ThoughtWorks®

RADAR TECNOLÓGICO

ABRIL '16

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

thoughtworks.com/radar

QUÉ HAY DE NUEVO?

Estas son las tendencias sobresalientes en esta edición:

EL CÓDIGO LIBRE COMO UN DERIVADO VIRTUOSO

Algunos de los programas más influyentes que aparecen en nuestro radar provienen de empresas cuya primera prioridad no es la creación de herramientas de software. Varios de nuestros aportes en el radar provienen de Facebook, que no se considera como un fabricante de herramientas de desarrollo de software tradicional. A diferencia del pasado, hoy en día muchas compañías construyen sus activos de software más importantes bajo la premisa de código libre para atraer a nuevos reclutas y certificarse a sí mismos. Esto crea un ciclo de retroalimentación virtuoso: Código Libre Innovador atrae buenos desarrolladores que a su vez están más propensos a innovar. Como efecto secundario, los marcos de trabajo y las librerías de estas empresas son algunos de los más influyentes en la industria. Esto representa un gran cambio en el ecosistema de desarrollo de software y es una prueba más de la eficacia del software del código libre ... bajo el contexto adecuado (nuestro consejo sobre Web Scale Envy sigue en pie)

ANALIZANDO EL ROMPECABEZAS DE LAS PLATAFORMAS COMO SERVICIO (PAAS)

Muchas grandes organizaciones ven la nube y la Plataforma como Servicio (PaaS) como una forma obvia para estandarizar la infraestructura, facilitar el despliegue, las operaciones, y hacer que los desarrolladores sean más productivos. Pero aún es muy pronto, la definición de PaaS sigue siendo nebulosa, y en muchos casos los enfoques son incompletos o sufren de la inmadurez de soportar marcos de trabajo y de herramientas. Algunas soluciones PaaS hacen que sea más difícil de hacer cosas fácilmente logrables con una simple infraestructura como servicio (IaaS), tales como el uso de un localizador de servicio o topologías de red complejas, y el jurado aún está deliberando sobre si el enfoque de “Contenedores como servicio” aportará un valor similar con una mayor flexibilidad. Vemos a muchas empresas utilizando implementaciones de PaaS o poco a poco construyendo sus propias, con diversos grados de éxito. Tenemos la sospecha de que cualquier PaaS construido hoy en día no será una versión final, sino más bien parte de un camino evolutivo. La migración de las empresas a la nube y a las PaaS, a pesar de traer múltiples beneficios, tiene dificultades y desafíos, particularmente alrededor del diseño general de pipelines y herramientas. Los consumidores de estas tecnologías deben buscar el punto de inflexión que indique “el momento estelar” para su propio contexto y deben evitar el acoplamiento forzado a los detalles de implementación de su PaaS.

DOCKER, DOCKER, DOCKER!

El uso de contenedores, y Docker en particular, ha demostrado ser enormemente beneficioso como una técnica de gestión de aplicaciones, racionalización del despliegue entre los ambientes y para la simplificación de los problemas “funciona aquí, pero no allá”. Vemos una cantidad significativa de energía centrada en el uso de Docker y particularmente, en el ecosistema que lo rodea, más allá del desarrollo/pruebas y todo el camino a la producción. Los contenedores Docker se utilizan como la “unidad de escalamiento” para muchas plataformas PaaS y “data center OS”, dando a Docker aún más impulso. A medida que madura, tanto como un entorno de desarrollo como de producción, la gente está prestando más atención a la contenerización, sus efectos secundarios y sus implicaciones.

¿EXCESO DE REACCIONES?

La programación reactiva- donde los componentes reaccionan a los cambios en los datos que se propagan a ellos en lugar de usar un modelo imperativo - se ha vuelto extremadamente popular, con extensiones reactivas disponibles en casi todos los lenguajes de programación. En particular las interfaces de usuario se escriben habitualmente en un estilo reactivo, y muchos ecosistemas se están asentando en este paradigma. Aunque nos gusta el modelo, el uso excesivo de los sistemas basados en eventos complica la lógica del programa, por lo que es difícil de entender; los desarrolladores deben utilizar este estilo de programación con criterio. Sin duda alguna, es muy popular: Hemos añadido un número significativo de marcos de trabajo reactivos y herramientas de soporte en esta edición del Radar.

CONTRIBUYENTES

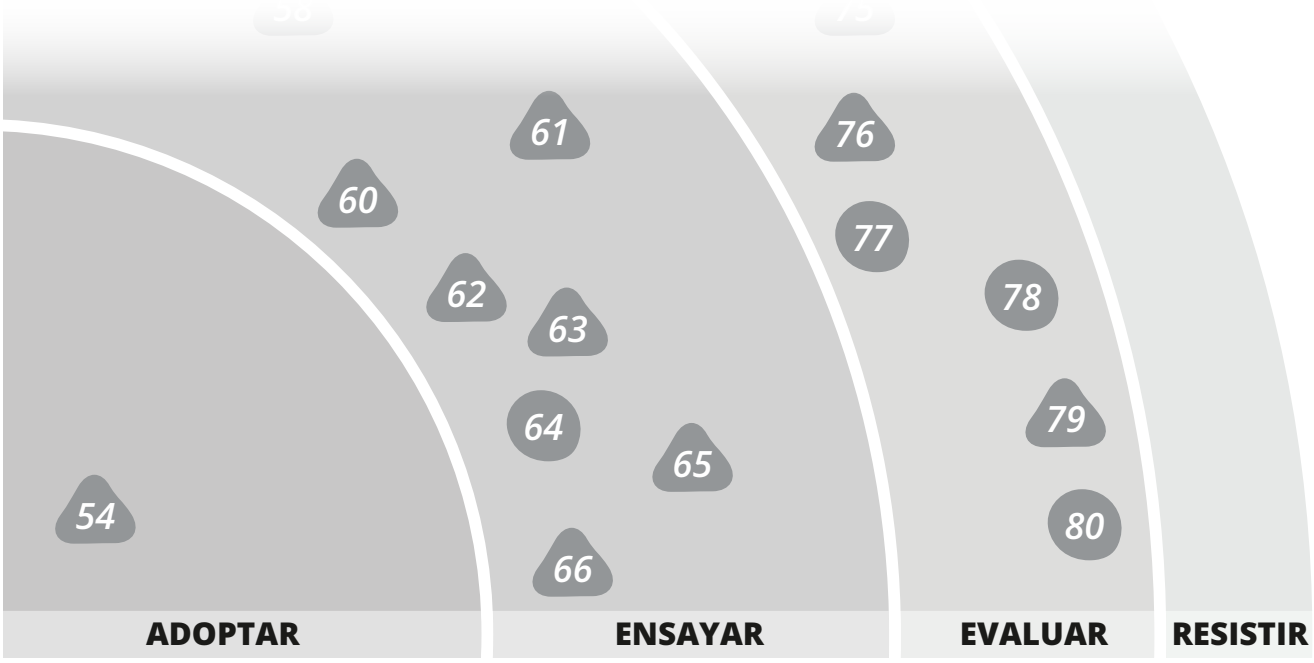
El radar tecnológico es preparado por el comité Consultor de Tecnología de ThoughtWorks, compuesto por:

Rebecca Parsons (CTO)	Dave Elliman	Ian Cartwright	Rachel Laycock
Martin Fowler(Chief Scientist)	Erik Doernenburg	James Lewis	Sam Newman
Anne J Simmons	Evan Bottcher	Jonny LeRoy	Scott Shaw
Badri Janakiraman	Fausto de la Torre	Mike Mason	Srihari Srinivasan
Brain Leke	Hao Xu	Neal Ford	Thiyagu Palanisamy

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. Desacoplamiento del deployment desde el release
2. Productos sobre proyectos
3. Modelos de amenazas

ENSAYAR
4. BFF - Backend para frontends
5. Bug bounties (Recompensas por errores)
6. Data Lake (Lago de Datos)
7. Event Storming
8. Flux
9. Filtro Idempotency
10. iFrames para sandboxing
11. NPM para todas las cosas
12. Ambientes Phoenix
13. QA en producción
14. Arquitecturas Reactivas

EVALUAR
15. Políticas de Seguridad de Contenido nuevo
16. Hosted IDE's
17. Alojamiento de datos PII en la EU nuevo
18. Monitoring of invariants
19. OWASP ASVS nuevo
20. Serverless architecture nuevo
21. Unikernels nuevo
22. VR beyond gaming nuevo

RESISTIR
23. A single CI instance for all teams nuevo
24. Big Data envy nuevo
25. Gitflow
26. High performance envy/web scale envy
27. SAFe™

