ThoughtWorks®

RADAR TECNOLÓGICO

NOV '16

Nuestros pensamientos acerca de la tecnología y las tendencias que están dando forma al futuro

thoughtworks.com/es/radar

#TWTechRadar

¿QUÉ HAY DE NUEVO?

Estos son los temas destacados de esta edición:

"DOCKER COMO PROCESO, PAAS COMO MÁQUINA, Y LA ARQUITECTURA DE MICROSERVICIOS COMO UN MODELO DE PROGRAMACIÓN"

El estilo de arquitectura de microservicios destaca las abstracciones cada vez más frecuentes en el mundo del desarrollo, debido al uso de contenedores y el énfasis en bajo acoplamiento, que ofrecen un alto nivel de aislamiento operativo. Los desarrolladores pueden pensar que un contenedor es un proceso autocontenido y que el PaaS es un objetivo de implementación común, usando la arquitectura de microservicios como estilo común. Desacoplar la arquitectura permite que se produzca lo mismo en los equipos, lo que disminuye el costo de coordinación entre núcleos. Debido a que resulta atractivo para desarrolladores y DevOps, se ha convertido en el estándar práctico en muchas organizaciones.

EMPODERAMIENTO INTELIGENTE

Los temas de investigación y desarrollo que han persistido durante mucho tiempo, tales como el aprendizaje de máquinas y la inteligencia artificial tienen repentinamente aplicaciones prácticas por medio de sistemas tales como Nuance Mix y TensorFlow. Los desarrolladores pueden descargar estos marcos que van desde NLP a librerías de aprendizaje de máquinas. Observamos, de forma satisfactoria, que las empresas, con frecuencia, tienen librerías y herramientas de fuente abierta y sofisticadas, en este espacio, que hace una década hubieran sido excesivamente caras y limitadas, por tanto. Ahora están disponibles a un amplio grupo de desarrolladores. Muchos factores han evolucionado y se han combinado para lograr crear nuevas herramientas, como por ejemplo: computación de comodidad, focalización en hardware específico tales como GPUs y recursos de nube. ¿Tal vez su acaparamiento de Grandes Datos está comenzando a dar resultados...?

EL EFECTO INTEGRAL DE LA ESTRUCTURA DEL EQUIPO

La estructura del equipo siempre ha tenido un gran impacto en una amplia variedad de temas de desarrollo de software y se ha vuelto un área de enfoque creciente debido a las bases tales como PaaS de autoservicio y los microservicios. Las empresas ahora prefieren pensar sobre productos en vez de proyectos. Las empresas de tecnología están divulgando el estilo "tú lo construyes y tú lo ejecutas", relacionado con la autonomía de equipos. Y estamos viendo el mismo tipo de pensamiento de producto aplicado a proyectos de empresas. Cuando la reestructuración de equipos produce mejores resultados, se demuestra nuevamente que el desarrollo de software sigue siendo, principalmente, un problema de comunicación. La creación de equipos interdisciplinarios aumenta la expansión provechosa de comunicaciones para funciones diferentes separadas tradicionalmente, que, a la vez, disminuye la fricción impuesta por estructuras artificiales como silos.

LA REALIDAD VIRTUAL Y LA REALIDAD AUMENTADA SE ESTÁN POPULARIZANDO GRADUALMENTE

Vemos que las realidades virtual y aumentada (AR/VR) generan intereses comerciales, aunque ambas tecnologías estaban antes relegadas a los juegos y novedades. Si bien seguir dibujos animados virtuales hizo que la gente se fijara en la realidad aumentada mediante SDKs móviles, hardware como Oculus Rift, HTC Vive y Microsoft HoloLens tiene cada vez más experiencia, a tal punto que los primeros pioneros pueden obtener beneficios sin tropezar con tecnología inmadura. A pesar de que las plataformas de software tales como OpenVR y Unity han logrado una madurez suficiente ya hace algún tiempo, las herramientas nuevas de procesamiento de lenguaje natural (NLP) tales como Nuance Mix y el hardware que proveen interacciones naturales tendrán un inmenso impacto en la adopción de las realidades virtual y aumentada. Ya tenemos laboratorios de realidades virtuales y aumentadas en nuestras oficinas para explorar aplicaciones futuras tales como interacciones remotas y sistemas de señalización al detalle. Nuestros experimentos demuestran que la realidad virtual es un medio sorpresivamente potente para una colaboración remota más empática y narración de cuentos, debido a su capacidad de evitar la abstracción y sumergir a los usuarios directamente en una experiencia. Sin embargo, veremos retos importantes en la creación y difusión de contenido de las realidades virtual y aumentada, cuando las destrezas y capacidades se rezaguen con respecto a las mejoras de hardware, especialmente en el mundo empresarial.

COLABORADORES

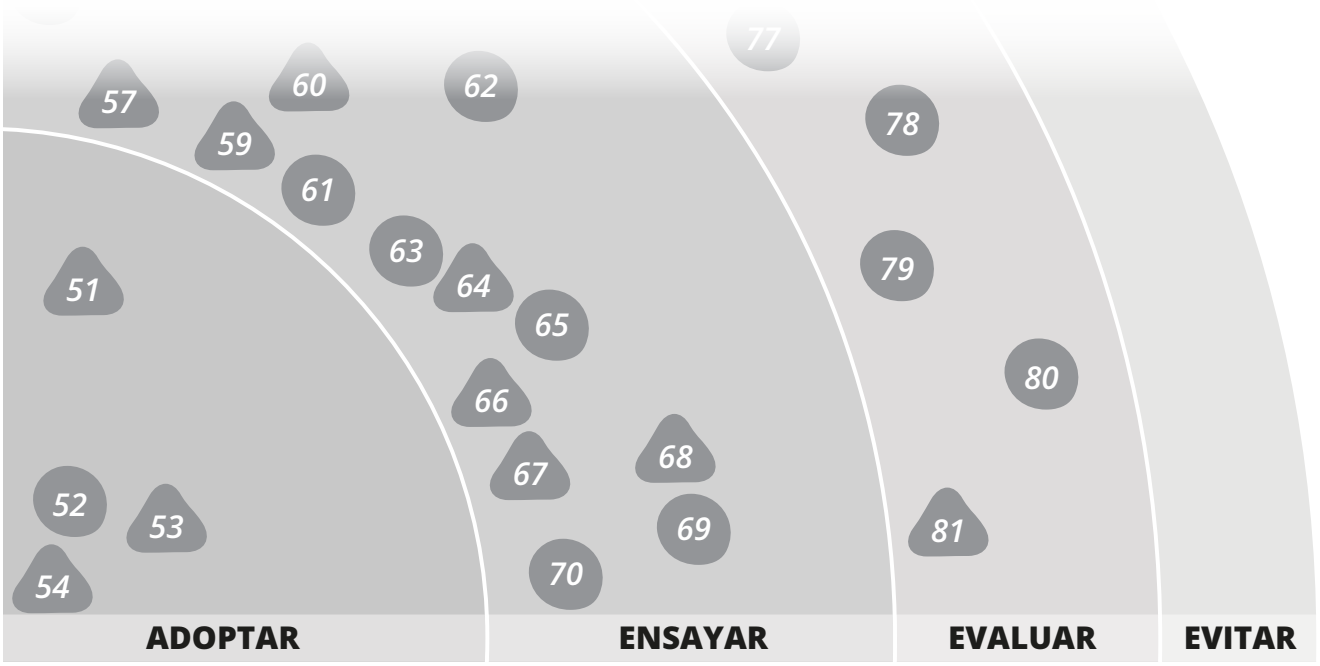
El Radar Tecnológico fue preparado por el Consejo Asesor de Tecnología de ThoughtWorks, cuyos miembros son:

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

ACERCA DEL RADAR TECNOLÓGICO

Los ThoughtWorkers son apasionados por la tecnología. La construimos, la investigamos, la probamos, la publicamos en código libre, escribimos acerca de ella, y constantemente buscamos mejorarla para todos. Nuestra misión es liderar la excelencia en el software y revolucionar la industria de las TI. Nosotros creamos y compartimos el Radar de Tecnología de ThoughtWorks como parte de esta misión. El Comité Consultor de Tecnología, es un grupo de experimentados líderes en tecnología de ThoughtWorks que crea el radar. Ellos se reúnen regularmente para discutir la estrategia global de tecnología en ThoughtWorks y las tendencias en la tecnología con un impacto significativo en nuestra industria.

El radar captura el resultado de las discusiones del Comité Consultor de Tecnología en un formato que provee valor a un amplio rango de interesados, desde CIOs hasta desarrolladores. El contenido está destinado a ser un resumen conciso. Nosotros alentamos a que explores estas tecnologías en para más detalles. El radar es gráfico por naturaleza, agrupando a los ítems en técnicas, plataformas, lenguajes y frameworks. Cuando los ítems del radar pueden aparecer en varios cuadrantes, escogemos el que nos parece más apropiado. Además agrupamos estos elementos también en cuatro anillos para reflejar nuestra postura actual respecto a ellos. Los anillos son:



Nosotros estamos convencidos de que la industria debería adoptar estos ítems. Nosotros los utilizamos cuando es apropiado en nuestros proyectos.

Valen la pena probar. Es importante entender como construir esta capacidad. Las empresas deberían probar esta tecnología en un proyecto en el que se puede manejar el riesgo.

Vale la pena explorar con la comprensión de el cómo va a afectar su empresa

Proceder con precaución..

