ThoughtWorks®

TECHNOLOGY RADAR *VOL.17*

Nuestra visión sobre la tecnología y tendencias que dan forma al futuro

thoughtworks.com/es/radar

#TWTechRadar

CONTRIBUYENTES

El Technology Radar está preparado por el Comité Consultor de Tecnología de ThoughtWorks, compuesto por:

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

¿QUÉ HAY DE NUEVO?

Temas destacados en esta edición:

CÓDIGO ABIERTO EN EL CRECIMIENTO DE CHINA

La marea está subiendo. Debido a los cambios en actitud y políticas, grandes empresas chinas como Alibaba y Baidu están lanzando rápidamente frameworks, herramientas y plataformas de código abierto. El crecimiento de sus ecosistemas de software está acelerándose rápidamente mientras se expanden de manera económica.

Debido a la cantidad de proyectos de software en los mercados en auge y masivos, se espera que aumenten el número y la calidad de proyectos de código abierto que se incluyen en GitHub y otros sitios de código abierto. ¿Por qué a las empresas chinas les interesaría tener tantos activos en código abierto? Al igual que sucede con muchos potenciales mercados de software en Silicon Valley, la competencia por desarrolladores es fuerte y ofrecer una compensación mayor solo sirve un poco.

La posibilidad de trabajar en proyectos de vanguardia de código abierto con otros desarrolladores inteligentes es un incentivo universal. Esperamos que las innovaciones principales de código abierto continúen la tendencia de que los archivos de README (léame) se escriban primero en chino y luego en inglés.

KUBERNETES, LA ELECCIÓN PARA EL ORQUESTADOR DE CONTENEDORES

Una gran cantidad de artículos del Radar se enfocaron en Kubernetes y su presencia cada vez más dominante en muchos proyectos. Parece que el ecosistema de desarrollo de software se está estableciendo en Kubernetes y las herramientas relacionadas para resolver los problemas comunes de implementación, expansión y operación de contenedores.

Los artículos del Radar tales como GKE, Kops, y Sonobuoy presentan servicios de plataforma gestionada y herramientas para mejorar la experiencia general de adoptar y ejecutar Kubernetes. En realidad, su capacidad de operar fácilmente varios contenedores como una unidad de programación habilita Service Mesh y sidecar para seguridad de los endpoints.

Kubernetes se ha convertido en el sistema operativo predeterminado para contenedores. Muchos proveedores de nube han aprovechado su arquitectura abierta y modular, para adoptar y ejecutar Kubernetes. A la vez, las herramientas aprovechan sus API para acceder a abstracciones tales como cargas de trabajo, grupos, configuración y almacenamiento.

Vemos que más productos utilizan Kubernetes como un ecosistema, por lo que se considera el siguiente nivel de abstracción después de los microservicios y contenedores. Esta es evidencia adicional de que los desarrolladores pueden aprovechar con éxito los estilos modernos de arquitectura a pesar de las complejidades inherentes de los sistemas distribuidos.

LA NUBE ES LO NORMAL AHORA

Otro tema dominante de conversación, al reunir los artículos de esta edición, tenía una naturaleza "nublada". Ahora que los proveedores de nube tienen más capacidad y están igualando la cantidad de opciones, el modelo de nube pública se ha convertido en el escogido por defecto en muchas organizaciones.

En vez de preguntar "¿por qué en la nube?", muchas empresas plantean ahora "¿por qué no usamos la nube?", cuando se inician proyectos nuevos. Sin duda, algunos tipos de software todavía requieren sistemas en las instalaciones, pero cuando los precios disminuyen y las capacidades se expanden, el desarrollo nativo en la nube se vuelve cada vez más factible.

A pesar de que las opciones básicas serán similares para los principales proveedores de la nube, también ofrecen opciones únicas con el fin de diferenciarse según tipos específicos de soluciones. Por tanto, vemos que las empresas aprovechan los diferentes proveedores mediante Polycloud, y escogen las capacidades especializadas de esas plataformas que más encajan con las necesidades de sus clientes.

LA CONFIANZA EN BLOCKCHAIN SE DISTRIBUYE MÁS UNIFORMEMENTE

A pesar del caos de las criptomonedas en los mercados, muchos de nuestros clientes están detectando maneras de aprovechar las soluciones de blockchain para registros distribuidos y contratos inteligentes. Varios artículos del Radar muestran la madurez de la utilización de tecnologías relacionadas con cadenas de bloque, al presentar formas cada vez más interesantes de implementación de contratos inteligentes, con una variedad de técnicas y lenguajes de programación.

Blockchain resuelve el problema ancestral de registros distribuidos, fiables, compartidos e indelebles. Hoy en día, las empresas están aumentando la confianza que sienten sus usuarios en los mecanismos subyacentes de las implementaciones de cadena de bloque. Muchos sectores tienen problemas específicos distribuidos de fiabilidad; esperamos que las soluciones de blockchain continúen detectando métodos para resolverlos.