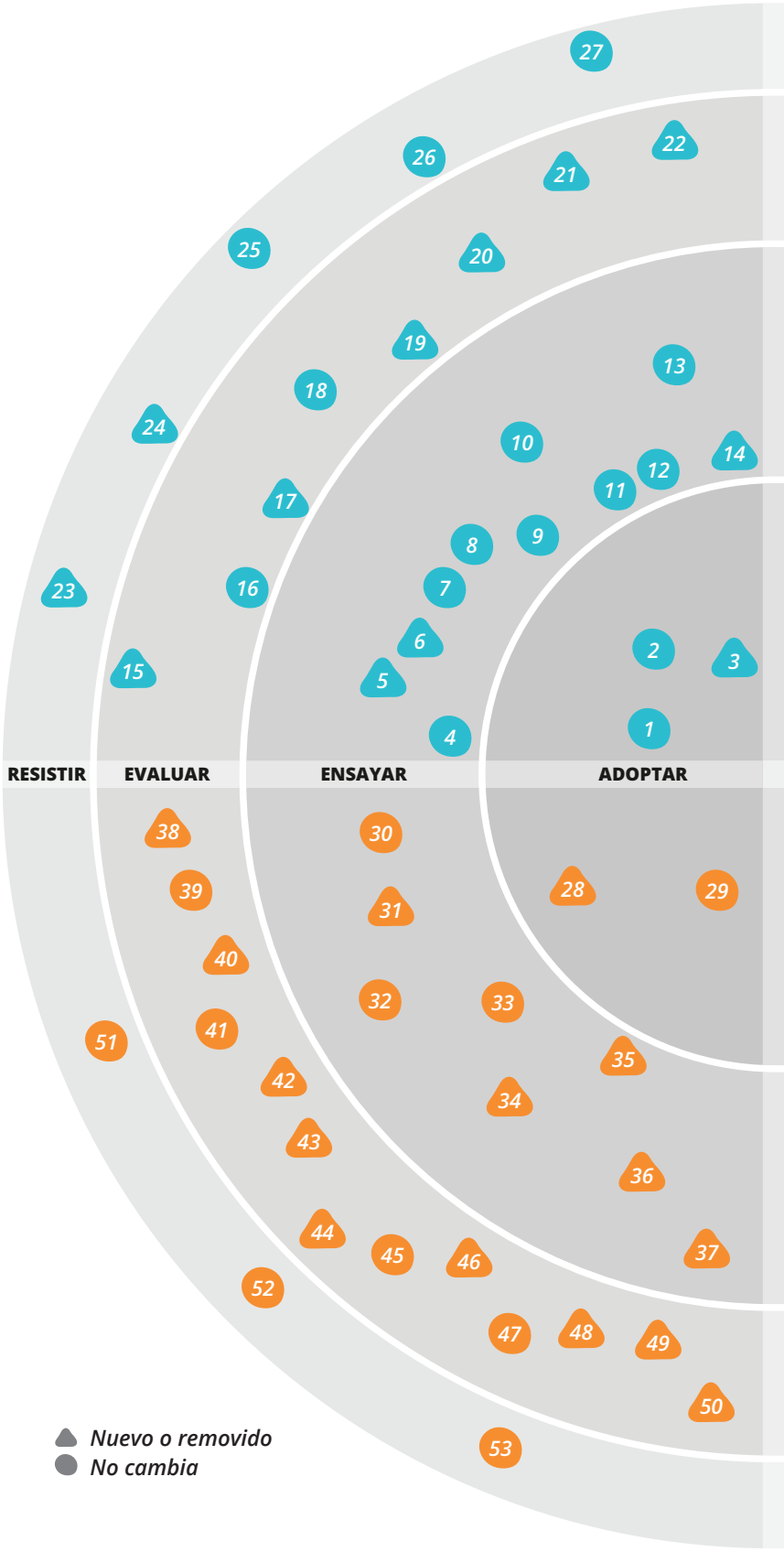
PLATAFORMAS

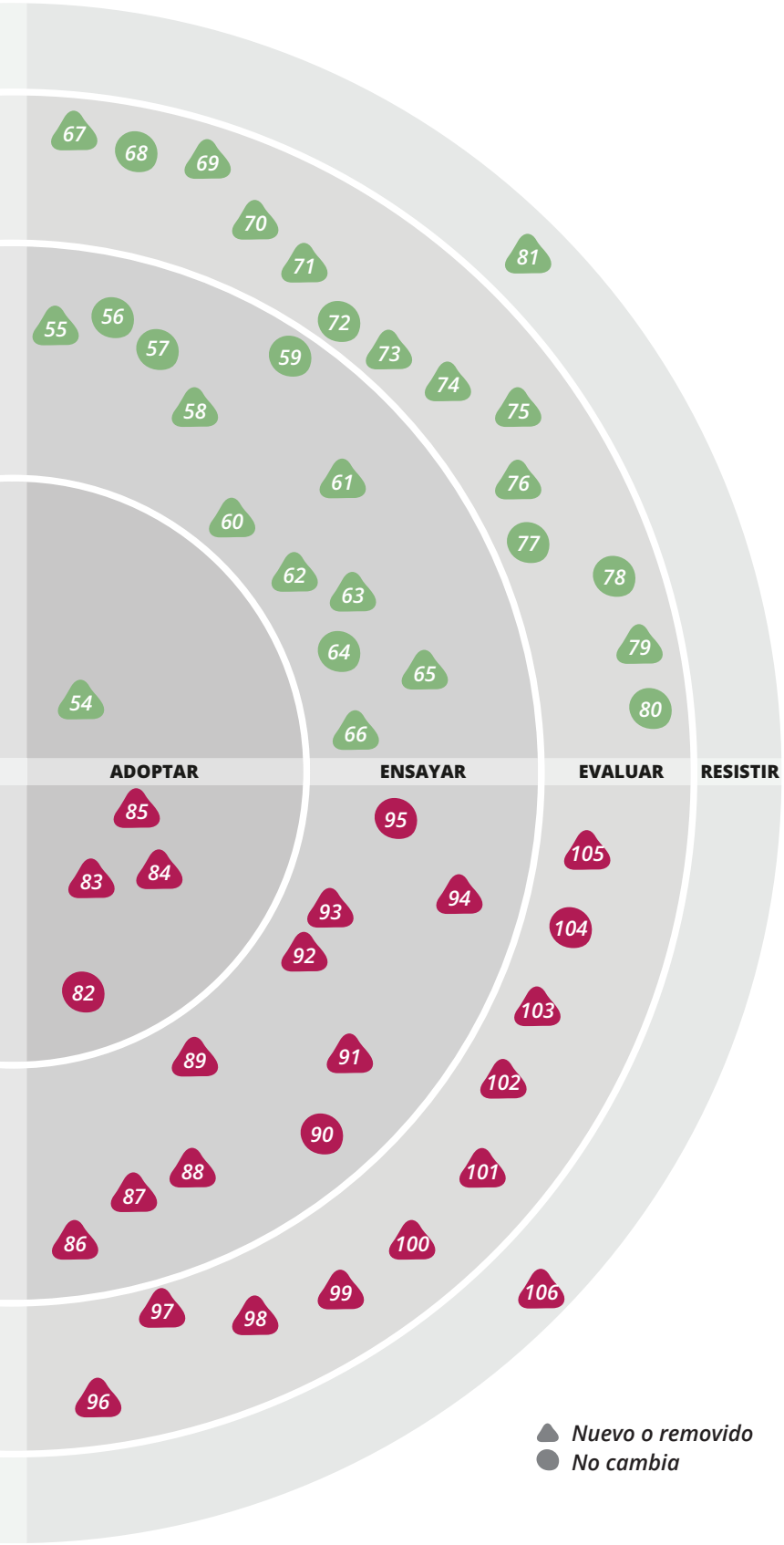
ADOPTAR
28. Docker
29. TOTP Two-Factor Authentication

ENSAYAR
30. Apache Mesos
31. AWS Lambda
32. H2O
33. HSTS
34. Kubernetes
35. Linux security modules
36. Pivotal Cloud Foundry nuevo
37. Rancher

EVALUAR
38. Amazon API Gateway nuevo
39. AWS ECS
40. Bluetooth Mesh nuevo
41. Ceph
42. Deflect nuevo
43. ESP8266 nuevo
44. MemSQL nuevo
45. Mesosphere DCOS
46. Nomad nuevo
47. Presto
48. Realm nuevo
49. Sandstorm nuevo
50. TensorFlow nuevo

RESISTIR
51. Application Servers
52. Over-ambitious API Gateways
53. Superficial private cloud





EL RADAR

HERRAMIENTAS

ADOPTAR
54. Consul

ENSAYAR
55. Apache Kafka
56. Browsersync
57. Carthage
58. Gauge
59. GitUp
60. Let's Encrypt
61. Load Impact nuevo
62. OWASP Dependency-Check nuevo
63. Serverspec nuevo
64. SysDig
65. Webpack nuevo
66. Zipkin

EVALUAR
67. Apache Flink nuevo
68. Concourse CI
69. Gitrob
70. Grasp nuevo
71. HashiCorp Vault nuevo
72. ievms
73. Jepsen nuevo
74. LambdaCD nuevo
75. Pinpoint nuevo
76. Pitest nuevo
77. Prometheus
78. RAML
79. Repsheet nuevo
80. Sleepy Puppy