Los ítems que son nuevos o que han tenidos cambios significativos desde el último radar son representados por triángulos, mientras que los ítems que no se han movido son representados como círculos. Son de nuestro interés muchos más ítems de los que pueden caber en un documento de este tamaño, así que removemos gradualmente algunos del último radar para crear espacio para los nuevos. Remover un ítem no significa que ya no sea de nuestro interés.

Para más información acerca del radar, visita thoughtworks.com/radar/faq

EL RADAR

TÉCNICAS

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

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

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

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

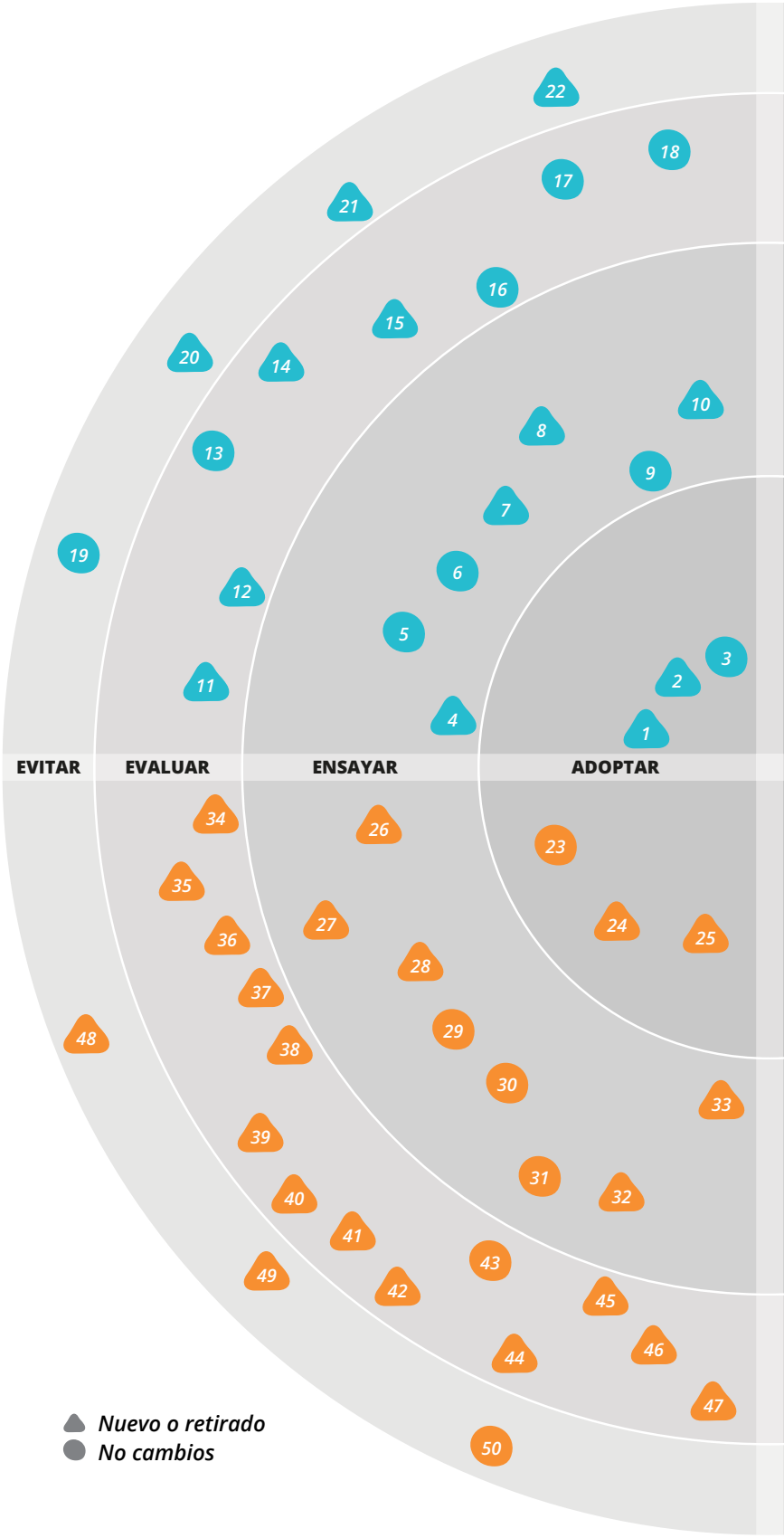
PLATAFORMAS

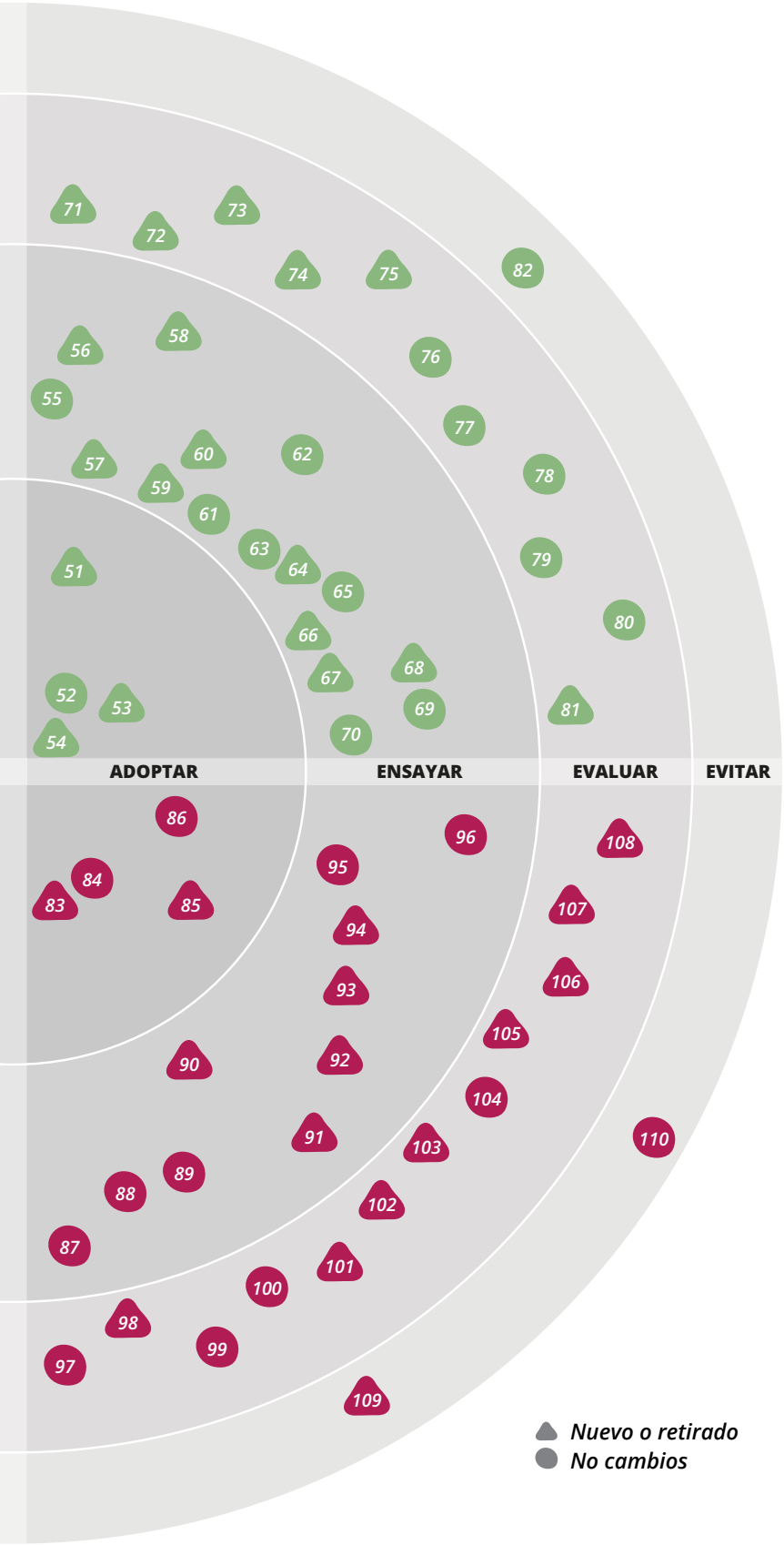
ADOPTAR
23. Docker
24. HSTS
25. Linux security modules

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

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

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





EL RADAR

HERRAMIENTAS

ADOPTAR

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

ENSAYAR

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

EVALUAR

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

EVITAR

82. Jenkins as a deployment pipeline

LENGUAJES & FRAMEWORKS

ADOPTAR

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

ENSAYAR

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

EVALUAR

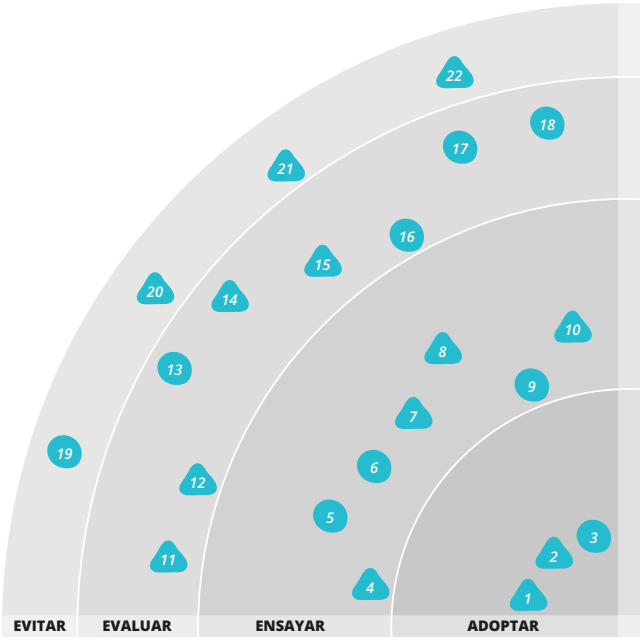
97. Aurelia
98. ECMAScript 2017 nuevo
99. Elm
100. GraphQL
101. JuMP nuevo
102. Physical Web nuevo
103. Rapidoid nuevo
104. Recharts
105. ReSwift nuevo
106. Three.js nuevo
107. Vue.js nuevo
108. WebRTC nuevo

