Bot evolucioni molt ràpidament pel que creiem que val la pena estar-ne pendent.

Estem veient moltes organitzacions amb problemes quan intenten utilitzar el seu **Sistema de Gestió de Continguts** (CMS per les sigles en anglès) **com a plataforma** per a compartir grans i complicades aplicacions digitals. Això acostuma a passar quan s'intenta evitar males organitzacions TIC i deixant que l'empresa ho passi tot a producció. Tot i que estem a favor de proporcionar als productors de contingut les eines i els flux de treball adequats, per aplicacions amb una lògica de negoci complexa recomanaríem tractar el sistema de gestió de contingut com un component de la plataforma (en mode híbrid o headless) que coopera amb altres serveis, en lloc d'intentar implementar tota la funcionalitat en el mateix sistema.

Una de les nostres queixes més comunes va dirigida a la implementació d'intel·ligència de negoci en el programari intermediari amb la intenció de fer funcionar la lògica crítica d'una aplicació. Els fabricants del competitiu mercat de les API Gateway no deixen d'afegir funcions que diferencien els seus productes. El resultat són **productes massa ambiciosos** que, a part de ser, essencialment, un servidor intermediari invers, convida a utilitzar dissenys difícils de provar i desplegar. Les API gateways poden ser útils per a solucionar alguns problemes genèrics, com ara l'autenticació o la limitació de flux, però coses com la transformació de dades o el processament de regles haurien de ser a aplicacions o serveis on poden ser controlades pels equips treballant estretament amb els dominis que suporten.

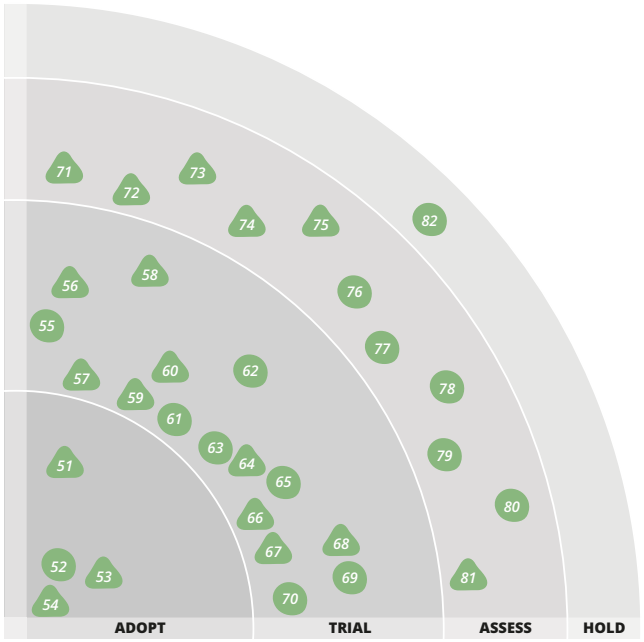
© November 2016, ThoughtWorks, Inc. All Rights Reserved.

EINES

Babel.js s'ha convertit en el compilador per defecte per escriure JavaScript d'última generació. El seu ecosistema s'està desenvolupament gràcies al seu sistema de connectors reestructurat. Permet que els desenvolupadors escriguin codi ES6 (i fins i tot ES7) que corre al navegador o al servidor sense sacrificar la retrocompatibilitat amb navegadors antics i pel qual cal molt poca configuració. Té un suport de primer nivell per a sistemes de desenvolupament i prova per la qual cosa es pot incorporar sense problemes a processos de treball ja existents. És un gran programari que ha facilitat l'adopció i la innovació de ES6 (i ES7).

Quan es combinen tècniques i estils d'arquitectura moderns, com els microservices, DevOps i el control de qualitat durant el desenvolupament, els equips de desenvolupament necessiten eines de supervisió cada vegada més sofisticades. Ja no és suficient mirar un gràfic sobre l'ús de la CPU i del disc. És per això que molts equips utilitzen eines com Graphite o Kibana per a recol•lectar mètriques sobre aplicacions i empreses. **Grafana** facilita la creació de panells d'instruments útils i elegants per a dades obtingudes de diverses fonts. Existeix una funció especialment útil que permet sincronitzar els temps de diferents gràfics per identificar correlacions en les dades. El sistema de plantilles que s'ha afegit té molt potencial i farà que la gestió de serveis similars sigui encara més fàcil. Grafana s'ha convertit en la nostra elecció en aquesta categoria gràcies a les seves funcions.