RESISTIR nuevo
81. Jenkins como un pipeline de despliegue

LENGUAJES & FRAMEWORKS

ADOPTAR
82. ES6
83. React.js
84. Spring Boot
85. Swift

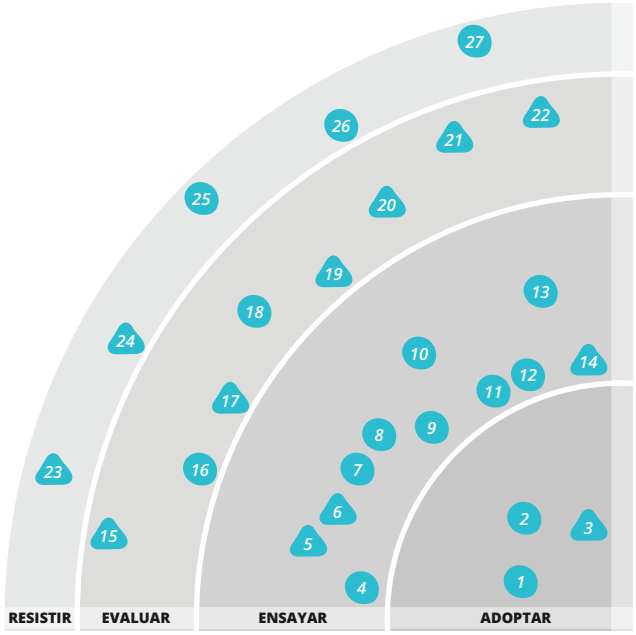
ENSAYAR
86. Butterknife nuevo
87. Dagger nuevo
88. Dapper nuevo
89. Ember.js
90. Enlive
91. Fetch nuevo
92. React Native
93. Redux nuevo
94. Robolectric nuevo
95. SignalR

EVALUAR
96. Alamofire nuevo
97. AngularJS
98. Aurelia nuevo
99. Cylon.js nuevo
100. Elixir
101. Elm
102. GraphQL nuevo
103. Immutable.js nuevo
104. OkHttp nuevo
105. Recharts

RESISTIR nuevo
106. JSPatch

TÉCNICAS

Debido a la gran cantidad de fallas en la seguridad de alto perfil en los últimos meses, los equipos de desarrollo de software ya no necesitan convencerse de que deben enfatizar el desarrollo de software seguro y tratar de forma responsable los datos de sus usuarios. Sin embargo, los equipos enfrentan un proceso arduo de aprendizaje, y puede ser abrumadora la gran cantidad de amenazas potenciales, desde el crimen organizado y el espionaje gubernamental hasta adolescentes que atacan sistemas solo por diversión ("por el lulz"). Los **Modelos de amenazas** presentan un conjunto de técnicas que le ayudarán a identificar y clasificar las amenazas potenciales, de forma temprana en el proceso de desarrollo. Es importante comprender que es solamente una parte de la estrategia para la gestión anticipada de amenazas. Cuando se usa junto con técnicas tales como establecer requisitos de seguridad multifuncionales para abordar los riesgos comunes de tecnologías que utilizan los proyectos y el uso de escaneadores de seguridad automatizados, los modelos de amenazas pueden ser un instrumento potente.



El uso de **bug bounties** (recompensas por errores) continúa creciendo en popularidad para muchas organizaciones, incluyendo empresas y entidades gubernamentales importantes. Un programa de recompensas por errores alienta a los participantes a identificar vulnerabilidades potencialmente dañinas a cambio de premios o reconocimiento. Las empresas tales como HackerOne y BugCrowd ofrecen servicios que pueden ayudar a organizaciones a manejar más fácilmente este proceso. Y estamos notando que estos servicios son adoptados con mayor frecuencia.

Un **Lago de datos** es un almacén de datos inmutables principalmente con datos "no procesados", que actúan como una fuente para analítica de datos. Si bien la técnica puede fácilmente usarse incorrectamente, la hemos utilizado exitosamente con clientes, por lo que alentamos que se use como prueba. Continuamos recomendando otros enfoques para colaboraciones operativas, limitando el uso de lagos de datos a la generación de reportes, temas analíticos y alimentación de datos hacia almacenes de datos (data mart).

Vemos la adopción continua y éxito de **arquitecturas reactivas**, siendo muy populares las extensiones de lenguaje reactivo y marcos de trabajo reactivos. (En esta edición del Radar agregamos varios blips de tal tipo.) Las interfaces de usuario, específicamente, se benefician en gran medida del estilo reactivo de programación. Las advertencias de la última vez siguen siendo aplicables. Las arquitecturas basadas en la transmisión asincrónica de mensajes introducen una complejidad y dificultan todavía más la comprensión de todo el sistema. Ya no es posible solo leer el código del programa para comprender qué hace el sistema. Recomendamos evaluar el rendimiento y las necesidades de agrandar su sistema antes de comprometerse con este estilo arquitectónico.

Nos hemos dado cuenta que las **Políticas de seguridad**

ADOPTAR
1. Decoupling deployment from release
2. Products over projects
3. Threat Modeling

ENSAYAR
4. BFF - Backend for frontends
5. Bug bounties
6. Data Lake
7. Event Storming
8. Flux
9. Idempotency filter
10. iFrames for sandboxing
11. NPM for all the things
12. Phoenix Environments
13. QA in production
14. Reactive architectures

EVALUAR
15. Content Security Policies
16. Hosted IDE's
17. Hosting PII data in the EU
18. Monitoring of invariants
19. OWASP ASVS
20. Serverless architecture
21. Unikernels
22. VR beyond gaming

RESISTIR
23. A single CI instance for all teams
24. Big Data envy
25. Gitflow
26. High performance envy/web scale envy
27. SAFe™

TÉCNICAS *continuación*

de contenido son un agregado útil a nuestro kit de herramientas de seguridad, cuando se trata de sitios web que extraen recursos de una variedad de contextos. La política define un conjunto de reglas acerca de dónde pueden provenir los componentes (y si se permitirán etiquetas scripts en línea). El navegador después se rehúsa a cargar o ejecutar JavaScript, CSS o imágenes que no cumplen con tales reglas. Cuando se utiliza junto con buenas prácticas, tales como codificación de salidas, provee una buena mitigación de ataques XSS. De forma interesante, el servicio opcional para publicar informes JSON de violaciones es cómo Twitter descubrió que los ISPs (proveedores de servicios de internet) estaban inyectando HTML o JavaScript en sus páginas.

En algunos países del mundo, vemos que varias agencias gubernamentales están intentando lograr un amplio acceso a información de identificación personal (IIP) privada. En la UE, la corte suprema ha invalidado los principios de Safe Harbor (Puerto Seguro) y se espera que no se permita tampoco Privacy Shield (Escudo de Privacidad), su sucesor. A la vez, el uso de computación en la nube está aumentando y todos los principales proveedores de servicios en la nube (Amazon, Google y Microsoft) ofrecen múltiples centros de datos y regiones en la Unión Europea. Por tanto, recomendamos que las empresas, especialmente aquellas con una base de usuarios globales, accedan a la opción de un puerto seguro para los datos de sus usuarios, que está protegido por las leyes de privacidad más avanzadas, por **Alojamiento de datos de IIP** en la UE.

Mientras más equipos de desarrollo incluyen temas de seguridad temprano en el ciclo de la vida de desarrollo, identificar requisitos para limitar los riesgos de seguridad puede parecer una tarea apabullante. Pocas personas tienen el conocimiento técnico requerido para identificar todos los riesgos que una aplicación puede enfrentar, y los equipos deben luchar para intentar decidir dónde comenzar. Confiar en marcos tales como **ASVS** (Estándar para la verificación de la seguridad de aplicaciones) de OWASP puede facilitar este proceso. A pesar ser algo largo, tiene una lista minuciosa de requisitos categorizados por funciones tales como autenticación, control de acceso y manejo y registro de errores, que pueden revisarse según haga falta. También es un recurso útil para encargados de pruebas, cuando llegue el momento de verificar el software.

La **arquitectura sin servidores** reemplaza a las máquinas virtuales de larga ejecución con capacidades de cómputo efímera, que comienzan a existir a pedido y desaparecen inmediatamente después de su uso. Entre los ejemplos, se incluyen Firebase y AWS Lambda. El uso de esta arquitectura puede mitigar algunas de las preocupaciones de seguridad tales como parches de seguridad y control de acceso de SSH, y puede permitir el uso mucho más eficiente de recursos informáticos. Estos sistemas cuestan muy poco para operar y tienen opciones para agrandar ya incorporadas (especialmente AWS Lambda). Una arquitectura de ejemplo pude ser una aplicación en JavaScript con componentes estéticos servidos por un CDN o S3 junto con llamadas AJAX de los que se encargan el enlace o gateway de API y Lambda. Aunque las arquitecturas sin servidores tienen beneficios importantes, hay algunas desventajas como las siguientes: Desplegar, gestionar y compartir código a través de servicios es más complejo, y la prueba local o fuera de línea es más difícil o casi imposible.