EVALUAR

109. AngularJS
110. JSPatch

TÉCNICAS

Hemos decidido volver a hablar de **pruebas de contratos guiados por consumidores** para esta edición, a pesar de permitir que se desvanecieran en el pasado. El concepto no es nuevo, pero con la aceptación masiva de microservicios, necesitamos recordar a las personas que los contratos guiados por consumidores son una parte esencial de una cartera madura de pruebas de microservicios, que permite despliegues de servicios independientes. Pero, además, deseamos destacar que las pruebas de contratos guiados por consumidores son una técnica y una actitud que no requiere ninguna herramienta especial para implementarse. Nos encantan marcos de trabajo como Pact, porque facilitan, en ciertos contextos, la implementación de pruebas de contratos adecuadas. Pero hemos notado una tendencia en los equipos ha enfocarse en el marco de trabajo en vez de centrarse en las prácticas generales. El escribir pruebas en Pact no garantiza que se estén creando contratos guiados por consumidores. De igual manera, en muchas situaciones deberían crearse buenos contratos guiados por consumidores, aun cuando no existan herramientas integradas de pruebas.



Los equipos están exigiendo la automatización de todos sus ambientes, incluyendo su infraestructura de desarrollo. **Pipelines como código** se refiere al definir el pipeline de despliegue utilizando código en vez de configurar una herramienta de ejecución de CI/CD. LambdaCD, Drone, GoCD y Concourse son ejemplos que permiten del uso de esta técnica. Además, existen herramientas para automatizar la configuración de sistemas de CI/CD tales como GoMatic, que se puede utilizar para tratar el proceso de despliegue como código, versionado y probado.

Las empresas han adoptado con entusiasmo las APIs como una manera de exponer el potencial del negocio a desarrolladores externos e internos. Las APIs prometen la posibilidad de experimentar rápidamente con nuevas ideas de negocios recombinando competencias núcleo. Pero, ¿qué distingue una API de un servicio de integración empresarial común? Una diferencia consiste en tratar la **API como producto**, incluso cuando el consumidor es un sistema interno. Los equipos que crean APIs deben comprender las necesidades de sus clientes y lograr que el producto sea convincente. A lo largo del tiempo, los productos también son mejorados, mantenidos y soportados. Por lo tanto deberían tener un dueño que represente al cliente y que exija una mejora continua. Los productos son mantenidos y soportados de forma activa, además de ser fáciles de encontrar y usar. En nuestra experiencia, una orientación al producto es el ingrediente faltante que marca la diferencia entre la integración empresarial común y un negocio ágil construído sobre una plataforma de APIs.

En algunos países, vemos que varias agencias gubernamentales están intentando conseguir amplio acceso a información privada identificable personalmente (IPI). El incremento en el uso de soluciones públicas en la nube no permite que las organizaciones protejan fácilmente los datos de usuario confiados a estas por y a la vez cumplan con todas las leyes pertinentes. La Unión Europea tiene algunas de las leyes de privacidad más avanzadas y todos los principales proveedores de la nube, como Amazon, Google y Microsoft, ofrecen varios centros y regiones de datos en la Unión Europea. Por tanto, recomendamos que las empresas, especialmente aquellas con una base de usuarios globales, evalúen la opción de un puerto seguro para los datos de sus usuarios, mediante un

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

ENSAYAR
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

EVALUAR
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

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

TÉCNICAS *continuación*

alojamiento de IPI en la UE.Desde que escribimos sobre esta técnica en el último Radar, hemos implantado un nuevo sistema interno que maneja información sensible relacionada con todos nuestros empleados, y hemos seleccionado un centro de datos ubicado en la Unión Europea para albergarlo.

A pesar de que gran parte de la documentación puede ser reemplazada con código y pruebas legibles, en un mundo de arquitectura evolutiva, es importante registrar ciertas decisiones de diseño para el beneficio de futuros miembros del equipo y en caso de que se tenga una inspección externa. Los **Registros Livianos de Decisiones de Arquitectura** son una técnica para capturar decisiones importantes de arquitectura junto con su contexto y consecuencias. Aún cuando estos ítems a menudo se guardan en una wiki o en una herramienta de colaboración, generalmente preferimos almacenarlos bajo un sistema de control de versiones con un lenguaje de marcado (markup language) sencillo.

La arquitectura sin servidores es un enfoque que reemplaza a las máquinas virtuales de largo tiempo de ejecución con recursos computacionales efímeros, que comienzan a existir a petición y desaparecen inmediatamente después de su uso. Desde el último Radar, varios equipos han implementado aplicaciones utilizando un estilo "sin servidores". A nuestros equipos les gusta este enfoque, está funcionando bien para ellos y además consideran que es una opción válida de arquitectura. Tome en cuenta que "sin servidor" no significa un enfoque de todo o nada: algunos de nuestros equipos han implantado una parte de sus sistemas utilizando un enfoque "sin servidor" y, a la vez, siguieron usando un método de arquitectura tradicional para otros componentes.

Apesar de que muchos de los problemas que las personas enfrentan con los enfoques REST para API pueden atribuirse al antipatrón REST anémico, ciertos casos de uso justifican la exploración de otros enfoques. En particular, las organizaciones que soportan una larga lista de aplicaciones cliente (y, por ello, una posible proliferación de versiones de API inclusive al utilizar contratos dirigidos por consumidores) y que tienen gran parte de su API como apoyo a una alimentación de actividad de estilo 'lista-interminable', pueden llegar a un límite en arquitecturas tipo RESTful. Ocasionalemente, esto puede ser mitigado al emplear el enfoque de **consultas orientadas a clientes** al momento de la interacción entre el cliente y el servidor. Dicho enfoque es usado exitosamente tanto en GraphQL como en Falcor. En ellos, los clientes tienen más control tanto sobre los contenidos como en la granularidad de los datos recibidos. Esto pone más responsabilidad sobre la capa de servicios y, al mismo tiempo, produce un acoplamiento estrecho con el modelo de datos subyacente. No obstante, vale la pena explorar los beneficios si las bien modeladas API tipo RESTful no funcionan de la mejor manera.

La revolución en el mundo de los contenedores promovida por Docker ha reducido masivamente la fricción generada al mover las aplicaciones entre ambientes pero, a la vez, ha creado una gran brecha en los controles tradicionales sobre lo que puede enviarse a producción o no. La técnica de **escaneo de seguridad del contenedor** es una respuesta necesaria a este vector de amenazas. Docker provee ahora sus propias herramientas de escaneao de seguridad, y también CoreOS las tiene, asimismo,hemos obtenido resultados positivos con los parámetros de referencia de seguridad de CIS. Sin importar el enfoque que escoja, creemos que el tema de la validación automática de la seguridad en los contenedores tiene un gran valor y es una parte necesaria dentro de la mentalidad de las Plataformas como Servicio.

Se ha sabido por mucho tiempo que los conjuntos de grandes volúmenes de datos "anonimizados" pueden revelar información sobre las personas, sobretodo cuando se genera una referencia cruzada entre varios conjuntos de datos. Debido a la creciente preocupación sobre la privacidad personal, algunas empresas—incluyendo a Apple y Google—están recurriendo a técnicas de **privacidad diferencial**, con el propósito de mejorar la privacidad individual, manteniendo la capacidad de realizar un análisis útil sobre grandes cantidades de usuarios. La privacidad diferencial es una técnica criptográfica que intenta maximizar la precisión de las consultas estadísticas de una base de datos y a la vez minimizar las probabilidades de identificar sus registros. Se puede lograr este resultado incorporando una cantidad baja de "ruido" en los datos, pero es importante considerar que esta es un área de investigación aún abierta. Apple anunció planes para incluir privacidad diferencial en sus productos—y aplaudimos con entusiasmo su compromiso con la privacidad de los clientes—pero el acostumbrado secretismo de Apple ha dejado a algunos expertos en seguridad rascando sus cabezas. Continuamos recomendando Datensparsamkeit como una opción alternativa: simplemente almacenar el mínimo de datos que necesita logrará mejores resultados de privacidad en la mayoría de casos.

Hemos notado significantes beneficios de la incorporación de arquitecturas de micro-servicios, que han permitido que los equipos amplíen el alcance de sus servicios implementados y mantenidos de forma independiente. Sin embargo, los equipos han luchado, con frecuencia, para evitar la creación de interfaces

TÉCNICAS *continuación*

gráficas monoliticas, que son aplicaciones de navegador grandes y muy extensas. Tales monolitos son muy difíciles de mantener y evolucionan de la misma manera que las aplicaciones monolíticas de servidores que hemos abandonado. Estamos viendo que surge un enfoque que nuestros equipos denominan **micro interfaces gráficas**. En este enfoque, se divide una aplicación web según sus páginas y características, con cada funcionalidad perteneciendo, de extremo a extremo, a un solo equipo. Existen varias técnicas para unir las funcionalidades de la aplicación, tanto antiguas como nuevas, para crear una experiencia de usuarios coherente, pero la meta sigue siendo permitir que se desarrolle cada característica, con el fin de implantarla independientemente de las demás. El enfoque de BFF - backend for frontends funciona bien, en este caso. Cada equipo desarrollará un BFF para apoyar su juego de funcionalidades para la aplicación.