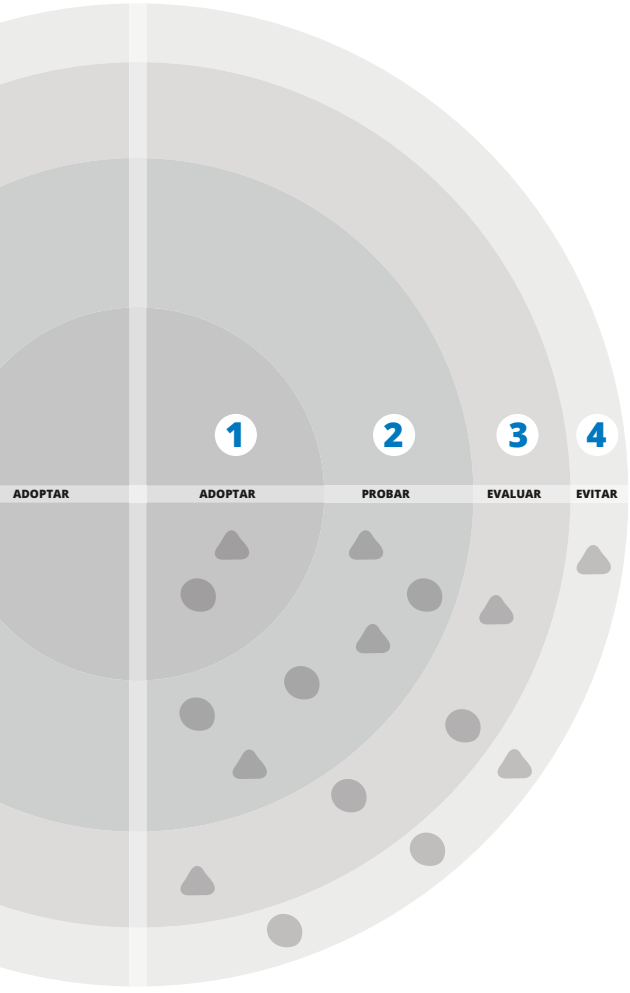
SOBRE EL RADAR

Las y los ThoughtWorkers somos apasionados por la tecnología. La construimos, la investigamos, la probamos, la publicamos en código libre, escribimos sobre ella y nos esforzamos por mejorarla constantemente para todos. Nuestra misión es liderar la excelencia en software y revolucionar la industria de las TI. Creamos y compartimos el Technology Radar de ThoughtWorks en honor a esta misión. El Comité de Tecnología de ThoughtWorks, un grupo de líderes senior en tecnología, crea el Radar. Ellos se reúnen periodicamente para discutir la estrategia global de tecnología y las tendencias tecnológicas que significativamente impactan a nuestra industria.

El Radar recoge los resultados de las discusiones del Comité en un formato que proporciona valor a un amplio rango de personas interesadas, desde desarrolladores a CTOs. El contenido está direccionado a ser un resumen conciso.

Te alentamos a explorar estas tecnologías con más detalle. El Radar es gráfico por naturaleza, agrupando los elementos en técnicas, herramientas plataformas, lenguajes y frameworks. Cuando los elementos del Radar encajan en múltiples cuadrantes, escogemos el que nos parezca más apropiado. Posteriormente agrupamos estos elementos en cuatro anillos que reflejan nuestra posición actual acerca de ellos.

Para mayor información sobre el Radar, visita thoughtworks.com/radar/faq



EL RADAR EN UN VISTAZO

1 ADOPTAR

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

2 PROBAR

Vale la pena probarlos. Es importante entender cómo desarrollar estas capacidades. Las empresas deberían probar esta tecnología en proyectos en que se puede manejar el riesgo.

3 EVALUAR

Vale la pena explorar, con la comprensión de cómo pordía afectar a su empresa.

4 EVITAR

Proceder con cautela.

▲ NUEVO O CAMBIADO

● SIN CAMBIOS

Los ítems nuevos o que han sufrido cambios significativos desde el último radar están representados con triángulos, mientras que los ítems que no han cambiado están representados por círculos.

! Nuestro radar mira al futuro. Para hacer espacio para nuevos ítems, atenuamos elementos que no se han movido recientemente, lo que no es un reflejo de su valor sino mñas bien de nuestro limitado espacio en el Radar.