Con el continuo aumento del dominio del modelo de contenedor liderado por la adopción de Docker, creemos que vale la pena llamar la atención al continuo y rápido desarrollo en el espacio de **Unikernel**. Unikernels son los sistemas operativos de biblioteca de un solo uso que se pueden compilar al descender de lenguajes de alto nivel para ejecutarse directamente en los hipervisores utilizados por plataformas básicas y comunes en la nube. Prometen una serie de ventajas si se comparan con contenedores, como, el significativo tiempo de arranque ultra-rápido y un área pequeña de superficie de ataque. Muchos están todavía en la fase de de investigación, tales como Drawbridge de Microsoft Research, MirageOS y HaLVM, entre otros, pero pensamos que las ideas son muy interesantes y combinan bien con la técnica de la arquitectura sin servidores.

La idea de la realidad virtual ha existido durante más de 50 años y, gracias a mejoras sucesivas de la tecnología informática, se han promocionado y explorado muchas ideas. Creemos que casi alcanzamos un punto de inflexión, ahora. Las tarjetas gráficas modernas proveen suficiente poder de cómputo para graficar escenas detalladas y realistas en resoluciones altas para, por lo menos, dos cascos con auriculares de realidad virtual orientados al consumidor de manera simultánea, (el HTC Vive y el Oculus Rift) de Facebook están llegando al mercado. Estos auriculares son asequibles, tienen monitores de alta resolución y eliminan el retraso percibible de control de movimiento, que estaban causando dolores de cabeza y náusea, anteriormente. Los cascos con auriculares se enfocaban, principalmente, en juegos de video de aficionados, pero estamos convencidos que abrirán muchas opciones de r**ealidad virtual que va más allá de los juegos**, especialmente ahora que los enfoques de baja fidelidad, tales como

TÉCNICAS *continuación*

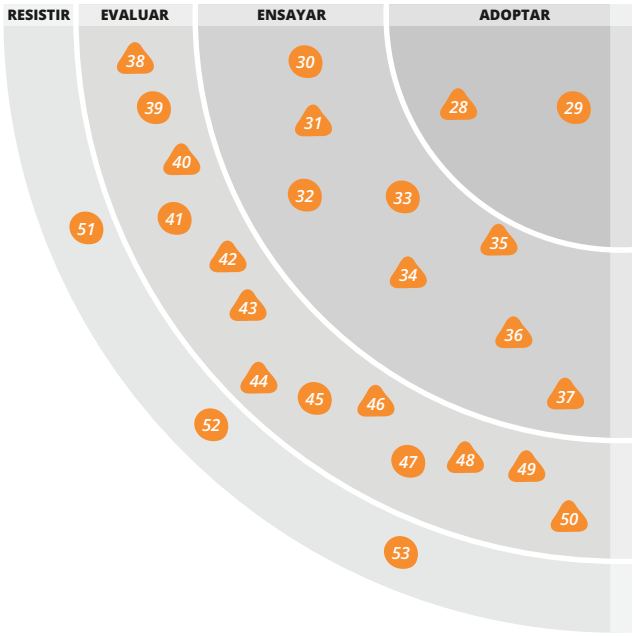
Google Cardboard, están impulsando un mayor conocimiento.

Podría haber la impresión que es más fácil gestionar **una sola instancia de CI (integración continua) para todos los equipos** porque les da un solo punto de configuración y monitoreo. Pero una instancia muy abultada que comparten todos los equipos de una organización podría causar muchos daños. Hemos visto que se pueden producir, con mayor frecuencia, problemas como la expiración del tiempo de versiones, conflictos de configuración y colas inmensas de generación de versiones. Tener un solo punto de falla puede interrumpir el trabajo de muchos equipos. Considere cuidadosamente las ventajas y desventajas entre estas dificultades y tener un solo punto de configuración. En organizaciones con varios equipos, es recomendable que se distribuyan las instancias de CI por equipos, con las decisiones con alcance a toda la empresa ya no basadas en una sola instalación de CI, sino más bien en la definición de pautas sobre la selección y configuración de instancias.

Aunque hemos comprendido desde hace mucho tiempo el valor de Big Data en el proceso de captar cómo las personas interactúan con nosotros, hemos notado una tendencia alarmante de **envidia de Big Data**: organizaciones utilizando herramientas complejas para manipular "conjuntos de datos que no son tan grandes". Los algoritmos distribuidos de mapeo-reducción son una técnica útil para conjuntos grandes de datos, pero muchos de estos podrían ser procesados fácilmente en una base de datos de un nodo, ya sea relacional o basada en grafos. Incluso con una cantidad superior de datos, generalmente, la mejor decisión es extraer el subconjunto de los datos que se requieren, que luego, con frecuencia, pueden procesarse en un solo nodo. Entonces, le alentamos a realizar una evaluación realista, antes de comenzar a usar clusters, y si se adecúan para un RAM, podría utilizar la opción más sencilla.

PLATAFORMAS

Seguimos entusiastas acerca de **Docker** ahora que está evolucionando de una herramienta a una plataforma compleja de tecnologías. A los equipos de desarrollo les encanta Docker, porque el formato de imágenes de Docker permite lograr la paridad más fácilmente entre el ambiente de desarrollo y el de producción, lo que sirve para lograr despliegues confiables. Encaja de forma natural en una aplicación con estilo de microservicios, como un mecanismo de empaquetamiento para servicios auto-contenidos. En el lado operativo, el soporte de Docker en herramientas de monitoreo (**Sensu**, **Prometheus**, **cAdvisor**, etc.), herramientas de orquestación (**Kubernetes**, **Marathon**, etc.) y herramientas de automatización de despliegue refleja la creciente madurez de la plataforma y su capacidad de estar lista para su uso en ambientes de producción. Sin embargo, presentamos esta advertencia. Hay la idea prevaleciente de que los contenedores de Docker y Linux, en general, son una "virtualización ligera", pero en tal caso no recomendaríamos usar Docker como un mecanismo de aislamiento seguro de procesos. No obstante, estamos prestando atención a la introducción de espacios de nombres (namespaces) y perfiles seccomp en la versión 1.10, en este aspecto.



Nuestros equipos siguen disfrutando el uso de **AWS Lambda** y están comenzado a utilizarlo para experimentar con arquitecturas sin servidores, que combinan Lambda con API Gateway para producir sistemas fácilmente escalables con una infraestructura invisible. Hemos enfrentado problemas graves al usar Java con funciones de Lambda, con demoras erráticas de hasta varios segundos, cuando se arranca el contenedor de Lambda. Recomendamos seguir usando JavaScript o Python por ahora.

Kubernetes es la respuesta de Google al problema de desplegar contenedores a un grupo de servidores (cluster), que se está convirtiendo en un escenario cada vez más común. No es la solución utilizada por Google internamente, pero es un proyecto de fuente abierta que se inició en Google y ha recibido una cantidad razonable de contribuciones externas. Como mencionamos Kubernetes en el Radar previo, nuestras impresiones iniciales positivas se confirmaron y estamos notando un uso exitoso de Kubernetes en ambientes de producción de nuestros clientes.

En versiones iniciales del Radar, hemos destacado el valor de los **módulos de seguridad de Linux**, al describir cómo permiten a las personas pensar en reforzar la seguridad de servidores como parte de su flujo de trabajo de desarrollo. Más recientemente, gracias a

ADOPTAR
28. Docker
29. TOTP Two-Factor Authentication

ENSAYAR
30. Apache Mesos
31. AWS Lambda
32. H2O
33. HSTS
34. Kubernetes
35. Linux security modules
36. Pivotal Cloud Foundry
37. Rancher

EVALUAR
38. Amazon API Gateway
39. AWS ECS
40. Bluetooth Mesh
41. Ceph
42. Deflect
43. ESP8266
44. MemSQL
45. Mesosphere DCOS
46. Nomad
47. Presto
48. Realm
49. Sandstorm
50. TensorFlow

RESISTIR
51. Application Servers
52. Over-ambitious API Gateways
53. Superficial private cloud

PLATAFORMAS *continuación*

los contenedores de LXC y Docker que se adjuntan de forma predeterminada a perfiles de AppArmor en ciertas distribuciones de Linux, muchos equipos se han visto forzados a aprender cómo funcionan estas herramientas. En el caso de que los equipos utilicen imágenes de contenedores para ejecutar cualquier proceso que no crearon ellos mismos, estas herramientas les ayudarán a evaluar las preguntas sobre quién tiene acceso a cuáles recursos en el servidor compartido y las capacidades que estos servicios contenidos tienen. De esta forma serán conservadores al gestionar los distintos niveles de acceso.

En el espacio PaaS (plataforma como servicio) ha habido muchas novedades, desde que mencionamos Cloud Foundry en 2012. Aunque hay varias distribuciones del núcleo de fuente abierta, nos ha impresionado la oferta y ecosistema ensamblados como **Pivotal Cloud Foundry**. Aunque esperamos una convergencia continua entre el enfoque no estructurado (Docker, Mesos, Kubernetes, etc.) y el estilo más estructurado y previamente empaquetado (buildpack) ofrecido por Cloud Foundry y otros, vemos un beneficio real para organizaciones que están dispuestas a aceptar los límites y tasa de evolución para adoptar PaaS. La velocidad de desarrollo es de particular interés, ya que proviene de la simplificación y estandarización de la interacción entre equipos de desarrollo y operaciones.