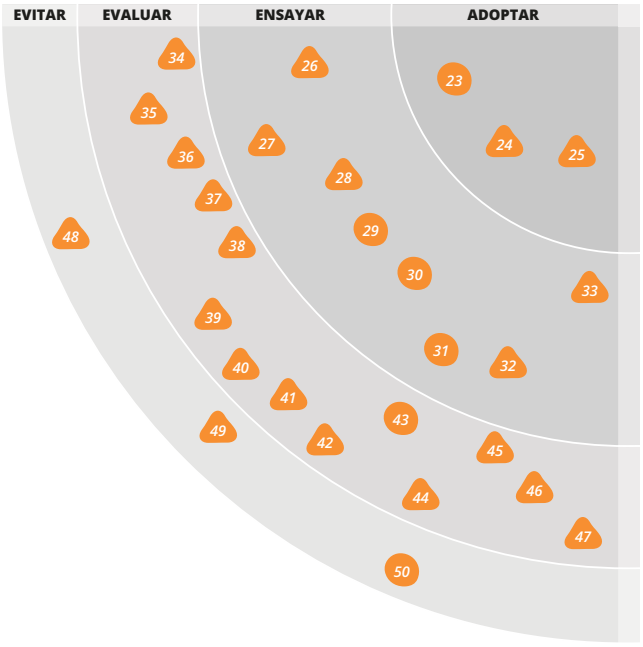
Debido a la popularidad creciente del patrón BFF - Backend for frontends y el uso de marcos de enlazado de datos de una vía, tales como React.js, hemos notado un rechazo a las arquitecturas con estilos similares a REST. Los críticos acusan a REST de producir interacciones poco eficientes demasiado abundantes entre sistemas y fallos de adaptación a las necesidades del cliente cuando evolucionen. Ofrecen marcos de trabajo como GraphQL o Falcor como mecanismos alternativos de extracción de datos que permiten al cliente especificar el formato de los datos resultantes. Pero, según nuestra experiencia, no es REST el que causa estos problemas. Más bien, surgen al modelar incorrectamente el dominio como un conjunto de recursos. El desarrollo ingenuo de servicios que simplemente expongan modelos de datos estáticos y jerárquicos mediante URLs con plantillas, produce una implementación de **REST anémico**. En un dominio de modelos con abundantes opciones, REST debería permitir algo más que la simple extracción repetitiva de datos. En una arquitectura RESTful completamente evolutiva, los eventos de negocios y conceptos abstractos también se modelan como recursos y la implementación debería usar eficazmente el hipertexto, las relaciones entre vínculos y los tipos de medios, con el fin de maximizar el desacoplamiento entre los servicios. Este antipatrón se relaciona estrechamente con el patrón de Modelo Anémico de Dominios y produce resultados en servicios de baja clasificación según el Modelo de Madurez de Richardson. Tenemos más consejos para el diseño de APIs de REST más eficaces en nuestro artículo de Insights.

Seguimos percibiendo que las organizaciones buscan tecnologías ""a la moda"", asumiendo complejidades y riesgos innecesarios cuando una opción más sencilla sería mejor. Un tema particular es el uso de sistemas distribuidos de Grandes Datos (""Big Data"") para conjuntos de datos relativamente pequeños. Este comportamiento nos hace resistir, una vez más, la **envidia de Grandes Datos**, con algunos datos adicionales de nuestra experiencia reciente. La base de datos Apache Cassandra promete escalabilidad masiva con hardware básico, pero hemos a equipos abrumarse con su complejidad arquitectónica y operativa. A menos que se tenga volúmenes de datos que requieran un conjunto de más de 100 nodos, no recomendamos el uso de Cassandra. El equipo operativo que se necesita solo para que ésta siga funcionando simplemente no vale la pena. Mientras redactábamos esta edición del Radar, discutimos varias tecnologías nuevas de base de datos, muchas de las cuales ofrecían mejoras de rendimiento diez veces mejor en relación con sistemas existentes. Siempre nos sentimos escépticos hasta que la nueva tecnología se ha comprobado adecuadamente, especialmente al tratarse de algo tan crítico como una base de datos. Jepsen provee un análisis de rendimiento de bases de datos bajo condiciones complicadas y ha encontrado numerosos errores en bases de datos NoSQL. Recomendamos mantener una dosis saludable de escepticismo y continuar echando un vistazo en sitios tales como Jepsen, al evaluar una tecnología de base de datos.

Ahora que más organizaciones están optando por desplegar aplicaciones en la nube, con frecuencia, hemos hallado grupos de TI que están intentando replicar su gestión de centros de datos existentes y enfoques de seguridad en la nube, de una manera poco económica. Con frecuencia, esto significa crear servidores de seguridad (firewall), balanceadores de carga, servidores proxy de redes, control de acceso y dispositivos y servicios de seguridad que se extienden a la nube sin requerir mayores esfuerzos de replanteo. Hemos vistos que las organizaciones crean sus propios APIs de orquestación delante de los proveedores de la nube, con el fin de limitar los servicios que puedan utilizar los equipos. En la mayoría de casos, estas capas solo sirven para paralizar la capacidad, al quitar la mayor parte de los beneficios previstos del traslado a la nube. En esta edición del Radar, hemos optado por volver a destacar la técnica de **cloud lift and shift** como algo que se debe evitar. Las organizaciones deben, más bien, revisar con mayor detenimiento la intención de sus controles de seguridad y operativos existentes, y buscar controles alternativos que funcionan en la nube, sin crear limitaciones innecesarias. Muchos de estos controles ya existen para proveedores de servicios en la nube con más años de experiencia y los equipos que adopten la nube pueden usar los APIs nativos para autoservicio de aprovisionamiento y operaciones.

PLATAFORMAS

HTTP Strict Transport Security (**HSTS**) (Seguridad Estricta de Transporte de HTTP) es una política actual ampliamente admitida, que permite que los sitios web se protejan de ataques de baja de versiones. En el contexto de HTTPS, un ataque de baja de versiones (downgrade attack) es aquel que puede hacer que los usuarios de su sitio regresen a HTTP, en lugar de HTTPS, lo que permite ataques adicionales tales como "hombre en el medio) "man-in-the-middle". Con HSTS, el servidor envía una cabecera que informa al explorador que solo deberá usar HTTPS para acceder a la página web. El soporte de navegadores es ahora lo suficientemente amplio para que esta opción sea fácil de implementar y agregar a cualquier sitio que utiliza HTTPS. Observatory de Mozilla puede ayudar a identificar esta y otras cabeceras útiles, además de opciones de configuración para ayudar a mejorar la seguridad y privacidad. Al implementar HSTS, es crítico verificar que todos los recursos se carguen adecuadamente por medio de HTTPS, ya que una vez que se activa HSTS, (casi) no hay forma de volver atrás hasta que se haya sobrepasado la hora de expiración. La directriz de incluir subdominios deberá agregarse pero, nuevamente, se requiere una verificación detallada de que todos los subdominios admitan el transporte seguro.



Listas blancas de aplicaciones han demostrado ser una de las formas más eficaces de mitigar ataques de intrusión. Una forma conveniente de implementar está práctica ampliamente recomendada es por medio de **módulos de seguridad de Linux**. Ahora que SELinux o AppArmor se incluyen por defecto en la mayoría de distribuciones de Linux, y con la disponibilidad de herramientas más completas y fáciles de usar como Grsecurity, hemos trasladado esta tecnología al anillo de Adoptar para esta edición. Estas herramientas ayudan a los equipos a evaluar preguntas acerca de quién tiene acceso a cuáles recursos en servidores compartidos, incluyendo los servicios contenidos. Este enfoque conservador de gestión de acceso ayudará a los equipos a incluir la seguridad en los procesos de ciclo de vida en el desarrollo de software (SDLC).

Seguimos teniendo experiencias positivas al desplegar la plataforma **Apache Mesos** para la gestión de grupos de recursos en sistemas altamente distribuidos. Mesos abstrae al exterior los subyacentes recursos de computación, como la UPC y el almacenamiento, con el objetivo de proveer una eficiente utilización a la vez de mantener el aislamiento. Mesos incluye a Chronos para la ejecución distribuida y tolerante a fallos de tareas programadas, y a Marathon para la orquestación de procesos de larga ejecución en contenedores.

Creemos, de forma cada vez más creciente, que para la mayoría de escenarios es rara vez indispensable implementar un código de autenticación propio. La gestión de identidad tercerizada acelera la entrega, disminuye los errores y tiene la tendencia de permitir

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

PLATAFORMAS *continuación*

una respuesta más rápida ante vulnerabilidades descubiertas recientemente. **Auth0** nos ha impresionado particularmente en este campo por su facilidad de integración, la gama de protocolos y conectores admitidos y su API de gestión enriquecida.