EL RADAR

TÉCNICAS

ADOPTAR
1. Registros Livianos de Decisiones de Arquitectura

PROBAR
2. Aplicación de gestión de productos a plataformas internas *NUEVO*
3. Función de aptitud de arquitectura *NUEVO*
4. Patrón de burbuja autónoma *NUEVO*
5. Chaos Engineering *NUEVO*
6. Desacoplamiento de gestión secreta como código fuente
7. DesignOps *NUEVO*
8. Legado en una caja
9. Micro frontends
10. Pipelines para infraestructura como código *NUEVO*
11. Arquitectura sin servidores
12. TDD para contenedores *NUEVO*

EVALUAR
13. Operaciones algorítmicas de TI *NUEVO*
14. Ethereum para aplicaciones descentralizadas *NUEVO*
15. Transmisión de eventos como fuente de la verdad *NUEVO*
16. Equipos de productos de ingeniaría de plataformas
17. Polycloud *NUEVO*
18. Service Mesh *NUEVO*
19. Sidecars para seguridad de endpoints *NUEVO*
20. Las tres Rs de la seguridad *NUEVO*

EVITAR
21. Una única instancia de CI para todos los equipos
22. Práctica de CI
23. Entornos empresariales para pruebas de integración
24. Recreación de entornos ESB con Kafka *NUEVO*
25. Generación de código basado en especificaciones