En el espacio emergente de Contenedores como Servicio (CaaS) hemos visto mucho movimiento recientemente, proporcionando una opción útil que combina IaaS básico (la Infraestructura como un servicio) y la más controversial PaaS (Plataforma como un servicio). Si bien **Rancher** ha creado menos ruido que otros participantes, hemos disfrutado de la sencillez que contribuye a contenedores de Docker en ambientes de producción. Puede usarse de forma independiente como una solución completa o junto a otras herramientas tales como Kubernetes.

Amazon API Gateway es la oferta de Amazon que permite a los desarrolladores exponer servicios de API a clientes de Internet, ofreciendo las características usuales de API gateway tales como gestión de tráfico, monitoreo, autenticación y autorización. Nuestros equipos han estado usando este servicio como proxy a otras funcionalidades de AWS tales como AWS Lambda como parte de arquitecturas sin servidores. Continuamos monitoreando en caso de que se presenten retos por medio de API gateways demasiado ambiciosas, pero en esta etapa la oferta de Amazon parece lo suficientemente ligera como para evitar estos problemas.

Aunque muchos despliegues hacia dispositivos inteligentes dependen de conectividad de Wi-Fi, hemos notado el éxito con redes de **Bluetooth Mesh** que no requieren un hub ni gateway. Con un mejor uso de energía comparado con y una mejor adopción por parte de los teléfonos inteligentes que con ZigBee, Bluetooth LE, desplegado como una malla de autocuración, provee enfoques interesantes y nuevos para conectar las redes de área locales de dispositivos. Todavía estamos esperando que surja un enfoque formal de Bluetooth SIG, pero ya hemos tenido despliegues exitosos. Nos gusta especialmente el hecho de que no se necesita infraestructura para crear una red descentralizada, pero todavía retenemos la opción de "mejorar progresivamente" el sistema al agregar una gateway y servicios en la nube.

Deflect es un servicio de código libre que protege a ONGs, activistas y empresas de medios independientes de ataques de negación de servicio distribuidos (DDoS). De forma similar a un CDN comercial, usa una caché en proxy-reverso distribuida, esconde las direcciones IP de servidores y bloquea el acceso público a URLs de administración. Un esfuerzo particular se realiza para combatir a los botnet usados típicamente para censurar de forma extrajudicial a voces independientes.

Nuestra creciente cantidad de hackers de hardware está emocionada por el microcontrolador de Wi-Fi **ESP8266**. En vez de una innovación tecnológica específica, es la combinación de un precio bajo y un pequeño factor de forma que ha provocado un punto de inflexión en las ideas de las personas acerca de la factibilidad de crear dispositivos personalizados de hardware. Sus principales características son: capacidades de conexión Wi-Fi (puede actuar como estación de trabajo, punto de acceso o una combinación de ambos), menos uso de energía, hardware abierto, capacidad de programación en el SDK y el IDE de Arduino, capacidad de programación en Lua, el amplio apoyo de la comunidad y bajo costo comparado con otros módulos IoT.

Tal como lo predice la Ley de Moore, continuamos aumentando la capacidad de los sistemas de computación y disminuyendo su costo, y las técnicas de procesamiento nuevas se vuelven posibles aún cuando solamente hace unos pocos años parecían fuera del alcance. Una de estas técnicas es la base de datos en memoria. En vez de usar discos lentos o SSDs relativamente lentos para almacenar datos, podemos mantenerlos en memoria para lograr un alto

PLATAFORMAS *continuación*

rendimiento. Una base de datos en memoria de tal tipo, MemSQL, está causando sensación porque es escalable horizontalmente a lo largo cluster y provee un lenguaje de consultas conocido basado en SQL. **MemSQL** también se conecta con Spark para generar analíticas contra datos en tiempo real, en vez de datos obsoletos de un depósito.

HashiCorp continúa creando software interesante. El más reciente que nos llama la atención es **Nomad**, que está compitiendo en el ámbito de planificadores, que es cada va más poblado. Los puntos importantes para lograr la venta incluyen no solo que se limita a contenerizar cargas de de trabajo sino que opera en despligues en multiples centros de datos / de varias regiones..

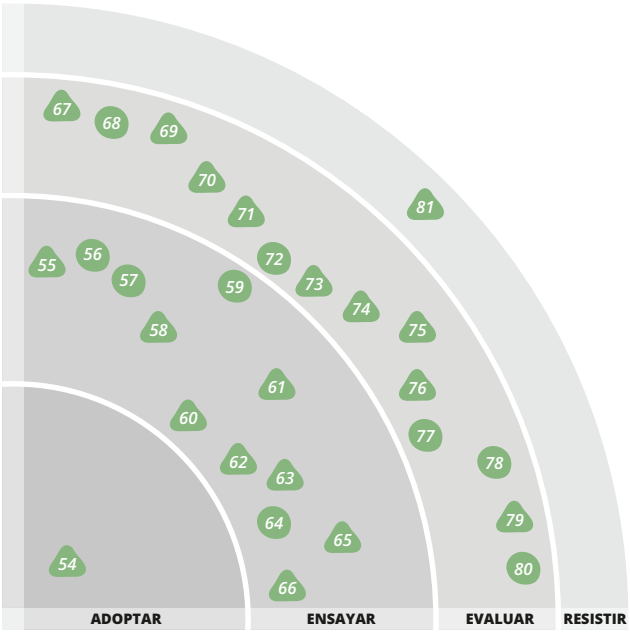
Realm es una base de datos diseñada para usarse en dispositivos móviles, con su propio motor de persistencia para lograr un gran rendimiento. Realm se comercializa como un reemplazo de SQLite y Core Data, y a nuestros equipos les ha encantado usarla. Tome en cuenta que las migraciones no son tan sencillas como parecen según la documentación de Realm. De todas maneras, Realm nos emociona y sugerimos que lo revise.

Para personas que deseen beneficiarse de las herramientas de colaboración basadas en la nube pero no quieren, de forma inadvertida, "convertirse en el producto" de uno de los grandes proveedores de la nube, **Sandstorm** les provee una interesante alternativa de código libre con el potencial de auto-alojamiento. Es de interés particular el enfoque de aislamiento, mediante el cual la contenerización se aplica por documento y no por aplicación. Además se agregó una lista blanca de llamadas al sistema para proteger aún más el sandbox.

TensorFlow de Google es una plataforma de "aprendizaje de máquina" que puede usarse para todo desde investigaciones hasta ambientes de producción y se ejecutará en hardware desde una CPU de dispositivos móviles, hasta un grupo grande de computadoras equipadas con una GPU. Es una plataforma importante porque permite que se implementen algoritmos de aprendizaje profundo de forma más accesible y conveniente. A pesar de la novelería, sin embargo, TensorFlow no es realmente algo nuevo desde un punto de vista de algoritmo. Todas estas técnicas han estado disponibles en el dominio público en el entorno académico, por algún tiempo. También es importante darse cuenta que casi todos los negocios no están ejecutando ni siquiera procesos analíticos predictivos básicos y que un salto al "aprendizaje profundo" posiblemente no tendrá mucho sentido en la mayoría de conjuntos de datos. Sin embargo para aquellos que tienen un problema y un conjunto de datos correctos, TensorFlow es un kit de herramientas útil.

HERRAMIENTAS

Hemos movido **Consul**, una herramienta de detección de servicios que admiten mecanismos de detección basados en DNS y HTTP, hacia Adopt. Esto va más allá de otras herramientas de detección; ya que provee chequeos de salud personalizables para los servicios que se encuentren registrados, lo que se garantiza que se marquen de forma correcta las instancias no saludables. Han surgido más herramientas para trabajar con Consul, para convertirlo en algo incluso más poderoso. Consul Template permite que los archivos de configuración sean llenados con información obtenida desde Consul, lo que facilita asuntos como el equilibrio de carga del lado del cliente mediante el uso de mod_proxy. En el mundo de Docker, registrator puede registrar automáticamente los contenedores Docker a medida que aparecen en Consul con muy poco esfuerzo, lo que facilita las gestiones de configuraciones basadas en contenedores. Todavía debería pensar bien y detenidamente acerca de si necesita una herramienta como esta o si requiere algo más sencillo, pero si decide que necesita la detección de servicios, no tendrá problemas si usa Consul.



Muchas organizaciones en estos días están revisando minuciosamente las nuevas arquitecturas de datos que captan información como secuencias inmutables de eventos a escala. **Apache Kafka** continúa impulsándose como un protocolo de mensajería de código abierto que provee una solución para publicar eventos ordenados, a grandes cantidades de consumidores independientes y livianos. La configuración de Kafka no es fácil, pero nuestros equipos están informando de experiencias positivas con este framework.