Nuestros equipos siguen disfrutando el uso de **AWS Lambda** y están comenzado a utilizarlo para experimentar con arquitecturas sin servidores, que combinan Lambda con el API Gateway. Recomendamos que las funciones de Lambda contengan solo una cantidad moderada de código. Es difícil garantizar la calidad de una solución basada en una maraña de varias funciones Lambda, y tal solución podría no ser eficaz en costos. Para necesidades más complejas, todavía son preferibles los despliegues basados en contenedores o maquinas virtuales. Además, hemos enfrentado problemas graves al usar Java para funciones de Lambda, con demoras erráticas de hasta varios segundos al arrancar el contenedor de Lambda. Ciertamente, podemos evitar este problema al usar JavaScript o Python, y si las funciones de Lambda no tienen mucho código, la selección del lenguaje de programación tampoco debería importar.

Realm es una base de datos creada para usarse en dispositivos móviles, con su propio motor de persistencia para lograr un gran rendimiento. Realm se mercadea como un reemplazo de SQLite y Core Data. Tome en cuenta que las migraciones no son tan sencillas como parecen según la documentación de Realm. Sin embargo, cada vez más equipos están escogiendo Realm como el mecanismo de persistencia en ambientes de producción para aplicaciones móviles.

Después de años de crecimiento como una plataforma para el desarrollo de juegos, **Unity** se ha convertido recientemente en la plataforma preferida para el desarrollo de aplicaciones de realidad virtual (VR) y realidad aumentada (AR). Ya sea que estés creando un mundo completamente inmersivo para los cascos Oculus o HTC Vive, una capa holográfica para tu nueva aplicación empresarial espacial o una serie de características de AR para tu aplicación móvil, Unity posiblemente provee lo necesario para crear prototipos y para desarrollar el producto final. Muchos de nosotros en ThoughtWorks creen que la realidad virtual y aumentada representan la siguiente transformación importante de las plataformas de computación y, por ahora, Unity es la única y más importante herramienta que usamos para desarrollar e impulsar este cambio. Hemos trabajado con Unity para desarrollar todos nuestros prototipos de realidad virtual, además de funcionalidades de realidad aumentada para cascos y aplicaciones para teléfonos y tabletas.

.NET Core es un producto modular de fuente abierta para crear aplicaciones que pueden ser fácilmente desplegadas en Windows, macOS y Linux. .NET Core hace posible el desarrollo de aplicaciones de web de multi-plataforma usando ASP.NET Core con un conjunto de herramientas, librerías y marcos de trabajo – que son otra opción para la arquitectura de microservicios. La comunidad alrededor de .NET Core y otros proyectos relacionados ha estado creciendo. Nuevas herramientas han aparecido y evolucionado rápidamente, tales como Visual Studio Code. Hay imágenes de Docker basadas en Linux y Windows (Nano Server) con .NET Core que simplifican la aplicación de una arquitectura de microservicios. CoreCLR y CoreFX aparecieron en una versión anterior del Radar. Sin embargo, hace unos meses Microsoft anunció el lanzamiento de .NET Core 1.0, la primera versión estable. Vemos buenas nuevas oportunidades, cambios y una comunidad vibrante como motivos para continuar evaluando este producto.

API Gateway de Amazon es la oferta de Amazon que permite a los desarrolladores exponer servicios de API a clientes de internet. Ofrece las usuales características de API Gateway como la gestión de tráfico, monitoreo, autenticación y autorización. Nuestros equipos han estado usando este servicio como una forma de acceder a otras capacidades de AWS tales como AWS Lambda, como parte del concepto de arquitecturas sin servidores. Continuamos monitoreando su desempeño frente a los retos presentados en gateways de API demasiado ambiciosos, pero en esta etapa la oferta de Amazon parece lo suficientemente ligera como para evitar esos problemas.

El interés continúa creciendo para **Apache Flink**, una plataforma de nueva generación para lotes ampliables y distribuidos y el procesamiento de flujos. En el núcleo de Apache Flink existe un motor de transmisión de flujo de datos, con soporte para operaciones de tipo tabular (similar a SQL), procesamiento de gráficos, y aprendizaje automático. Apache Flink se destaca por sus capacidades llenas de funcionalidades para el procesamiento de flujos, tales como tiempo de eventos, operaciones abundantes para ventanas de flujo, tolerancia de fallos y semántica de exactamente una vez. El proyecto muestra actividades continuas importantes, y la última versión (1.1) presenta nuevas integraciones de fuente y recepción de datos además de opciones mejoradas de transmisión.

Amazon lanzó recientemente el **Balanceador de carga de aplicaciones AWS** (ALB), un reemplazo directo de los balanceadores de carga elásticos, que se introdujeron

PLATAFORMAS *continuación*

en 2009. ALB admite inspecciones de tráfico de capa 7 y fue creado para soportar opciones modernas de arquitectura en la nube. Si está desarrollando un sistema basado en microservicios que utiliza ECS, los nuevos balanceadores de carga comprenderán directamente el alojamiento y expansión de contenedores, con varios contenedores y puertos para cada instancia de EC2. El enrutamiento de datos basados en contenido permite la segmentación de pedidos a grupos de servidores meta, junto con una expansión independiente de dichos grupos. Los controles de funcionamiento realizados por los balanceadores de carga han mejorado mucho, con la capacidad de captar métricas detalladas sobre el rendimiento de aplicaciones. Nos gusta todo lo que hemos visto aquí, y equipos han comenzado a informar de usos exitosos de ALB.

La base de datos Cassandra de Apache es una solución potente y escalable de Big Data para almacenar y procesar grandes cantidades de datos, frecuentemente usando cientos de nodos distribuidos en múltiples lugares del mundo. Es una gran herramienta y nos gusta, pero a menudo vemos que los equipos enfrentan problemas al usarla. Recomendamos el **uso de forma cuidadosa de Cassandra**. Los equipos, a veces, no comprenden el caso de uso de Cassandra, e intentan usarlo como un almacenamiento de datos de tipo general, cuando en realidad se optimizó para lecturas rápidas de conjuntos grandes de datos, con base en claves o índices predefinidos. Su dependencia en esquemas de almacenamiento puede dificultar su evolución a largo plazo. Cassandra tiene también una complejidad operativa significativa y bordes ásperos, entonces, a menos que realmente necesite la capacidad de expansión que provee, una solución más sencilla es usualmente la mejor opción. Si no necesita las características de caso de uso y expansión específicas de Cassandra, podría ser que la escogió solo por la envidia del Big Data. El uso cuidadoso de Cassandra incluirá pruebas automatizadas extensas y estamos encantados en recomendar CassandraUnit como parte su estrategia de pruebas.

Electron es un marco de trabajo sólido para desarrollar clientes nativos de escritorio con base en tecnologías web tales como HTML, CSS y JavaScript. Los equipos pueden aprovechar sus conocimientos y experiencia en el desarrollo web para entregar aplicaciones de escritorio pulidas para múltiples plataformas sin dedicar tiempo a aprender otro conjunto de tecnologías.

El alboroto parece haber alcanzado un apogeo en relación a blockchain y las cripto monedas, tal como se nota en los anuncios previos a chorro que están desacelerándose hasta gotear, y esperamos que algunos de los esfuerzos más especulativos se extingan a través del tiempo. Uno de los blockchains, **Ethereum**, está avanzando bien y vale la pena observarlo. Ethereum es un blockchain público con un lenguaje de programación incorporado que permite "contratos inteligentes" sean incluídos en este. Estos son movimientos algorítimicos de "éter" (la cripto moneda de Ethereum) que responden a la actividad que ocurre en el blockchain. R3Cev, la tecnología de blockchain para crear consorcios para bancos, desarrolló sus primeras pruebas de concepto basadas en Ethereum. Ethereum se ha usado para crear una organización autónoma distribuida (DAO), una de las primeras "corporaciones algorítmicas", aunque un robo reciente de 150 millones de dólares de Ether demostró que el blockchain y las cripto monedas todavía forman parte del mundo del lejano oeste de la tecnología.

Con el **HoloLens** Microsoft ha entregado el primer casco de realidad aumentada verdaderamente usable. No sólo es una pieza de diseño industrial realmente hermosa y un dispositivo muy cómodo de llevar puesto, sino también demuestra claramente la promesa de AR para el entorno empresarial mediante su bella óptica y profunda integración con Windows 10. Esperamos que HoloLens será la primera plataforma de AR con cual entreguemos funcionalidades de aplicacion sustanciales para nuestros clientes en un plazo muy cercano y observamos con interés su evolución a medida que gane más popularidad.

IndiaStack es un conjunto de Open APIs diseñado con el objetivo de transformar a India de un país pobre en datos a uno rico en datos. Esta pila enfatiza la innovación en capas al especificar un conjunto mínimo de APIs y alienta al resto del ecosistema a desarrollar aplicaciones personalizadas basadas en estos APIs. Aadhaar es una de las capas de base que provee servicios de autenticación a más de mil millones de ciudadanos de la India. Además, hay servicios para permitir transacciones sin papel por medio de firmas digitales (eSign), pago en línea unificado (UPI) y una capa de aceptación electrónica (e-KYC) para proveer detalles de Aadhaar a proveedores de servicios de forma segura. Creemos en la iniciativa impulsada por Open API para brindar innovación digital, y que los conceptos detrás de IndiaStack podrían usarse como un agente de cambio para otras regiones o países.