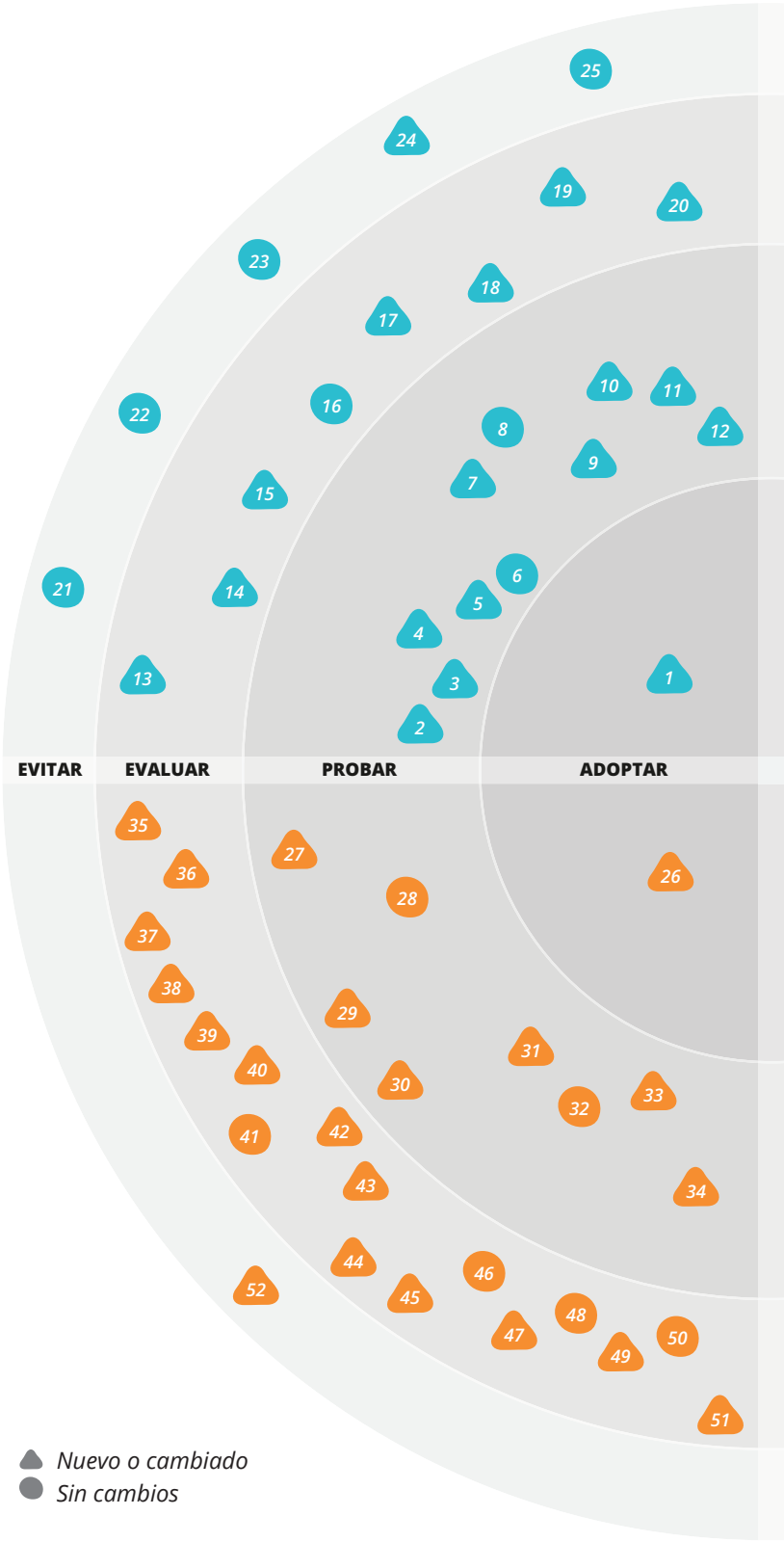
PLATAFORMAS

ADOPTAR
26. Kubernetes

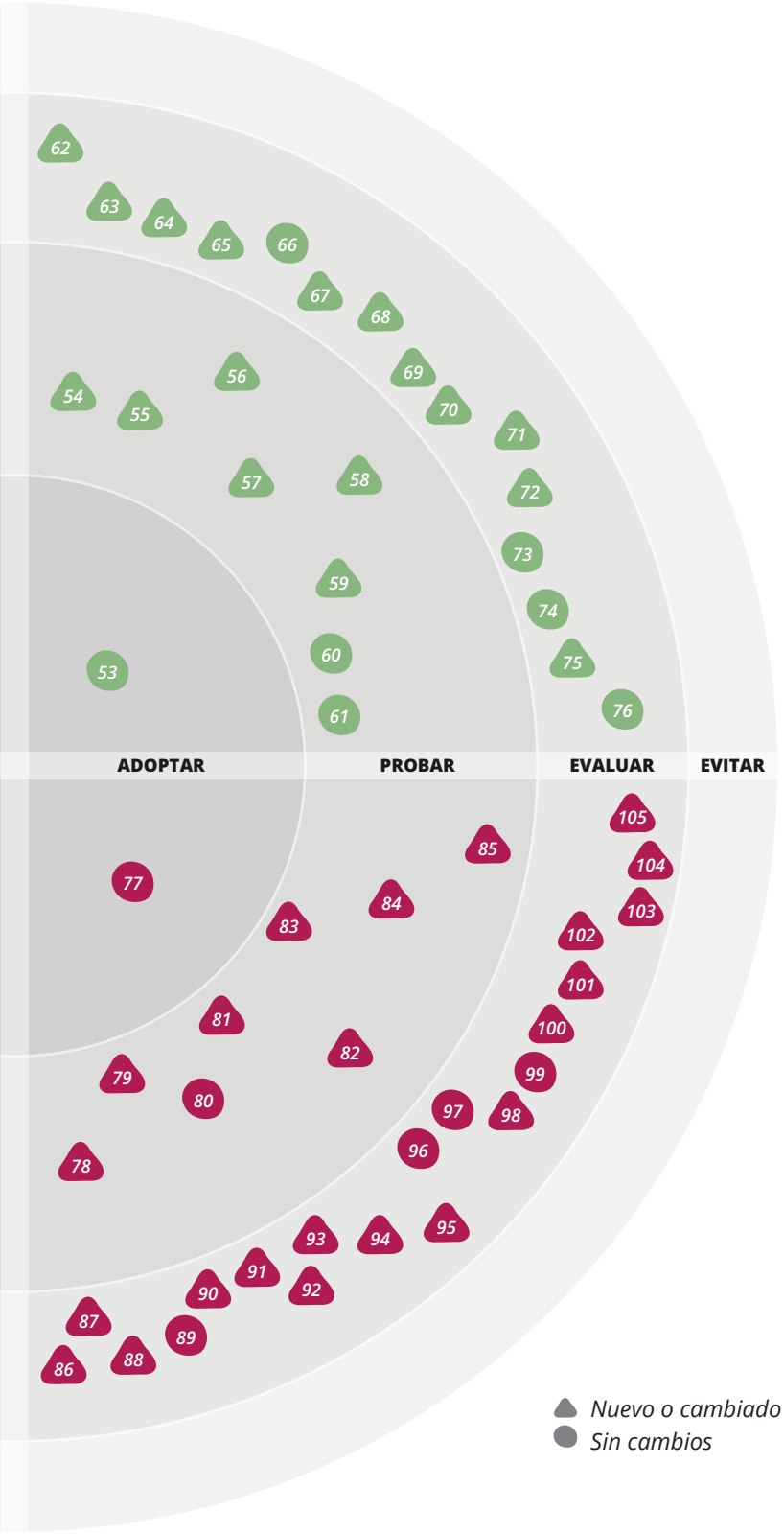
PROBAR
27. .NET Core
28. AWS Device Farm
29. Flood IO *NUEVO*
30. Google Cloud Platform *NUEVO*
31. Keycloak
32. OpenTracing
33. Unity más allá de los videojuegos
34. WeChat *NUEVO*

EVALUAR
35. Azure Service Fabric *NUEVO*
36. Cloud Spanner *NUEVO*
37. Corda *NUEVO*
38. Cosmos DB *NUEVO*
39. DialogFlow
40. GKE *NUEVO*
41. Hyperledger
42. Kafka Streams
43. Language Server Protocol *NUEVO*
44. LoRaWAN *NUEVO*
45. MapD *NUEVO*
46. Mosquitto
47. Netlify *NUEVO*
48. PlatformIO
49. TensorFlow Serving *NUEVO*
50. Plataformas de voz
51. Contenedores de Windows *NUEVO*

EVITAR
52. Gateways API demasiado ambiciosas



EL RADAR



HERRAMIENTAS

ADOPTAR
53. fastlane

PROBAR
54. Buildkite *NUEVO*
55. CircleCI *NUEVO*
56. gopass *NUEVO*
57. Headless Chrome para pruebas front-end *NUEVO*
58. jsoniter *NUEVO*
59. Prometheus
60. Scikit-learn
61. Framework sin servidores