Gauge es una herramienta liviana de automatización de pruebas que se puede usar en varias plataformas. Las especificaciones se pueden escribir usando Markdown sin atarse a un formato pre establecido, de esta forma los casos de prueba pueden ser escritos usando el lenguaje del negocio, lo cual es totalmente opuesto a usar el más común, pero restringido formato "given-when-then". El soporte de lenguaje y de IDE se implementan como plug-in de una implementación de núcleo único, lo que permite a los encargados de pruebas utilizar los mismos IDE que el resto del equipo, con capacidades potentes tales como la auto-completación de código y refactoreo. Esta herramienta, cuyo código fuente fue liberado por ThoughtWorks, también permite la ejecución en paralelo, sin ninguna personalización adicional, para todas las plataformas soportadas.

Let's Encrypt fue publicado en la edición más reciente de Radar, y a partir de diciembre de 2015 este proyecto ha cambiado su estado de un beta privado a público. Esto significa que los usuarios ya no necesitan una invitación para usarlo. Let's Encrypt otorga acceso a un mecanismo más sencillo para obtener y gestionar certificados, para un conjunto más grande de usuarios que están buscando una forma de proteger sus sitios

ADOPTAR
54. Consul

ENSAYAR
55. Apache Kafka
56. Browsersync
57. Carthage
58. Gauge
59. GitUp
60. Let's Encrypt
61. Load Impact
62. OWASP Dependency-Check
63. Serverspec
64. SysDig
65. Webpack
66. Zipkin

EVALUAR
67. Apache Flink
68. Concourse CI
69. Gitrob
70. Grasp
71. HashiCorp Vault
72. ievms
73. Jepsen
74. LambdaCD
75. Pinpoint
76. Pitest
77. Prometheus
78. RAML
79. Repsheet
80. Sleepy Puppy

RESISTIR
81. Jenkins as a deployment pipeline

HERRAMIENTAS *continuación*

web. Esto es un gran paso hacia adelante en lo que respecta a seguridad y privacidad. Esta tendencia ya ha comenzado en ThoughtWorks y muchos de nuestros proyectos tienen ahora certificados verificados por Let's Encrypt.

Load Impact es una herramienta SaaS para pruebas de carga, que puede generar cargas muy realistas de hasta 1,2 millones de usuarios simultáneos. Permite registrar y reproducir las interacciones web usando un plug-in para Chrome que simula las conexiones de red, ya sea para usuarios móviles o de computadoras de escritorio, y genera carga desde hasta 10 lugares distintos alrededor del mundo. Aunque no es la única herramienta para pruebas de carga bajo demanda que hemos usado, también nos gusta BlazeMeter. Y nuestros equipos están muy entusiasmados acerca de Load Impact.

En un mundo lleno de librerías y herramientas que simplifican la vida de muchos desarrolladores de software, las deficiencias de su seguridad se han vuelto visibles y han aumentado la superficie de vulnerabilidad de las aplicaciones que las usan. **OWASP Dependency-Check** identifica automáticamente los problemas de seguridad potenciales en el código y comprueba si hay vulnerabilidades divulgadas públicamente y conocidas. Después usa métodos para actualizar constantemente la base de datos de vulnerabilidades públicas. Dependency-Check tiene algunas interfaces y plugins para automatizar esta verificación en Java y .NET (que hemos usado exitosamente) además de Ruby, Node.js y Python.

En el pasado hemos incluido Provisioning Testing automatizados como una técnica recomendada, y en esta edición destacamos a **Serverspec** como una herramienta popular para implementar estas pruebas. A pesar de que esta herramienta no es nueva, hemos notado que se ha vuelto más común su uso ahora que equipos multifuncionales distribuidos han asumido la responsabilidad de provisión de infraestructura. Serverspec se basa en la librería de Ruby RSpec e incluye un conjunto completo de componentes utilitarios para asegurar que la configuración del servidor es la correcta.

Webpack se ha solidificado como un componente que combina módulos de JavaScript de fácil uso. Gracias a su lista de cargadores siempre creciente, provee un único árbol de dependencias para todos tus recursos estáticos, lo que permite una manipulación flexible de JavaScript, CSS, etc., y minimiza lo que debe enviarse al navegador y el momento de hacerlo. Es particularmente importante la integración suave con AMD, CommonJS y módulos de ES6. Ha permitido que los equipos trabajen con ES6 y transcompilar, de forma transparente, (usando Babel) a versiones anteriores, para lograr compatibilidad con los navegadores. Muchos de nuestros equipos también aprecian Browserify, que cubre un espacio similar pero se enfoca más en lograr que los módulos de Node.js estén disponibles para uso por el cliente.

El desarrollo en **Zipkin** una herramienta para medir el tiempo tiempo de latencia de punta a punta de muchas peticiones lógicas, ha continuado rápidamente y desde mediados de 2015 se ha trasladado a la organización openzipkin/zipkin en GitHub. Ahora, hay bindings (vínculos) para Python, Go, Java, Ruby, Scala y C#. Y hay imágenes de Docker disponibles para aquellos que quieren comenzar rápido a usarla. A nosotros nos sigue gustando esta herramienta, hay una comunidad activa y creciente involucrada en su uso, y se está facilitando cada vez más su implementación por lo que continúa siendo una opción muy válida.

Apache Flink es una plataforma de nueva generación para el procesamiento escalable y distribuido de lotes y streams de datos. En su núcleo tiene a un motor de transmisión de datos como streams. También soporta operaciones tabulares (similares a SQL), de procesamiento de grafos, y de aprendizaje de máquina (machine learning). Apache Flink se destaca por sus capacidades llenas de opciones para el procesamiento de streams como eventos temporizados, ricas operaciones de ventana en base a streams, tolerancia a fallos y semántica de exactamente una vez. Si bien no ha llegado a la versión 1.0, ha logrado despertar bastante interés en la comunidad debido a las innovaciones en el procesamiento de flujos, manejo de memoria, gestión de estados y la sencillez de la configuración.

Los atacantes siguen usando software automatizado para investigar depósitos públicos de GitHub para encontrar credenciales de AWS y generar instancias de EC2 para extraer Bitcoins o para otros propósitos nefastos. A pesar de que es cada vez más común la adopción de herramientas como git-crypt y Blackbox para guardar de forma segura secretos como contraseñas y tokens de acceso en repositorios de código, sigue siendo muy común que se guarden secretos sin protección. Tampoco es extraño que los desarrolladores guarden los secretos de proyectos en repositorios personales. **Gitrob** puede ayudar a minimizar el daño de revelar secretos. Esta herramienta escanea los repositorios de GitHub de una organización y coloca marcas en todos los archivos que tienen

HERRAMIENTAS *continuación*

información sensible que no deberían haberse enviado al repositorio. La versión actual de la herramienta tiene algunas limitaciones. Solo puede usarse para escanear organizaciones públicas en GitHub y sus miembros. No revisa el contenido de los archivos, ni revisa el historial completo de cambios y revisa por completo todos los repositorios cada vez que se ejecuta. A pesar de estas limitaciones, puede ser una herramienta reactiva útil para ayudar a alertar a equipos antes de que sea demasiado tarde. Debe considerarse como complemento de herramientas proactivas como Talisman.

Esta pequeña herramienta de refactorización JavaScript de línea de comandos llamada **Grasp** nos sorprendió colectivamente. Está muy por delante de jugar con sed y grep, porque provee un conjunto abundante de selectores y opera en base a un árbol abstracto de sintaxis. Esta es una adición util a la caja de herramientas en nuestra búsqueda constante para tratar a JavaScript como un lenguaje de primera clase.

Disponer de una forma de administrar de manera segura datos secretos del proyecto es un problema cada vez más grande. La idea tradicional de solo tener un archivo con secretos o variables de entorno está volviéndose difícil de gestionar, especialmente en ambientes con varias aplicaciones como microservicios o ambientes de micro-contenedores, en donde las aplicaciones deben acceder a una gran cantidad de secretos. **HashiCorp Vault** es una herramienta que parece ser prometedora e intenta resolver el problema mediante mecanismos para acceder de forma segura a secretos por medio de una interfaz unificada. Tiene algunas características que facilitan la vida, tales como cifrado y la generación automática de secretos para herramientas conocidas, entre otras.

Por el mayor uso de bases de datos NoSQL y el crecimiento de la popularidad de enfoques políglotas para la persistencia, los equipos tienen ahora muchas opciones a la hora de decidir cómo almacenar sus datos. A pesar de que esto ha traído muchas ventajas, el comportamiento del producto combinado a redes inestables puede introducir problemas sutiles (y no tan sutiles) que, con frecuencia, no son bien comprendidos inclusive por los mismos desarrolladores. Las herramientas de **Jepsen** y el blog complementario se han convertido en la referencia de hecho para cualquier persona que desee comprender cómo reaccionan en condiciones adversas las distintas tecnologías de base de datos y de colas. De forma esencial, el enfoque en las pruebas, que incluye a clientes transaccionales, destaca posibles modos de fallas que pueden ser de utilidad para muchos equipos que desarrollan microservicios.