Nuance Mix es un marco de trabajo para procesamiento de lenguaje natural de la empresa que creó la tecnología de voz a texto Dragon Speaking y la primera versión de Siri. Este marco permite crear gramáticas para una interacción del usuario de forma libre mediante la voz. El desarrollador define una gramática específica de un dominio que el marco puede usar para su propio adiestramiento con el fin de comprenderla. Los resultados son respuestas a los datos introducidos por el usuario que identifican sus

PLATAFORMAS *continuación*

intenciones y conceptos de interacción. Al principio, se limita a frases cercanas a aquellas usadas para entrenarlo, pero después de un tiempo puede comenzar a identificar significados de frases más variadas. A pesar de estar en versión beta, la precisión de la exploración inicial ha sido convincente y vale la pena estar atentos al producto final para formularios que podrían aprovechar la interacción de usuarios de manos libre, como móviles, IoT, AR, VR y espacios interactivos.

OpenVR es el SDK subyacente en los trabajos con Unity de muchas de las pantallas de Realidad Virtual colocadas en la cabeza (HMD por sus siglas en Inglés) y posiblemente continuará creciendo en importancia. Gran parte del trabajo de VR en ThoughtWorks fue creado sobre OpenVR, porque funciona en cualquier HMD, a diferencia de otros SDKs. A pesar de que no es de fuente abierta, es gratis mediante la licencia. El SDK de Oculus es más limitado en su licencia y solo funciona en dispositivos de Oculus. Si bien OSVR es en verdad de fuente abierta, no ha sido muy adoptado. Si va a desarrollar una aplicación de realidad virtual y desea que funcione en la mayor cantidad de dispositivos posible, y no usar Unity ni Unreal para desarrollarla, OpenVR es la solución más concreta y pragmática por ahora.

Tarantool es una solución de código abierto NoSQL que combina una base de datos y el cache en una entidad y provee APIs para desarrollar lógica de aplicaciones en Lua. Se admiten motores en memoria y basados en disco y los usuarios pueden crear múltiples índices (HASH, TREE, RTREE, BITSET) basados en sus casos de uso. Los datos en sí se almacenan en formato de MessagePack y utilizan el mismo protocolo para comunicarse entre clientes y el servidor. Tarantool soporta bitácora de grabación anticipada, transacciones y replicación de maestro a maestro asincrónica. Nos agrada la decisión de arquitectura de motivar la política de un solo escritor y las operaciones de múltiples tareas cooperativas para manejar las conexiones concurrentes.

Las tendencias alrededor de la inteligencia artificial han alcanzado un gran apogeo, pero al igual que con el "Big Data", hay herramientas y marcos de trabajo útiles que están esperando ser descubiertos entre tanto alboroto. Una de estas herramientas es **wit.ai**, una plataforma SaaS que permite a los desarrolladores crear interfaces de conversación usando procesamiento de lenguaje natural (NLP). Wit funciona con entradas de voz o de texto, ayuda a los desarrolladores a gestionar la intención de las conversaciones y permite la implementación de lógica de negocios personalizada usando JavaScript. El sistema es gratuito para usos comerciales y no comerciales y alienta la creación de aplicaciones abiertas. Esté pendiente que tiene que estar de acuerdo con que Wit use sus datos para mejorar el servicio y para su propio análisis, por lo que se advierte que lea cuidadosamente los términos y condiciones. Otro contendor en este espacio es Microsoft Bot Framework, pero solo está disponible en vista previa limitada al redactar este artículo. Al igual que otros productos Microsoft, esperamos que Bot Framework evolucione rápidamente por lo que vale la pena hacerle un seguimiento.

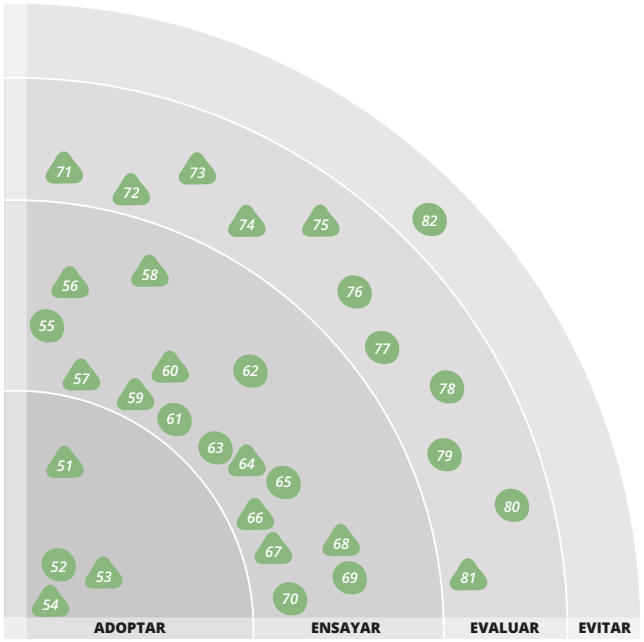
Hemos visto que demasiadas organizaciones se enfrentan a problemas cuando intentan usar su **CMS como plataforma** para manejar aplicaciones digitales grandes y complejas. Con frecuencia, esto se ve impulsado por proveedores con esperanzas de circunvalar a organizaciones de TI que no responden y permitir que los negocios arrastren y suelten cambios directamente en producción. Si bien apoyamos que se provea a los productores de contenido las herramientas y flujos de trabajo correctas, para aplicaciones con lógica compleja de negocios, generalmente, recomendamos tratar su CMS como un componente de su plataforma (en modo híbrido o sin cabecera), que coopere sin tropiezos con otros servicios, en vez de intentar implementar toda su funcionalidad en el CMS en sí. Una de nuestras quejas recurrentes es de inteligencia de negocio implementada en middleware, que resulta en software de interconexión que desea ejecutar lógica aplicaciones críticas. Los proveedores del mercado altamente competitivo de los Gateways de API continúan agregando funcionalidades para diferenciar sus productos. Esto genera productos **Gateways de API demasiado ambiciosos** cuya funcionalidad, además de proveer básicamente un proxy inverso, promueve la proliferación de diseños que son difíciles de probar y desplegar. Los Gateways de API pueden ser útiles para manejar algunas responsabilidades genéricas, tales como autenticación y límites de tasas, pero los aspectos inteligentes del dominio como la transformación de datos o el procesamiento de reglas deben residir en aplicaciones o servicios en donde pueden ser controladas por los equipos del producto que trabajen estrechamente en los dominios a los que brindan servicios.

HERRAMIENTAS

Babel.js se ha convertido en el compilador por defecto para desarrollar JavaScript de la siguiente generación. Su ecosistema está logrando grandes avances, gracias a su sistema reestructurado de plugins. Permite que los desarrolladores creen código de ES6 (e incluso ES7) que se ejecuta en el navegador o en el servidor sin sacrificar la retro-compatibilidad para navegadores más antiguos, y con muy poca configuración. Tiene soporte de primera clase para distintos sistemas de crear y probar, que facilita la integración con cualquier flujo de trabajo actual. Es un gran software que se ha convertido en el principal impulsor de adopción e innovación de ES6 (y ES7).

Cuando se combinan técnicas modernas y estilos de arquitectura, tales como microservicios, DevOps y QA de producción, los equipos de desarrollo necesitan procesos de monitoreo con una sofisticación cada vez mayor. Ya no es suficiente revisar los gráficos de uso de discos ni de UPC, y muchos equipos reúnen métricas de aplicaciones y específicos de negocios, con herramientas tales como Graphite y Kibana. **Grafana** facilita la creación de tableros útiles y elegantes para datos con una gran cantidad de fuentes. Una opción bastante útil permite sincronizar escalas temporales de gráficos distintos, lo que ayuda a identificar correlaciones de datos subyacentes. El sistema de plantillas que se está agregando luce prometedor y facilitará, posiblemente, la gestión de conjuntos de servicios similares. Con base en sus fortalezas, Grafana se ha convertido en nuestra opción preferida para esta categoría.

Las imágenes de máquina se han vuelto esenciales para procesos pipelines de despliegues modernas y hay varias técnicas y herramientas para crear las imágenes. Debido a su conjunto comprehensivo de funcionalidades y experiencias positivas que hemos tenido, recomendamos **Packer** a otras alternativas. También aconsejamos no intentar desarrollar scripts personalizados para hacer lo que Packer hace por defecto.



En la cúspide de la pirámide de pruebas para el desarrollo de aplicaciones Android, nuestros equipos usan cada vez más **Espresso** como una herramienta de pruebas funcionales. Su API de núcleo pequeño esconde los detalles de implementación poco elegantes y ayuda a escribir pruebas concisas, con una ejecución más rápida y confiable para las pruebas. Con Espresso, puede ejecutar pruebas de interfaz de usuario automatizadas que simulen las interacciones del usuario con una sola aplicación de meta en emuladores y dispositivos reales, para distintas versiones de Android.

fastlane es nuestra herramienta preferida para automatizar casi todas las actividades aburridas de desarrollo, pruebas, documentación y entrega de aplicaciones móviles de Android e iOS. Una configuración

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

HERRAMIENTAS *continuación*

sencilla, una gama de herramientas y múltiples pipelines convierten a fastlane en un ingrediente clave para la entrega continua de aplicaciones para dispositivos móviles.

El probar que capas de elementos y estilos para sitios web funcionen según lo esperado a través de varios factores de formas, pueden ser un proceso lento y frecuentemente manual. **Galen** ayuda a aliviar este problema al proveer un lenguaje sencillo que funciona encima de Selenium, y le permite especificar las expectativas para el aspecto externo de su sitio web en varios tamaños de pantalla. A pesar de que Galen adolece de los problemas típicos de fragilidad y velocidad de cualquier enfoque de pruebas de extremo a extremo, hemos aprovechado el beneficio que nos permite tener una idea clara de posibles errores de diseño en etapas iniciales.