EVALUAR
62. Apex *NUEVO*
63. assertj-swagger *NUEVO*
64. Cypress *NUEVO*
65. Flow *NUEVO*
66. InSpec
67. Jupyter *NUEVO*
68. Kong API Gateway *NUEVO*
69. kops *NUEVO*
70. Lighthouse *NUEVO*
71. Rendertron *NUEVO*
72. Sonobuoy *NUEVO*
73. spaCy
74. Spinnaker
75. Spring Cloud Contract *NUEVO*
76. Yarn

EVITAR

LENGUAJES Y FRAMEWORKS

ADOPTAR
77. Python 3

PROBAR
78. Angular
79. AssertJ *NUEVO*
80. Avro
81. CSS Grid Layout *NUEVO*
82. Módulos CSS *NUEVO*
83. Jest *NUEVO*
84. Kotlin
85. Spring Cloud

EVALUAR
86. Componentes de Arquitectura de Android *NUEVO*
87. ARKit/ARCore *NUEVO*
88. Atlas y BeeHive *NUEVO*
89. Caffe
90. Clara rules *NUEVO*
91. CSS-en-JS *NUEVO*
92. Digdag *NUEVO*
93. Druid *NUEVO*
94. ECharts *NUEVO*
95. Gobot *NUEVO*
96. Instana
97. Keras
98. LeakCanary *NUEVO*
99. PostCSS
100. PyTorch *NUEVO*
101. single-spa *NUEVO*
102. Solidity *NUEVO*
103. TensorFlow Mobile *NUEVO*
104. Truffle *NUEVO*
105. Weex *NUEVO*

EVITAR

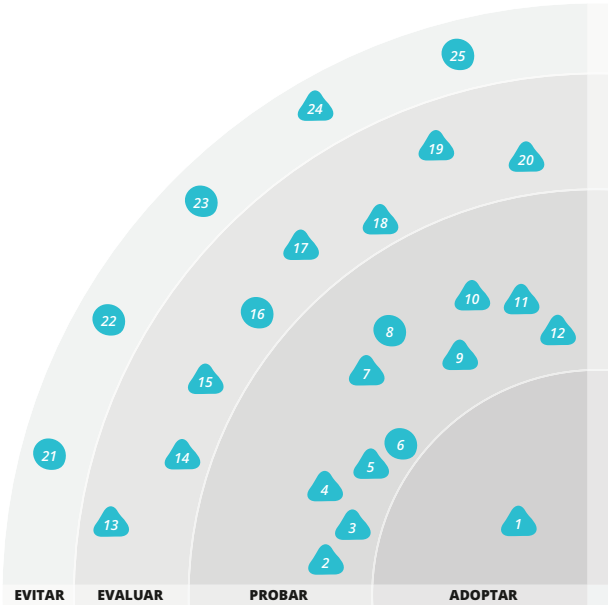
TÉCNICAS

ADOPTAR
1. Registros Livianos de Decisiones de Arquitectura

PROBAR
2. Aplicación de gestión de productos a plataformas internas *NUEVO*
3. Función de aptitud de arquitectura *NUEVO*
4. Patrón de burbuja autónoma *NUEVO*
5. Chaos Engineering *NUEVO*
6. Desacoplamiento de gestión secreta como código fuente
7. DesignOps *NUEVO*
8. Legado en una caja
9. Micro frontends
10. Pipelines para infraestructura como código *NUEVO*
11. Arquitectura sin servidores
12. TDD para contenedores *NUEVO*

EVALUAR
13. Operaciones algorítmicas de TI *NUEVO*
14. Ethereum para aplicaciones descentralizadas *NUEVO*
15. Transmisión de eventos como fuente de la verdad *NUEVO*
16. Equipos de productos de ingeniaría de plataformas
17. Polycloud *NUEVO*
18. Service Mesh *NUEVO*
19. Sidecars para seguridad de endpoints *NUEVO*
20. Las tres Rs de la seguridad *NUEVO*

EVITAR
21. Una única instancia de CI para todos los equipos
22. Práctica de CI
23. Entornos empresariales para pruebas de integración
24. Recreación de entornos ESB con Kafka *NUEVO*
25. Generación de código basado en especificaciones



Gran parte de la documentación puede reemplazarse con código y pruebas legibles. En un mundo de arquitecturas evolutivas, sin embargo, es importante registrar ciertas decisiones de diseño para el beneficio de personas nuevas futuras del equipo y para inspecciones externas. Los **REGISTROS LIVIANOS DE DECISIONES DE ARQUITECTURA** son una técnica para capturar decisiones importantes de arquitectura junto con su contexto y consecuencias. Recomendamos almacenar estos detalles en control de código fuente, en vez de un wiki o sitio web, ya que pueden proveer un registro que continúe sincronizado con el código en sí. Para la mayoría de proyectos, no vemos un motivo por el cual no se utilizaría esta técnica.