Les imatges de màquina s'estan convertint en un canal de desplegament bàsic i existeixen diverses eines i tècniques per crear-les. Nosaltres recomanem **Packer** abans que d'altres alternatives per la gran quantitat de funcions i les experiències positives que hem tingut. També recomanem no escriure scripts propis per fer el que Packer pot fer per defecte.



Els nostres equips estan utilitzant, cada vegada més, **Espresso** com a eina de proves en el desenvolupament d'aplicacions de Android. La seva API de mida reduïda oculta els detalls confusos de la implementació i ajuda a escriure proves concises i amb una execució de les proves més ràpida i fiable. Amb Espresso es poden dur a terme proves automatitzades sobre la interfície d'usuari que simulen la interacció de l'usuari sobre una aplicació en concret tant a emuladors com a dispositius reals i a diferents versions de Android.

fastlane és la nostra eina predilecte per automatitzar la majoria de les activitats necessàries per crear, provar, documentar i subministrar aplicacions mòbils per a iOS i Android. Una configuració simple, una gran varietat

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

TRIAL
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

ASSESS
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

HOLD
82. Jenkins as a deployment pipeline

© November 2016, ThoughtWorks, Inc. All Rights Reserved.

EINES *continuació*

d'eines i múltiples canals el fan un ingredient clau per a l'entrega continua per a mòbils.

Provar que la disposició i l'estil de pàgines web funciona correctament en diferents mides pot ser un procés lent i, sovint, manual. **Galen** ajuda a facilitar aquest procés gràcies a un llenguatge simple, que corre a sobre de Selenium, que permet especificar les expectatives sobre l'aparença de la pàgina web a pantalles de diverses mides. Tot i que Galen té els mateixos problemes de fragilitat i de velocitat típics de qualsevol altre mètode de proves continues, trobem que els primers comentaris sobre disseny són molt beneficiosos.

El fet de poder gestionar els secrets de manera segura s'està convertint en un problema cada cop més gran pels projectes. L'antiga pràctica de guardar els secrets en un arxiu o en variables d'entorn s'està fent cada cop més difícil de gestionar, especialment en entorns amb múltiples aplicacions i un gran nombre de microserveis. **HashiCorp Vault** soluciona aquest problema proporcionant mecanismes per accedir als secrets a través d'una interfícies unificada. Ens ha sigut de gran ajuda en un bon nombre de projectes i als nostres equips els ha agradat com n'era de fàcil d'integrar Vault als seus serveis. Emmagatzemar i actualitzar secrets és una mica incòmode de fer ja que depèn d'una eina de línia d'ordres i una gran quantitat de disciplina per part de l'equip.

Hi ha cada vegada més projectes que emeten i consumeixen informació formatada en JSON. Escriure proves en Java per a JSON pot ser laboriós. **JSONassert** és una petita llibreria creada per ajudar a escriure proves més petites que s'encarreguin de JSON simplificant assercions i proporcionant millors missatges d'error.

Pa11y és un provador d'accessibilitat automàtic que pot córrer des de la línia de comandaments i es pot incrustar en un canal. Els nostres equips han tingut èxit utilitzant Pa11y en una pàgina molt dinàmica creant, en primer lloc, una versió estàtica en HTML i, després, passant les proves d'accessibilitat. Per a molts sistemes, especialment pàgines governamentals, és necessari córrer proves d'accessibilitat i Pa11y ho fa molt més fàcil.

Tenint eines tan bones com Vault, no hi ha excusa que valgui per guardar secrets en els repositoris de codi, especialment quan aquests acostumen a ser el punt feble de sistemes importants. Ja hem mencionat anteriorment eines d'escaneig de repositoris com Gitrob però ara estem impulsant eines proactives com **Talisman** (creada per ThoughtWorks). Es tracta d'un ganxo pre-push per Git que escaneja els commits en busca de secrets que coincideixin amb patrons predefinits.