Tener una forma de gestionar secretos de forma segura es cada vez más un enorme problema en proyectos. La vieja práctica de guardar secretos en un archivo o en variables de ambiente se está volviendo difícil de gestionar, especialmente en ambientes con varias aplicaciones y grandes cantidades de microservicios. **HashiCorp Vault** enfrenta el problema al proveer mecanismos para acceder de forma segura a secretos por medio de una interfaz unificada. La hemos utilizado con éxito en una gran cantidad de proyectos y a nuestros equipos les gustó lo fácil que era integrar Vault a sus servicios. Almacenar y actualizar secretos es algo complicado, porque dependen de una herramienta de línea de comandos y una gran dosis de disciplina del equipo.

Cada vez más proyectos están emitiendo y consumiendo información con formato JSON. Desarrollar pruebas de JSON en Java puede ser laborioso. **JSONassert** es una librería pequeña que permite escribir pruebas de JSON más cortas al simplificar las afirmaciones y proveer mejores mensajes de error.

Pa11y sirve para realizar pruebas de accesibilidad automáticas que se pueden ejecutar desde la línea de comandos e incorporar en un pipeline de generación. Nuestros equipos han logrado buenos resultados con Pa11y en un sitio altamente dinámico al crear primero una versión estática de HTML, y luego ejecutar pruebas de accesibilidad en contra de este. Para muchos sistemas, especialmente sitios web del gobierno, las pruebas de accesibilidad son un requisito y Pa11y las facilita en gran medida.

Gracias a la madurez de herramientas como Vault, ya no hay una excusa para almacenar secretos en repositorios de código, particularmente porque con esto se convierte en la parte débil de sistemas importantes. Ya hemos mencionado herramientas de escaneo de repositorios tales como Gitrob, pero ahora estamos alentado el uso de herramientas proactivas tales como **Talisman** (que fue creado por ThoughtWorks), que es un enganche previo de Git, para escanear aceptaciones de cambios donde puede haber patrones predefinidos de coincidencia con secretos.

Con **Terraform** se puede gestionar la infraestrucutra cloud escribiendo definiciones declaratorias. La configuración de los servidores instanciados por Terraform se deja usualmente a herramientas como Puppet, Chef o Ansible. Nos gusta Terraform porque la sintaxis de sus archivos es fácil de leer y porque soporta un buen número de proveedores de cloud sin intentar proveer una abstarcción artificial entre estos proveedores. Después de nuestra primera, más cauta, mención de Terraform de hace dos años, se ha visto un desarrollo continuo y ha evolucionado a un producto estable que ha probado su valor en nuestros proyectos. El tema con la gestión del archivo de estado puede ahora esquivarse utilizando lo que Terraform llama un “remote state backend”. Hemos utilizado Consul exitosamente para ese propósito.

La programación en pares es una técnica esencial para nosotros y, dado a que cada vez vemos que más equipos tienen miembros distribuidos en varios lugares, hemos experimentado con algunas herramientas que apoyen este trabajo remoto en pares. Ciertamente, nos gustó ScreenHero pero nos preocupa su futuro. Para equipos que no dependen de un IDE gráfico, el uso de **tmate** para trabajo en pares se ha vuelto una gran solución. tmate es una bifurcación de la popular herramienta tmux, y si se compara con tmux para pares remotos, la configuración es mucho más sencilla. El ancho de banda y los requisitos de recursos son modestos, al compararlos con soluciones gráficas de pantalla compartida, y obviamente nunca habrá pantallas borrosas. Los equipos pueden también configurar su propio servidor, para retener el control completo de la privacidad e integridad de la solución.

Android-x86 es un puerto del proyecto de fuente abierta Android a plataformas x86. El proyecto se inició al albergar varios parches realizados por la comunidad de soporte de x86 pero luego esta creó su propia base de código con el fin de ofrecer soporte para varias plataformas x86. Hemos visto mucho ahorro de tiempo al usar Android-x86 en nuestros servidores de

HERRAMIENTAS *continuación*

integración continua en vez de emuladores para pruebas herméticas de interfaces de usuario. Sin embargo, para las pruebas específicas de interfaz de usuario que se enfocan en una resolución de dispositivos en particular, al simular una baja cantidad de memoria, poco ancho de banda y baja batería, es mejor quedarse con los emuladores.

Nuestros equipos han logrado buenos resultados con **axios**, un cliente HTTP basado en promesas de JavaScript que han descrito como ""mejor que Fetch."" El proyecto ha logrado mucho apoyo y actividad en GitHub, y cuenta con nuestra aprobación.

Debido al crecimiento del interés en arquitecturas de datos de transmisión y los depósitos de datos finales que ellos alimentan, nos hemos dado cuenta de una dependencia creciente en herramientas para "capturar cambios en datos" para conectar los depósitos de datos transaccionales a sistemas de procesamiento de flujos. **Bottled Water** es un componente adicional bienvenido en este campo, por convertir los cambios de la bitácora de grabación anticipada de PostgreSQL a eventos de Kafka. Sin embargo, un aspecto negativo de este enfoque es que estará atado a los eventos de base de datos de bajo nivel en vez de los eventos de negocios de alto nivel, que recomendamos como la base de una arquitectura orientada a eventos.

Uno de los debates perpetuos entre desarrolladores incluye la clasificación de tipeo de lenguajes: ¿Cuánto es lo justo necesario? Clojure, Lisp funcional de tipeo dinámico en el JVM, agregó una nueva opción que desdibuja las lineas en esta discusión **Clojure.spec** es una nueva facilidad incorporada en Clojure que permite a los desarrolladores empaquetar el tipo y otros criterios de verificación alrededor de estructuras de datos, como rangos permitidos de valores. Una vez establecido, Clojure utiliza estas especificaciones para proveer una gama de beneficios: pruebas generadas, validación, desestructuración de estructuras de datos entre otros. Clojure.spec es una manera prometedora de tener los beneficios de tipos y rangos donde los desarrolladores los necesiten, pero no en todas partes.

El probar la parte visual de las aplicaciones de iOS puede ser lento, penoso y poco confiable, por lo que nos alegra incluir **FBSnapshotTestcase** a nuestra caja de herramientas. FBSnapshotTestcase automatiza la captura, almacenamiento y búsqueda de diferencias en los componentes visuales para que estos se puedan mantener perfectos hasta el último pixel. Ya que esta herramienta se ejecuta como una prueba unitaria (en el simulador), es más veloz y confiable que otras estrategias basadas en pruebas funcionales.

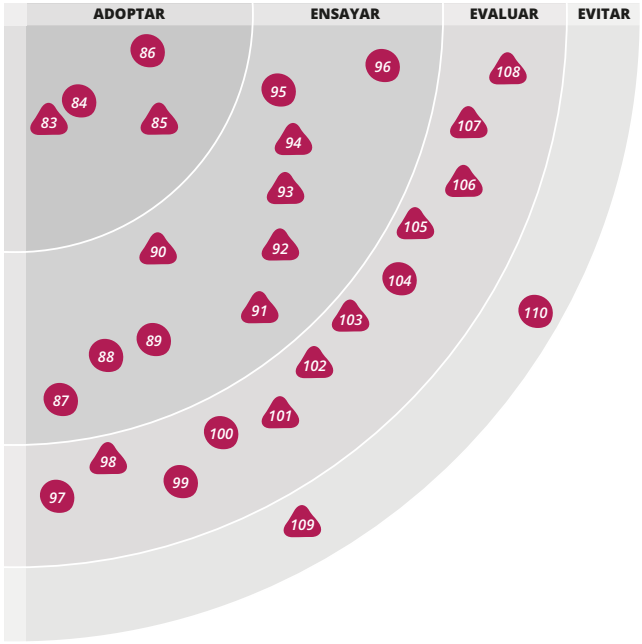
Scikit-learn es una librería desarrollada en Python para aprendizaje de máquina cada vez más popular. Provee un conjunto robusto de modelos de aprendizaje de máquina tales como agrupaciones, clasificación, regresión y reducción de dimensiones, y un conjunto abundante de funcionalidad para tareas anexas como selección y evaluación de modelos, además de preparación de datos. Debido a que fue diseñado para ser sencillo y reusable en varios contextos, además de estar bien documentado, podemos notar que esta herramienta será accesible incluso para quienes no son expertos, para que exploren el espacio de aprendizaje de máquinas.

LENGUAJES & FRAMEWORKS

Si se enfrenta a la tarea de desarrollar una aplicación de página-única (SPA) e intentar escoger un marco de trabajo, **Ember.js** ha pasado a ser la opción preferida. Nuestros equipos han elogiado a Ember por su experiencia altamente productiva para el desarrollador, con mucho menos sorpresas que otros marcos de trabajo tales como AngularJS. La herramienta Ember CLI es un refugio en el torbellino de herramientas de construcción de JavaScript, y el equipo principal y comunidad de Ember son muy activos y responden con celeridad.