Hemos notado un gran aumento de interés en el tema de plataformas digitales durante los 12 últimos meses. Las empresas que buscan implementar soluciones digitales, de forma rápida y eficiente, están construyendo plataformas internas, que ofrecen a los equipos un acceso de autoservicio a los APIs de negocios, herramientas, conocimiento y apoyo necesario para crear y operar sus propias soluciones. Pensamos que estas plataformas son las más eficaces cuando se las respeta de la misma manera como si se tratara de una oferta externa de productos. La **APLICACIÓN DE LA GESTIÓN DE PRODUCTOS A LAS PLATAFORMAS INTERNAS** significa establecer empatía con consumidores internos (es decir, desarrolladoras y desarrolladores) y colaborar con estas personas en el diseño. Los gerentes de productos de plataformas establecen mapas de ruta y garantizan que estas entreguen valor al negocio y mejoren la experiencia de la persona desarrolladora. Algunos dueños incluso crean una identidad de marca para la plataforma interna y la utilizan para mercadear los beneficios a sus colegas. Los gerentes de productos de plataformas se preocupan de la calidad de la plataforma, reúnen métricas de uso y las mejoran continuamente con el tiempo. El tratar a la plataforma como un producto ayudará a crear un ecosistema próspero y evitará el problema de construir otra arquitectura orientada a servicios estancada y subutilizada.

Un término proveniente de la computación evolutiva es la función de aptitud, que se utiliza para resumir la probabilidad de que una solución de diseño específica logre las metas indicadas. Cuando se define un algoritmo evolutivo, el diseñador busca el mejor algoritmo. La función de aptitud define el significado del término "mejor" en este contexto. Una **FUNCIÓN DE APTITUD DE ARQUITECTURA**, según se define en Building Evolutionary Architectures, presenta

una evaluación objetiva de integridad de algunas características de la arquitectura, que puede abarcar criterios de verificación existentes, tales como pruebas de unidad, parámetros, monitoreo, etc. Pensamos que los arquitectos pueden comunicar, validar y conservar las características de arquitectura de una forma automatizada y continua, que es la clave para construir arquitecturas evolutivas.

Un término proveniente de la computación evolutiva es la función de aptitud, que se utiliza para resumir la probabilidad de que una solución de diseño específica logre las metas indicadas.

(Función de aptitud de arquitectura)

Muchas organizaciones con las que trabajamos están intentando fuertemente utilizar enfoques de ingeniería modernos para crear nuevas capacidades y funcionalidades, y al mismo tiempo teniendo que coexistir con una larga cola de sistemas legados. Una estrategia antigua que, basada en nuestra experiencia, se ha vuelto cada vez más útil en estos escenarios es el **PATRÓN DE BURBUJA AUTÓNOMA** de Eric Evans. Este enfoque incluye crear un contexto fresco para el desarrollo de aplicaciones nuevas que se protege de los enredos del mundo de sistemas legados. Este es un paso más allá de solo usar una capa anticorrupción. Brinda al nuevo contexto de burbuja un control completo sobre sus datos de respaldo, los cuales se mantienen actualizados de forma no sincronizada con los sistemas legados. Se requiere algo de trabajo para proteger los límites de la burbuja y mantener consistentes a ambos mundos, pero la autonomía resultante y la disminución en la fricción del desarrollo es un primer paso valiente hacia una arquitectura futura y modernizada.

En ediciones previas de Radar, hablamos acerca de usar Chaos Monkey de Netflix para pruebas de cómo un sistema activo puede afrontar las interrupciones en producción, al desactivar al azar las instancias y medir los resultados. **CHAOS ENGINEERING** es el término innovador para la aplicación más amplia de esta técnica. Al ejecutar experimentos en sistemas distribuidos en producción, somos capaces de generar suficiente confianza de que tales sistemas funcionan según lo esperado en condiciones turbulentas. Un buen lugar para comenzar a comprender esta técnica es el sitio web Principles of Chaos Engineering.

Inspirado por el movimiento DevOps, **DESIGNOPS** es un cambio cultural y un conjunto de prácticas que permite a las personas de una organización, continuar rediseñando productos sin afectar la calidad, la coherencia del servicio ni la autonomía del equipo. DesignOps promueve la creación y evolución de una infraestructura de diseño que minimiza el esfuerzo necesario para crear conceptos y variaciones nuevos de interfaz de usuario, y para establecer un bucle de retroalimentación rápido y confiable con los usuarios finales. Con herramientas como Storybook, que promueven una colaboración estrecha, la necesidad de un análisis abierto y transferencias de especificaciones se reduce al mínimo absoluto. Con DesignOps, el diseño ya no es una práctica específica sino que llega a formar parte del trabajo de todos.