LambdaCD brinda a los equipos una manera de definir procesos de entrega continua en Clojure. Esto trae los beneficios de la Infraestructura como código a la configuración de servidores de CD, tales como la gestión de control de código fuente, pruebas unitarias, refactorización y la reutilización de código. En el espacio de "procesos secuenciales (pipelines) como código", LambdaCD se destaca por ser liviano, auto-contenido y totalmente programable, lo que permite a los equipos trabajar en sus propios pipelines de la misma manera que lo hacen con su código.

Los equipos que usan Phoenix Server o las técnicas del Phoenix Environment no han encontrado un apoyo de las herramientas de Gestión de Rendimiento de Aplicaciones (APM del inglés Application Performance Management). Sus modelos de licencias, que se basan en premisas de hardware con cantidades limitadas y de larga ejecución y la dificultad de tratar con hardware efímero, han logrado que sea demasiado complicado como para valer la pena. Sin embargo, los sistemas distribuidos requieren monitoreo y en algún punto algunos equipos reconocen la necesidad de usar una herramienta de APM. Pensamos que **Pinpoint**, una herramienta de fuente abierta en este espacio, vale la pena investigarse como una alternativa de AppDynamics y Dynatrace. Pinpoint se desarrolló en Java, con plugins disponibles para muchos servidores, bases de datos y marcos de desarrollo. Aunque pensamos que se puede lograr mucho al utilizar una combinación de otras herramientas de código libre, —Zipkin, por ejemplo— vale la pena considerar Pinpoint, si está buscando un APM.

Pitest es una herramienta de análisis de cobertura de pruebas para Java que usa una técnica de pruebas de mutación. El análisis tradicional de cobertura de pruebas tiende a medir la cantidad de líneas que las pruebas ejecutan. Por tanto, solo puede identificar el código que definitivamente no fue sometido a pruebas. Por otro lado, las pruebas de mutación intentan comprobar la calidad de aquellas líneas que su código de prueba ejecuta y que aún así podrían tener errores generales. Varios problemas pueden encontrarse de esta manera, lo que ayudarán al equipo a medir y desarrollar un conjunto de pruebas saludable. Casi todas las herramientas de este

HERRAMIENTAS *continuación*

tipo tienen la tendencia de ser lentas y difíciles de usar, pero Pitest ha demostrado tener un mejor rendimiento, es fácil de configurar y su soporte técnico es muy activo.

Los ataques contra los sitios web usando bots se están volviendo cada vez más sofisticados. Identificar a estos piratas y su comportamiento es la meta del proyecto **Repsheet**. Este un plug-in para Apache o NGINX que registra las actividades de los usuarios, genera huellas digitales para estos usando reglas predefinidas o establecidas por usuarios y luego permite que se tomen las acciones, incluyendo la capacidad de bloquear a los posibles atacantes. Incluye un utilitario que visualiza a los piratas actuales. Esto pone en las manos de los miembros de los equipos de desarrollo la capacidad de gestionar amenazas basadas en bots, lo que aumenta la generación de conciencia entorno a la seguridad en los equipos. Nos gusta esto al ser un buen ejemplo de una herramienta sencilla que resuelve un problema muy real pero a veces invisible—los ataques basados en bot.

Sabemos que nos adentramos en un territorio peligroso en este caso, porque desarrollamos una herramienta que compite con Jenkins, pero sentimos que tenemos que abordar un problema que subsiste. Las herramientas de integración continua tales como CruiseControl y Jenkins son valiosas para el desarrollo de software, pero mientras aumenta la complejidad del proceso de generación de versiones, se requiere algo adicional a la integración continua. Se requiere un pipeline de despliegue. Con frecuencia, notamos que los desarrolladores intentan usar **Jenkins como un pipeline** de despliegue con la ayuda de plug-ins, pero nuestra experiencia es que estas comienzan a complicarse rápidamente. Jenkins 2.0 presenta "Pipeline como código", pero continúa con el modelo de pipelines que usa plug-ins y falla al no cambiar el núcleo del producto Jenkins para modelar pipelines de forma directa. Conforme a nuestra experiencia, las herramientas que se crean basadas en una representación de primera clase de pipelines de despliegue son mucho más idóneas. Esto es lo que nos hizo reemplazar CruiseControl con GoCD. Ahora notamos que hay varios productos que adoptan pipelines de despliegue, que incluyen ConcourseCI, LambdaCD, Spinnaker, Drone y GoCD.

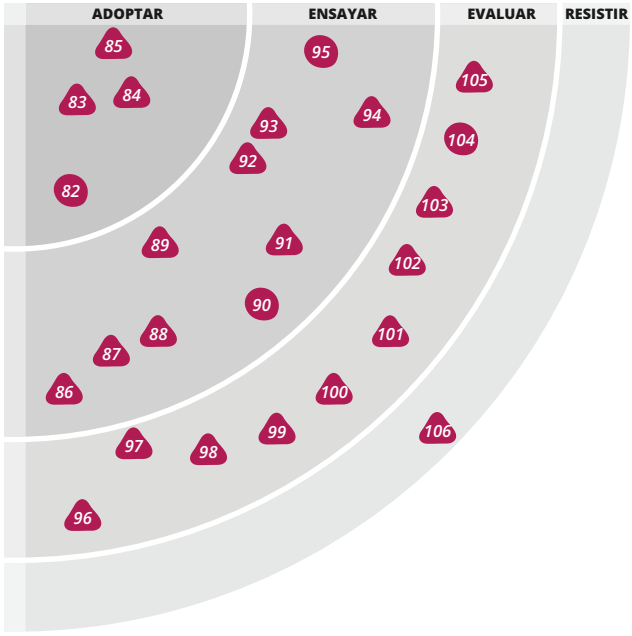
LENGUAJES & FRAMEWORKS

En la avalancha de marcos de desarrollo en JavaScript para la implementación de interfaces de usuario (front-end), **React.js** se destaca debido a su diseño alrededor de un flujo de datos reactivo. Al permitir la vinculación (binding) de datos en un solo sentido, simplifica, en gran medida, la lógica de generación y evita muchos de los problemas que se asocian, comúnmente, de forma negativa, a aplicaciones desarrolladas en otros marcos de trabajo. Nos hemos dado cuenta de los beneficios de React.js en una creciente cantidad de proyectos, grandes y pequeños, y a la vez continuamos preocupados acerca del estado y el futuro de otros marcos de trabajo populares como AngularJS. Esto ha hecho que React.js se convierta en nuestra opción preferida por defecto para desarrollo de interfaces en JavaScript.

Se ha trabajado mucho con **Spring Boot** para disminuir la complejidad y dependencias, que en gran parte alivia nuestras dudas previas. Si se encuentra en un ecosistema con Spring, y se está desplazando hacia microservicios, Spring Boot es ahora la opción obvia. Para quienes no están en un ecosistema Spring, Dropwizard también merece una consideración seria.

Swift es ahora nuestra opción predeterminada para desarrollo de aplicaciones en el ecosistema de Apple. Gracias al lanzamiento de Swift 2, el lenguaje alcanzó el nivel de madurez suficiente para proveer de la estabilidad y el rendimiento requerido por casi todos los proyectos. Una gran cantidad de librerías que ayudan al desarrollo de iOS, SwiftyJSON, Quick, etc., han migrado ahora a Swift, que es hacia donde se deberían dirigir las demás aplicaciones. Swift se ha vuelto de fuente abierta y vemos que ahora hay una comunidad de desarrolladores dedicados a la mejora continua del desarrollo en iOS.

Butterknife es una librería que utiliza métodos y propiedades para la inyección de vistas. Permite la inyección de objetos arbitrarios, vistas y observadores garantizando un código más limpio con menor cantidad de código de integración para el desarrollo de aplicaciones en Android. Con Butterknife, se



pueden agrupar vistas múltiples en una lista o arreglo con acciones comunes aplicadas a las vistas de forma simultánea, sin depender excesivamente de configuraciones en XML. Nuestros equipos de proyectos han usado esta librería y se han beneficiado de su sencillez y facilidad de uso.

Debido a la creciente necesidad de aplicaciones basadas en Android, **Dagger** ofrece un framework de inyección de dependencias completamente estático, y en tiempo de compilación. La implementación generada de forma estricta y no dependiente de soluciones basadas en reflexión abordan muchos de los problemas comunes de rendimiento y desarrollo, convirtiéndola en una solución idónea para el implementación de aplicaciones para Android. Con Dagger, existe una traceabilidad completa con fácil depuración debido a la disponibilidad de la pila de llamadas para el aprovisionamiento y la creación.

Dapper es un ORM minimalista y liviano para .NET. En vez de intentar escribir consultas en SQL, Dapper mapea consultas de SQL a objetos dinámicos. Aunque

ADOPTAR
82. ES6
83. React.js
84. Spring Boot
85. Swift

ENSAYAR
86. Butterknife
87. Dagger
88. Dapper
89. Ember.js
90. Enlive
91. Fetch
92. React Native
93. Redux
94. Robolectric
95. SignalR

EVALUAR
96. Alamofire
97. AngularJS
98. Aurelia
99. Cylon.js
100. Elixir
101. Elm
102. GraphQL
103. Immutable.js
104. OkHttp
105. Recharts

RESISTIR
106. JSPatch

LENGUAJES & FRAMEWORKS *continuación*

no es nuevo en el mercado, Dapper ha logrado un crecimiento continuo en su aceptación dentro de los equipos en ThoughtWorks que trabajan con .NET. Para el desarrollador en C#, elimina algo del trabajo tedioso de mapear consultas relacionales a objetos, mientras que continúa permitiendo el control completo sobre SQL o procedimientos almacenados.