Amb **Terraform** és possible gestionar infraestructures al núvol escrivint definicions declaratives. La configuració dels servidors exemplificada per Terraform normalment es deixa a eines com Puppet, Chef o Ansible. Ens agrada, però, Terraform perquè la sintaxi dels seus arxius és bastant fàcil de llegir i perquè suporta diversos proveïdors de serveis al núvol sense intentar proporcionar una abstracció artificial entre aquests proveïdors. El desenvolupament i l'evolució de Terraform des de la nostra primera, i més cauta, menció de ara fa gairebé dos anys l'ha convertit en un producte estable i que ens ha sigut de gran ajuda en molts projectes. Actualment, el problema de gestió dels arxius d'estat es pot evitar utilitzant el que Terraform anomena "remote state backend" i hem sigut capaços d'utilitzar Consul per aquesta finalitat.

La programació en parella és una tècnica essencial per a nosaltres i, com que cada vegada veiem més equips on els membres es troben a diferents llocs, hem provat diverses eines que permeten treballar remotament. Ens ha agradat ScreenHero però ens preocupa el seu futur. Hem vist que tmate és una bona solució per equips que no depenen de IDE. **tmate** es basa en la popular eina tmux però, a diferència de tmux, la configuració per treballar remotament és molt més senzilla. En comparació amb solucions per compartir pantalla, l'amplada de banda i els recursos necessaris són modestos i, com és d'esperar, no té mai el problema de pantalles borroses. Els equips, a més, poden establir el seu propi servidor per tenir el control total sobre la privacitat i la integritat de la solució.

Android-x86 és la migració del projecte Android open source a plataformes x86. El projecte va començar allotjant pedaços per x86 creats per la comunitat però després va crear la seva pròpia base de codi per donar suport a diverses plataformes x86. Hem estalviat una gran quantitat de temps en proves hermètiques de la interfície d'usuari utilitzant Android-x86 en lloc d'emuladors en els nostres servidors d'integració contínua. No obstant això, és millor seguir utilitzant emuladors per a proves on es busquen certes característiques com poca memòria,

© November 2016, ThoughtWorks, Inc. All Rights Reserved.

poca amplada de banda o poca bateria.

Els nostres equips han tingut èxit amb **axios**, un client HTTP basat en peticions a JavaScript, que descriuen com "millor que Fetch." El projecte té molts seguidors i és molt actiu a GitHub i, a més, té el nostre vist i plau.

Amb l'augment en l'interès per arquitectures de transmissió de dades en temps real i els oceans de dades que alimenten hem vist un increment en la dependència d'eines de "captura de canvi de dades" per a connectar dipòsits transaccionals de dades a sistemes de processament de transmissions. **Bottled Water** és una incorporació molt benvinguda a aquest camp ja que converteix els canvis al registre d'escriptura anticipada de PostgreSQL en esdeveniments Kafka. Un punt negatiu, però, és que ens lliga a esdeveniments de base de dades de baix nivell en lloc dels esdeveniments de negoci de nivell més alt que recomanem que s'utilitzin com a base per a les arquitectures basades en esdeveniments.

Un dels debats recurrents entre desenvolupadors tracta sobre l'escriptura de llenguatge: quant és adequat? Clojure, el Lisp escrit dinàmicament en la màquina virtual de Java (JVM per les seves sigles en anglès) va introduir un nou punt en aquest debat que va difuminar les línies. **Clojure.spec** és un nou servei inclòs a Clojure que permet als desenvolupadors d'embolicar criteris de verificació a estructures de dades com rangs de valors admissibles. Un cop s'han establert, Clojure utilitza aquestes especificacions per a proporcionar un munt de beneficis: proves generades, validació i desestructuració d'estructures de dades entre d'altres. Clojure.spec és una manera prometedora d'aconseguir els beneficis de tipus i rangs allà on es volen i no a tot arreu.

Posar a prova la part visual d'aplicacions iOS pot ser pesat, lent i tediós. És per això que ens alegra afegir **FBSnapshotTestcase** a la nostra caixa d'eines. Automatitza la feina de fer, emmagatzemar i comparar captures de components de la interfície d'usuari per tenir interfícies sense ni un píxel fora de lloc. Com que funciona com a unitat de prova al simulador, és més ràpida i més fiable que les proves funcionals.