Hemos notado beneficios significativos de la incorporación de arquitecturas de microservicios, que han permitido que los equipos amplíen el alcance de sus servicios desplegados y mantenidos de forma independiente. Lamentablemente, también notamos que muchos equipos crean monolitos *front-end*, una sola aplicación de navegador grande y extensa, que se instala encima de los servicios de *back-end*. Nuestro enfoque preferido (y demostrado) es dividir el código basado en navegador en varios **MICRO FRONTENDS**. En este enfoque, la aplicación web se divide según sus funcionalidades y cada equipo distinto se adueña de una funcionalidad, de *frontend* a *backend*. Esto garantiza que se desarrollará, probará e implantará cada funcionalidad de forma independiente con respecto a las demás. Hay varias técnicas para recombinar las funcionalidades, a veces como páginas y en otras ocasiones como componentes, con el fin de crear una experiencia cohesiva para el usuario.

El uso de pipelines de entrega continua para orquestar el proceso de despliegue de software se ha convertido en un concepto convencional. Sin embargo, cambios en las pruebas automatizadas de la infraestructura como código no es un concepto ampliamente entendido. Las herramientas de integración continua (CI) y entrega continua (CD) pueden ser usadas para probar la configuración del servidor (ej. recetas de Chef, módulos de Puppet, playbooks de Ansible), creación de la imagen del servidor (ej. Packer), provisionamiento del ambiente (ej. Terraform, CloudFormation) e integración de ambientes. El uso de **PIPELINES PARA INFRAESTRUCTURA COMO CÓDIGO** permite encontrar errores antes de que los cambios sean aplicados a ambientes operacionales, incluyendo

ambientes usados para desarrollo y pruebas. También ofrecen métodos para asegurar que la infraestructura se esté ejecutando constantemente desde los agentes de CI/CD, en lugar de ejecutarse como estaciones de trabajo individuales. Aún quedan algunos retos pendientes como los largos bucles de retroalimentación asociados al levantamiento de contenedores y máquinas virtuales. A pesar de esto, hemos encontrado que sigue siendo una técnica valiosa.

Las herramientas de CI y CD pueden ser usadas para probar configuración del servidor, creación de imagen del servidor, provisionamiento e integración de ambientes.

(Pipelines para infraestructura como código)

La utilización de una **ARQUITECTURA SIN SERVIDORES** se ha convertido muy rápidamente en un enfoque aceptado para las organizaciones que implantan aplicaciones de nube, con una gran gama de opciones disponibles para la implementación. Incluso las organizaciones conservadoras tradicionalmente están utilizando parcialmente algunas tecnologías sin servidores. Gran parte de las conversaciones se centran en funciones como un servicio (por ejemplo, AWS Lambda, Google Cloud Functions y Azure Functions) mientras los modelos adecuados de uso todavía están surgiendo. La implantación de funciones sin servidores, de forma indiscutible, elimina el esfuerzo no insignificante que tradicionalmente se requiere para la configuración y orquestación de sistemas operativos y servidores. Sin embargo, las funciones sin servidores no satisfacen todos los requisitos. En esta etapa, debe estar preparado para recurrir a contenedores de implantación o incluso instancias de servidores para requisitos específicos. Mientras tanto, los demás componentes de una arquitectura sin servidores, tales como Backend como un servicio, se han vuelto una opción predeterminada.

Muchos equipos han adoptado prácticas de TDD para preparar código de aplicaciones, debido a los beneficios brindados por éstas. Otros han recurrido a contenedores para empaquetar y desplegar su software, y existe la práctica aceptada de utilizar *scripts* automatizados para crear los contenedores. Lo que hemos visto que pocos equipos hacen, hasta ahora, es combinar las dos tendencias y guiar el desarrollo de *scripts* de contenedores basándose en pruebas. Con frameworks como Serverspec y Goss, se puede expresar la funcionalidad requerida sea para contenedores aislados u orquestrados, con bucles cortos de realimentación. De esta forma, es posible usar los mismos principios que hemos defendido para el código con **TDD PARA CONTENEDORES**. Nuestra experiencia inicial siguiendo esta estrategia ha sido muy positiva.

La cantidad de datos almacenados por operaciones de TI ha aumentado con el tiempo. Por ejemplo, la tendencia hacia los microservicios significa que más aplicaciones están generando sus propios datos operativos y las herramientas como Splunk, Prometheus, o ELK stack facilitan el almacenamiento y procesamiento de datos, para obtener conocimientos operacionales. Cuando se combinan con herramientas de aprendizaje de máquina cada vez más democratizadas, es inevitable que los operadores comiencen a incorporar en sus juegos de herramientas modelos estadísticos y algoritmos de clasificaciones entrenados. A pesar de que estos algoritmos han estado disponibles por años, y se han hecho varios intentos para automatizar la gestión de servicios, solo ahora comenzamos a comprender cómo las máquinas y los seres humanos pueden colaborar para identificar interrupciones de forma más temprana o encontrar sus causas. A pesar de que hay el riesgo de sobrevalorar las **OPERACIONES ALGORÍTMICAS DE TI**, una mejora constante de los algoritmos de aprendizaje de máquina, de forma inevitable, cambiarán el rol de los seres humanos en los centros de datos del futuro.