Ember.js ha desarrollado un apoyo adicional basado en experiencias de proyectos y es obviamente un rival fuerte en el campo del marco de trabajo de aplicaciones de JavaScript. Ember es aclamado por su experiencia para el desarrollador, con muchos menos sorpresas que otros marcos de trabajo tales como AngularJS. El conjunto de herramientas de versiones de Ember CLI, que es de convención sobre configuración y el soporte a ES6 también lograron comentarios positivos.

Nuestros equipos se están alejando de JQuery o XHR sin procesar para llamadas remotas de JavaScript. Ahora, más bien, están usando el nuevo API Fetch y el polyfill **Fetch** , de forma específica. La semántica no se modificó pero hay un soporte más ordenado a promesas y CORS. Estamos notando que este es el nuevo enfoque de facto.

Estamos viendo un éxito continuo cuando se usa **React Native** para el desarrollo rápido de móviles de múltiples plataformas. A pesar de algo de agitación ahora que experimenta un desarrollo continuo, las ventajas de la integración trivial entre código y vistas nativas y no nativas, el rápido ciclo de desarrollo (recarga instantánea, depuración en Chrome, y distribución FlexBox) y crecimiento general del estilo React nos están convenciendo. Al igual que sucede con muchos marcos de trabajo, se debe tener cuidado en contar con código bien estructurado, pero el uso minucioso de una herramienta como Redux realmente nos ayuda, en tales casos.

Redux es una herramienta madura y muy buena que ha ayudado a muchos de nuestros equipos a replantear lo que piensan acerca de gestionar el estado de las aplicaciones del lado del cliente. Si se usa un enfoque Flux-style, esto permite una arquitectura de máquina de estados sin conexión directa, sobre la que se puede razonar fácilmente. Consideramos que es un buen compañero para algunos marcos de trabajo favoritos de JavaScript, tales como Ember y React.

En el mundo de desarrollo de aplicaciones Android, Robolectric es un marco de trabajo de pruebas unitarias que está siendo usado por varios equipos dentro de nuestra comunidad técnica. Ofrece la mejor opción entre las disponibles para crear pruebas unitarias reales que se amplíen o interactúen directamente con componentes de Android, y que admiten el uso de pruebas de JUnit. No obstante, presentamos una advertencia, debido a que es una implementación del SDK de Android, podría haber problemas específicos con dispositivos para algunas pruebas pasan en **Robolectric**. Para imitar todas las dependencias de Android y garantizar que solo se hagan pruebas de sistemas en modo de prueba, se requerirá mucho código completo, y este marco de trabajo logra abordar esto de forma eficaz.

Decodificar y manejar redes en aplicaciones iOS han sido areas complejas por muchos años. Ha habido muchas librerías e intentos de resolver este problema. Parece que **Alamofire** es la librería más robusta y fácil de usar para desarrolladores para la decodificación de JSON. Fue desarrollada por el mismo creador que su contraparte de Objective-C (AFNetworking), que se usó mucho en la época de Objective-C.

Aunque hemos completado exitosamente muchos proyectos usando **AngularJS** y estamos viendo una aceleración de la adopción en ambientes corporativos, hemos decidido mover Angular de regreso al anillo de Evaluar en esta edición de Radar. La intención de este regreso se considera como una nota de advertencia: React.js y Ember ofrecen alternativas sólidas, la ruta de migración de la versión 1 de Angular a la versión 2 está causando incertidumbre, y vemos que algunas organizaciones han adoptado el marco de trabajo sin pensar realmente si una aplicación de una sola página satisface sus necesidades. Hemos tenido apasionados debates internos sobre este tema, pero hemos visto ciertamente que algunas bases de código se han vuelto demasiado complejas dada la combinación de vinculación en doble sentido y patrones incongruentes de gestión de estado. Creemos que en vez de desechar un marco de desarrollo sólido, se pueden resolver estos asuntos por medio de un diseño minucioso y el uso, desde el inicio, de Redux o Flux.

Aurelia se considera el framework de JavaScript de la próxima generación para desarrollo del lado del cliente y se desarrolló usando una versión moderna de JavaScript: ECMAScript 2016 Aurelia fue creado por Rob Eisenberg, el creador de Durandal. Abandonó el equipo principal de Angular 2.0 para dedicar su tiempo a este proyecto. La gran cosa acerca de Aurelia es que es muy modular, tiene librerías pequeñas y sencillas y se creó para ser personalizado fácilmente. Aurelia sigue el patrón de convención sobre configuración, que permite una

LENGUAJES & FRAMEWORKS *continuación*

producción y consumo de módulos más fáciles, pero no hay convenciones firmes que se deban cumplir. Aurelia tiene una gran comunidad, y en el sitio web del proyecto puede aprender más al usar los tutoriales.

La intersección entre dispositivos del IoT (Internet de las cosas) y el ecosistema de JavaScript ofrece posibilidades interesantes. **Cylon.js** es una librería de JavaScript para desarrollar interfaces de robótica y del Internet de las cosas, que ha despertado mucho interés en nuestra comunidad técnica. Ofrece soporte para más de 50 dispositivos de plataformas, y para entrada / salida de propósitos generales con un conjunto compartido de controladores provisto por el módulo cylon-gpio. El control de los dispositivos se efectuará, entonces, por medio de una interfaz de navegador web.

Continuamos percibiendo mucha emoción de los desarrolladores que usan el lenguaje de programación **Elixir**. 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 un pipeline de funciones de forma similar a la shell de comandos 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 permite herramientas como la de versiones Mix, la linea de comandosl interactiva lex y el protocolo de pruebas unitarias ExUnit.

Nos han pedido que volvamos a considerar **Elm** debido a la rápida adopción de un protocolo de Redux. Elm, la inspiración original para Redux, ofrece la división en componentes de vistas y la reactividad de React.js, junto con el estado predecible de Redux, en un lenguaje funcional, compilado con asignación firme de tipos. Elm se desarrolló en Haskell y tiene una sintaxis similar a Haskell, pero se compila a HTML, CSS y JavaScript para el navegador. Los desarrolladores de JavaScript que se apuren para adoptar React.js y Redux podrían también querer considerar Elm como una alternativa segura para tipos, en algunas aplicaciones.

Cuando revisamos las implementaciones de REST en sitios agrestes, con frecuencia vemos que REST se usa de mala forma para ingenuamente recuperar grafos de objetos, por medio de interacciones locuaces entre clientes y servidores. **GraphQL** de Facebook es una alternativa interesante a REST, que podría ser un enfoque mejor para este caso de uso muy común. Como protocolo para la recuperación remota de grafos de objetos, GraphQL ha recibido una gran atención recientemente. Una de las características más interesantes de GraphQL es su naturaleza orientada a consumidores: La estructura de una respuesta está impulsada, por completo, por el cliente y no por el servidor. Así se desacopla al consumidor y se obliga al servidor a cumplir con la ley de Postel. Las implementaciones de clientes están ahora disponibles en muchos lenguajes de programación, pero hemos visto una avalancha de interés en Relay de Facebook, un protocolo de JavaScript que se diseñó para admitir el modelo de componentes sin estado de React.js.

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 normales de JavaScript, por lo que se deben evitar las referencias a objectos JavaScript de objetos inmutables deberian ser evitadas. Nuestros equipos han notado que es valioso utilizar esta librería para rastrear la mutación y mantener estados. Alentamos a los desarrolladores a que investiguen esta librería, especialmente cuando se combina con el resto del stack de Facebook.

Hemos disfrutado cómo **Recharts** integra a tablas D3 en React.js de una forma limpia y declarativa.

Muchos desarrolladores de iOS están usando **JSPatch** para parchar dinámicamente sus aplicaciones. Cuando una aplicación habilitada para JSPatch se ejecuta, esta carga un pedazo de JavaScript (posiblemente a través de una conexión HTTP no segura) y luego lo vincula al código principal de la aplicación en Objective-C, con el fin de cambiar el comportamiento, resolver problemas, etc. Si bien es conveniente, pensamos que parchar en caliente (monkey patch) las aplicaciones en ejecución es una mala idea y debe evitarse. Cuando se efectúa parches incrementales, es muy importante que su proceso de pruebas corresponda a los que los usuarios finales van a experimentar para validar apropiadamente la funcionalidad. Otra alternativa es usar React Native para la aplicación junto con AppHub y CodePush para aplicar pequeñas actualizaciones y características nuevas.

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 3500 empleados repartidos a lo largo de sus 35 oficinas en 12 países: Australia, Brasil, Canadá, China, Ecuador, Alemania, India, Singapur, Sudáfrica, Turquía, Reino Unido y los Estados Unidos.

ThoughtWorks®