Scikit-learn és una llibreria d'aprenentatge automàtic escrita en Python cada cop més popular. Proporciona una bona col•lecció de models d'aprenentatge automàtic com l'agrupament, la classificació, la regressió i reducció dimensional i un gran nombre de funcions per tasques relacionades com la selecció i l'avaluació del model i la preparació de dades. Com que s'ha dissenyat per ser senzilla, reutilitzable en diversos contextos i ben documentada, creiem que aquesta eina pot ser utilitzada, fins i tot, per què no experts puguin explorar el món de l'aprenentatge automàtic.

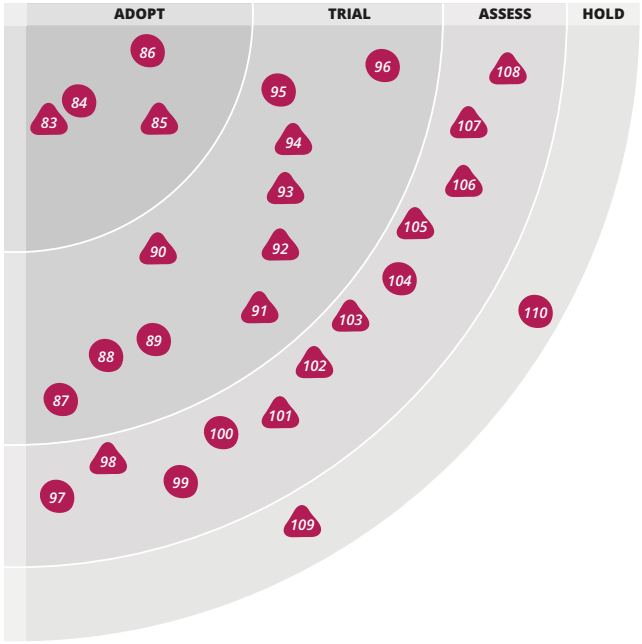
© November 2016, ThoughtWorks, Inc. All Rights Reserved.

LLENGUATGES & ENTORNS DE TREBALL

Si s'ha de crear una aplicació de pàgina única (SPA per les seves sigles en anglès) i no se sap quin entorn de treball utilitzar, **Ember.js** s'està convertint en la opció més destacada. Als nostres equips els ha agradat molt la gran productivitat de desenvolupament que permet i la menor quantitat de sorpreses en comparació amb altres entorns de treball com AngularJS. La interfície de línia d'ordres de Ember és un paradís entre les eines de desenvolupament JavaScript. A més, l'equip central de Ember i la seva comunitat són molt actius.

Ara que les aplicacions d'una sola pàgina en JavaScript són cada vegada més complexes, estem veient que és cada vegada més important que la gestió de l'estat del client sigui predictible. **Redux**, amb els seus tres principis de restriccions per actualitzar l'estat, ha demostrat ser d'un valor incalculable en alguns dels projectes que hem implementat. Els tutorials Getting Started with Redux i idiomatic Redux són un bon començament tant per a nouvinguts com per a usuaris avançats. El seu disseny minimalista de llibreries ha fet aparèixer una gran quantitat d'eines i us animem a visitar el projecte redux-ecosystem-links per a veure exemples, programari intermediari i llibreries d'utilitats. També ens agrada especialment la facilitat de prova que ofereix: les accions de Dispatching, les transicions d'estat i la renderització es poden posar a prova independentment i amb el mínim mocking possible.

L'interès pel llenguatge de programació **Elixir** segueix creixent i el veiem, cada vegada més, en projectes seriosos. Els desenvolupadors, a més, troben que el seu model d'Actors és robust i molt ràpid. Elixir, construït a sobre de la màquina virtual Erlang, sembla molt prometedor per a la creació de sistemes concurrents i a prova de fallades. Té, a més, funcions distintives com l'operador Pipe que permet que els desenvolupadors creïn un canal de funcions tal com es faria en el shell de comandes UNIX. El bytecode compartit permet que Elixir operi juntament amb Erlang i utilitzi llibreries existents a la vegada que suporta eines com l'eina de Mix, la shell interactiva IEx i l'entorn de treball de proves



ExUnit.