Debido a la complejidad creciente de aplicaciones de una página de JavaScript, hemos notado una necesidad más apremiante de hacer que la gestión del estado del lado de clientes sea predecible. **Redux**, con sus tres principios de restricciones para actualizar el estado, se ha convertido en algo invaluable en varios proyectos que hemos implementado. Los tutoriales Getting Started with Redux e idiomatic Redux son buenos puntos de partida para usuarios nuevos y aquellos con experiencia. Su diseño minimalista de librerias ha producido una amplia gama de herramientas, y le alentamos a revisar el proyecto redux-ecosystem-links donde hay ejemplos, middleware y conjuntos de utilitarios. Nos gusta especialmente el relato de capacidad de realizar pruebas: Las acciones de envío, transiciones de estado y generaciones pueden testearse por separado como unidades de una a otra y con cantidades mínimas de simulaciones.

Continúa creciendo el interés por el lenguaje de programación **Elixir**. De forma creciente, vemos que se usa en proyectos serios y escuchamos comentarios de desarrolladores que consideran a su modelo Actor como robusto y muy rápido. Elixir, que se desarrolló basándose en la máquina virtual Erlang, está demostrando muchas posibilidades para crear sistemas altamente concurrentes con tolerancia a fallas. Elixir tiene características diferenciadoras como el operador Pipe que permite a los desarrolladores crear una tubería de funciones de forma similar a comandos shell de Unix. El código de bytes compartido le permite a Elixir operar de forma conjunta con Erlang y aprovechar las librerías existentes, a la vez que admite herramientas como la de construcción Mix, el shell interactivo IEx y el marco de trabajo para pruebas unitarias ExUnit.



Hemos disfrutado las pruebas rápidas de interfaz a nivel de componentes que **Enzyme** provee para aplicaciones de React.js. A diferencia de muchos otros marcos de trabajo de pruebas basados en capturas de pantalla, Enzyme le permite realizar pruebas sin generación en el dispositivo, para lograr testear de forma más rápida y más granular. Este es un factor que contribuye a nuestra capacidad de reducir masivamente la cantidad de pruebas funcionales que pensamos que debemos hacer para aplicaciones React.

La inmutabilidad, con frecuencia, es algo que se enfatiza en el paradigma de programación funcional, y la mayoría de lenguajes tienen la capacidad de crear objetos inmutables, que no pueden cambiarse después de crearse. **Immutable.js** es una librería para JavaScript que provee muchas estructuras de datos persistentes e inmutables, que son muy eficientes en máquinas virtuales modernas de JavaScript. Los objetos de Immutable.js no son, sin embargo, objetos de JavaScript normales, por lo que se deben evitar las referencias a objetos JavaScript desde objetos inmutables. Más equipos están usando esta librería para rastrear la

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

LENGUAJES & FRAMEWORKS *continuación*

mutación y mantener el estado en la producción. Recomendamos que los desarrolladores investiguen esta librería, especialmente cuando se la combine con el resto del stack de Facebook.

Algunos de nuestros equipos de ThoughtWorks han tenido experiencias muy positivas con **Phoenix**, un marco de trabajo MVC del lado de servidor para web desarrollado en Elixir. Además de ser ligero y fácil de usar, Phoenix aprovecha Elixir para ser extremadamente rápido. A algunos desarrolladores, Phoenix les recuerda la felicidad que sintieron al descubrir por primera vez Ruby y Rails. A pesar de que el ecosistema de librerías para Phoenix no es tan amplio como el de algunos marcos de trabajo más maduros, debería beneficiarse del éxito continuo y del creciente apoyo para Elixir.

Actualmente, la mayoría de nuestros equipos de iOS usan el par **Quick y Nimble** para pruebas unitarias. Como parte de la familia de herramientas de pruebas de desarrollo basadas en comportamiento (BDD) RSpec, estos proveen pruebas altamente legibles (con bloques de descripciones) tanto para Swift como Objective-C.

Además, tienen buen soporte para pruebas asincrónicas.

ECMAScript 2017, que no debe confundirse con ES7 (es decir, ECMAScript 2016) cuenta con varias mejoras notables al lenguaje. Se espera que los navegadores implementen este estándar, por completo, en el verano de 2017, pero el compilador de JavaScript Babel actualmente ya soporta muchas de estas funcionalidades. Si usa con frecuencia JavaScript y su base de código está bajo desarrollo activo, recomendamos que agregue Babel a su proceso de desarrollo y comience a usar sus características soportadas.

JuMP es un lenguaje de dominio específico para optimizaciones matemáticas de Julia. JuMP define un API común denominado MathProgBase y permite a los usuarios desarrollar en Julia código no dependiente del solucionador. Entre los solucionadores admitidos, podemos mencionar Artelys Knitro, Bonmin, Cbc, Clp, Couenne, CPLEX, ECOS, FICO Xpress, GLPK, Gurobi, Ipopt, MOSEK, NLopt y SCS. Otro beneficio adicional es la implementación de la técnica de diferenciación automática en modo invertido, con el fin de calcular derivadas, para que los usuarios no se limiten a los cálculos estándar como seno, coseno, logaritmo y raíz cuadrada, sino también puedan implementar su propias funciones objetivas personalizadas en Julia.

Seguimos intrigados por el estándar **Physical Web** creado por Google. La idea de Physical Web es sencilla, balizas difunden una URL, pero las posibilidades son muy amplias. De forma básica, es una manera de hacer anotaciones en el mundo físico, al relacionar los objetos y lugares con el ámbito digital. El mecanismo actual de transporte es URL de Eddystone por medio de Bluetooth LE, y clientes de muestra estan disponibles. A pesar de que hay preocupaciones obvias de seguridad con seguir los vínculos descubiertos al azar, estamos más interesados en casos de uso para clientes personalizados, con el fin de filtrar o representar los URL como se requiera.

Rapidoid es una colección de módulos de marco de trabajo web, incluyendo un servidor HTTP rápido de bajo nivel implementado desde su inicio basado en Java NIO. El uso inteligente de memorias provisionales fuera de la pila de entrada y salida, grupos de objetos y estructuras de datos locales de hebra dan una ventaja a Rapidoid con respecto a otros servidores basados en NIO, tales como Netty. Por ser un proyecto bastante nuevo, Rapidoid todavía no implementa características tales como la memoria cache incorporada y el soporte para SSL. Sugerimos que revise la hoja de ruta para actualizaciones.

Nos emociona que el paradigma Redux haya sido incluido en el mundo Swift en la forma de **ReSwift**. Hemos encontrado grandes beneficios tanto en la simplicidad como en la legibilidad de las bases de código, al momento de gestionar los estados, y sus cambios dentro de un sitio centralizado y de formato común. Es también un apoyo al desarrollar aplicaciones “primero fuera de línea”.

A pesar del interés que genera la oleada de nuevos dispositivos de realidad aumentada y virtual, creemos que hay muchos escenarios que tienen sentido en el navegador, especialmente de dispositivos móviles. Debido a esta tendencia, hemos visto un aumento en el uso de **Three.js**, un potente marco de trabajo JavaScript para visualización y generación 3D. El crecimiento de soporte para WebGL, en el que se basa, ha ayudado a su adopción, asi como el apoyo activo de la comunidad en este proyecto de fuente abierta.

En el mundo cambiante de marcos de front-end de JavaScript, **Vue.js** ha logrado abrirse campo como una alternativa liviana a AngularJS. Se diseñó para ser una librería muy flexible y con menos opiniones firmes, que ofrece un conjunto de herramientas para desarrollar interfaces activas de web basadas en conceptos como la modularidad, componentes y flujo reactivo de datos. Es fácil de aprender, lo que lo hace interesante para desarrolladores de menos experiencia y principiantes. Vue.js en sí no es un marco con todos los componentes. Se enfoca solo en la capa de vista y, por tanto, es fácil integrarlo con otras librerías o proyectos existentes.

La amplia adopción de la realidad aumentada (AR)

y virtual (VR) como un medio de colaboración y comunicación requiere de una plataforma moderna y fácilmente disponible para la transmisión de video. **WebRTC** es un estándar emergente para la comunicación en tiempo real entre navegadores que permite la transmisión de video entre tecnologías disponibles comunmente en la web. Los navegadores que soportan este estándar están aumentando, pero Microsoft y Apple han demorado en adoptar WebRTC para sus navegadores propietarios. Si continua creciendo el momento, WebRTC podría convertirse en base para la colaboración AR/VR en la web.

AngularJS helped revolutionize the world of single-page JavaScript applications, and we have delivered many projects successfully with it over the years. However, we are no longer recommending it for teams starting fresh projects unless they already have experience or an existing investment in it. Although Angular 2 is a different beast from version 1, we prefer the ramp-up speed and more maintainable codebases we are seeing with Ember and React, particularly in conjunction with Redux.

ThoughtWorks es una compañía de software y una comunidad de individuos apasionados cuyo propósito es revolucionar el diseño, creación y entrega de productos de software. Pensamos de forma disruptiva para ofrecer tecnología que haga frente a los retos más difíciles de nuestros clientes, a la vez que buscamos revolucionar la industria de TI y crear un cambio social positivo. Creamos herramientas innovadoras para equipos de desarrollo de software que aspiran a ser grandes. Nuestros productos ayudan a las organizaciones a mejorar continuamente y entregar software de calidad para sus necesidades más críticas. Fundada hace 20 años, ThoughtWorks ha crecido a partir de un pequeño grupo de personas en Chicago hasta convertirse en una compañía de más de 4000 empleados repartidos a lo largo de sus 40 oficinas en 14 países: Australia, Brasil, Canadá, Chile, China, Ecuador, España, Alemania, India, Singapur, Sudáfrica, Turquía, Reino Unido y los Estados Unidos.

ThoughtWorks®