Ens han agradat molt les proves d'interfície d'usuari que proporciona **Enzyme** per a aplicacions React.js. A diferència de molts altres entorns de treball de proves basats en imatges, Enzyme permet fer proves sense haver de renderitzar en el dispositiu, la qual cosa permet proves més ràpides i granulars. Això, sense dubte, ens ajuda a reduir molt la quantitat de proves funcionals que hem de fer a aplicacions React.

S'acostuma a emfatitzar la immutabilitat en el paradigma de programació funcional i la majoria de llenguatges tenen l'habilitat de crear objectes immutables, és a dir, que no es poden canviar un cop creats. **Immutable.js** és una llibreria per a JavaScript que proporciona diverses estructures de dades immutables que són molt eficients en màquines virtuals de JavaScript modernes. Els objectes Immutable.js no són, però, objectes de JavaScript normals per la qual cosa s'hauria d'evitar fer referències a objectes de JavaScript des d'objectes

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

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

ASSESS
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

HOLD
109. AngularJS
110. JSPatch

© November 2016, ThoughtWorks, Inc. All Rights Reserved.

LLENGUATGES & ENTORNS DE TREBALL *continuació*

immutables. Més equips estan utilitzant aquesta llibreria per ubicar mutacions i per mantenir l'estat en la producció. Recomanem als desenvolupadors que investiguin aquesta llibreria, especialment quan es combina amb la resta de llibreries de Facebook.

Alguns dels nostres equips a ThoughtWorks han quedat molt satisfets amb Phoenix, un entorn de treball model-vista-controlador (MVC) per a servidors escrits en Elixir. A més de ser fàcil d'utilitzar, Phoenix s'aprofita de Elixir per ser extremadament ràpid. Alguns desenvolupadors diuen gaudir tant de Phoenix com quan van descobrir Ruby on Rails. Tot i que l'ecosistema de llibreries de Phoenix no és tan gran com el d'altres entorns de treball més madurs, s'hauria de beneficiar de l'èxit i el creixement de Elixir.

La majoria dels nostres equips de iOS utilitzen l'aparellament **Quick i Nimble** per les seves proves unitàries. Dintre de la família RSpec d'eines de proves de desenvolupament guiat per comportament (BDD per les seves sigles en anglès), aquest proporciona proves molt fàcils de llegir (amb blocs descrits) sobre Swift i Objective-C i té un bon suport per proves asíncrones.

ECMAScript 2017, que no s'ha de confondre amb ES7 (també conegut com ECMAScript 2016), aporta diverses millores dignes de mencionar al llenguatge. S'espera que els navegadors implementin aquest estàndard l'estiu del 2017 però el compilador de JavaScript Babel ja suporta algunes de les característiques. Si es fa un gran ús de JavaScript i la base de codi s'està desenvolupant activament recomanem afegir Babel al canal de desenvolupament i provar les seves funcions.

JuMP és un llenguatge específic de domini per optimitzacions matemàtiques a Julia. JuMP defineix una API comuna anomenada MathProgBase i permet als usuaris d'escriure codi solucionador a Julia. Els solucionadors suportats actualment són: Artelys Knitro, Bonmin, Cbc, Clp, Couenne, CPLEX, ECOS, FICO Xpress, GLPK, Gurobi, Ipopt, MOSEK, NLopt i SCS. Un altre benefici és la implementació de tècniques de derivació automàtica en mode invers per a la derivació. D'aquesta manera els usuaris no estan limitats als operadors estàndards com cos, sin, log i sqrt i poden implementar les seves pròpies funcions a Julia.

Estem intrigats per l'estàndard **Physical Web** creat per Google. La idea darrera la Physical Web és simple (emetre una URL) però les possibilitats són enormes. Es tracta, bàsicament, d'una manera de portar el món real al regne digital. Actualment, el mecanisme de transport és Eddystone URLs , es fa per Bluetooth de baixa energia i ja hi ha disponibles clients de mostra. Tot i que entenem els problemes de seguretat que suposen seguir enllaços descoberts aleatòriament, ens interessa el seu ús amb clients personalitzats en els que es pot filtrar o allotjar les URLs segons sigui necessari.

Rapidoid és una col•lecció de mòduls d'entorns de treball web que inclou un ràpid servidor HTTP de baix nivell implementat des de zero a sobre de Java NIO. Un ús intel•ligent de les memòries d'entrada/sortida fora del monticle, de les piscines d'objectes i de les estructures de dades locals fan que Rapidoid sigui superior a servidors no basats en NIO com Netty. Com que encara és un projecte bastant nou, Rapidoid encara ha d'implementar algunes funcions com la memòria cau incorporada i el suport per SSL. Recomanem que es doni un cop d'ull al full de ruta per saber quan sortiran actualitzacions.

Ens entusiasma que el paradigma Redux s'hagi fet un lloc al món de Swift amb **ReSwift**. Veiem beneficis reals en la simplicitat i la llegibilitat de bases de codi un cop es gestionen l'estat i els canvis d'estat des d'un lloc centralitzat i amb un mateix idioma. Això també és útil per programar aplicacions "primer fora de línia."

Nosaltres creiem que, tot i la passió que desperten tots aquests nous cascos, hi ha diversos escenaris en què RA i RV es poden utilitzar al navegador, especialment en mòbils. En relació amb això, hem vist un augment en l'ús de **Three.js**, un poderós entorn de treball en JavaScript dissenyat per a la visualització i la renderització 3D. L'augment en el suport per WebGL, en el qual es basa, ha ajudat que s'adopti i compta amb una fervent comunitat.

En el món sempre canviant dels entorns de treball en JavaScript, **Vue.js** es desmarca com una alternativa lleugera per a AngularJS. S'ha dissenyat per ser una llibreria molt flexible i menys esbiaixada que ofereix una sèrie d'eines per a la creació d'interfícies web interactives al voltant de conceptes com la modularitat, els components i el flux reactiu de dades. És, a més, fàcil d'aprendre, la qual cosa el fa interessant per a desenvolupadors junior i principiants. Vue.js per si mateix no és un entorn de treball complet sinó que es centra només en la capa visible i, per tant, és fàcil d'integrar amb d'altres llibreries o a projectes ja existents.

© November 2016, ThoughtWorks, Inc. All Rights Reserved.

L'adopció massiva de RA/RV com un mitjà de col•laboració i comunicació requereix una plataforma de transmissió de dades moderna i ja disponible. **WebRTC** és un estàndard emergent per a la comunicació en temps real entre navegadors que permet la transmissió de vídeo amb tecnologies web comunes. El nombre de navegadors que suporten aquest estàndard no para de créixer però tant a Microsoft com a Apple els està costant adoptar-lo en els seus navegadors. Si la cosa segueix així, WebRTC podria formar els futurs fonaments per a la col•laboració RA/RV a la web.

AngularJS ha ajudat a revolucionar el món de les aplicacions JavaScript d'una sola pàgina i, a nosaltres, ens ha ajudat a dur a terme molts projectes durant molt de temps. No obstant això, ja no el recomanem (v1) per equips que comencin projectes de zero. Preferim les bases de codi més ràpides i fàcils de mantenir com Ember i React, especialment quan s'utilitza conjuntament amb Redux.

ThoughtWorks és una consultoria i comunitat tecnològica formada per persones apassionades i mogudes per objectius. Ajudem els nostres clients a posar la tecnologia al centre del seu negoci i, junts, creem els programes que més els importen. Ens dediquem al canvi social positiu: la nostra missió és crear una humanitat millor a través de programaris i ens associem amb moltes organitzacions que comparteixen els mateixos objectius.

Fundada fa més de 20 anys, ThoughtWorks ha crescut fins a convertir-se en una empresa formada per més de 4000 persones i amb una divisió de productes encarregada de crear eines pioneres pels equips de programació. ThoughtWorks té 40 oficines repartides en 14 països: Alemanya, Austràlia, Brasil, Equador, Espanya, Estats Units, Índia, Itàlia, Regne Unit, Singapur, Sudàfrica, Turquia, Xile i Xina.

ThoughtWorks®

© November 2016, ThoughtWorks, Inc. All Rights Reserved.