Blockchain ha sido ampliamente catalogada como la panacea para todo: desde la Banca a la moneda digital, para transparentar la cadena de suministro.

(Ethereum para aplicaciones descentralizadas)

Blockchain ha sido ampliamente catalogada como la panacea para todo lo que combine las finanzas y la tecnología (*fintech*), desde la banca hasta la moneda digital, para transparentar la cadena de suministro. Previamente, hemos presentado Ethereum debido a su conjunto de características que incluyen los contratos inteligentes. Ahora, estamos viendo más desarrollo utilizando **ETHEREUM PARA APLICACIONES DESCENTRALIZADAS** en otras áreas. Aunque todavía es una tecnología poco explorada, animamos a ver su

uso en la construcción de aplicaciones descentralizadas más allá de la criptomoneda y la banca.

Un service mesh ofrece descubrimientos, seguridad, rastreabilidad, monitoreo y manejo de fallas uniformes, sin que se requieran un activo compartido tal como un gateway API o ESB.

(Service mesh)

Ahora que las plataformas de transmisión de eventos, tales como Apache Kafka, han aumentado en popularidad, muchos las consideran una forma avanzada de colas de mensajes, utilizadas únicamente para transmitir eventos. Incluso cuando se usan de esta manera, la transmisión de eventos tiene sus beneficios si se compara con las colas tradicionales de mensajes. Sin embargo, estamos más interesados en cómo las personas usan la **TRANSMISIÓN DE EVENTOS COMO LA FUENTE DE LA VERDAD** con plataformas (Kafka específicamente) como el lugar de almacenamiento primario para datos como eventos inmutables. Un servicio con un diseño de Fuente de Eventos, por ejemplo, puede usar Kafka como su almacenamiento de eventos, que estarán disponibles para el consumo por otros servicios. Esta técnica tiene el potencial de disminuir los esfuerzos de duplicación entre la persistencia local y la integración.

Los proveedores principales de nube (Amazon, Microsoft y Google) están enfrascados en una carrera enérgica para mantener la paridad en capacidades fundamentales, aunque sus productos se diferencian solo ligeramente. Esto está haciendo que unas pocas organizaciones adopten la estrategia de **POLYCLOUD**, en vez de apostar todo en un solo proveedor. Están pasando distintos tipos de cargas de trabajo a varios proveedores en un enfoque de escoger el mejor. Esto puede incluir, por ejemplo, usar servicios estándar en AWS, pero emplear Google para aprendizaje de máquina, Azure para aplicaciones .NET que utilizan SQLServer, o potencialmente usar la solución de blockchain de Ethereum Consortium. Esto es distinto a una estrategia agnóstica de nube que se enfoca en la capacidad de transferencia entre proveedores, que es costosa y les fuerza a pensar en el menor denominador común. Polylcloud se centra más bien en utilizar lo mejor que ofrece la nube.

Ahora que las organizaciones grandes inician la transición a equipos más autónomos al adueñarse y operar sus propios microservicios, ¿cómo pueden garantizar la coherencia y compatibilidad necesarias entre estos servicios sin depender en una infraestructura de alojamiento centralizado? Para trabajar juntos de forma eficiente, incluso los micro-servicios autónomos necesitan alinearse con algún estándar organizacional. Un **SERVICE MESH** ofrece descubrimientos, seguridad, rastreabilidad, monitoreo y manejo de fallas uniformes, sin que se requieran un activo compartido tal como un gateway API o ESB. Una implementación típica involucra procesos de proxy inverso liviano implementados junto a cada proceso de servicio, tal vez en un contenedor separado. Estos proxis se comunican con registros de servicio, proveedores de identificación, agregadores de bitácora, etc. La interoperabilidad de servicios y la capacidad de observación se logran por medio de una implementación compartida de este proxy pero no con una instancia de tiempo de ejecución compartida. Hemos promovido un enfoque descentralizado a la gestión de microservicios por algún tiempo y nos agrada ver cómo surge este patrón coherente. Los proyectos de código abierto tales como linkerd e Istio continuarán madurando y facilitarán todavía más la implementación de service meshes.

La arquitectura de microservicios, con una gran cantidad de servicios que exponen sus activos y capacidades por medio de APIs y una superficie de ataque mayor, exige una arquitectura de seguridad con cero confiabilidad. “No confíe nunca, siempre verifique.” Sin embargo, con frecuencia, se descuida la tarea de hacer cumplir los controles de seguridad para la comunicación entre servicios, debido a una complejidad mayor del código de servicios y una falta de librerías y apoyo de lenguaje en un ambiente políglota. Para resolver esta complejidad, algunos equipos delegan la seguridad a un sidecar lejos del proceso primario. Este puede ser un proceso o un contenedor que se implanta y programa con cada servicio, que comparte el mismo contexto, parte receptora e identidad de ejecución. Los sidecars implementan capacidades de seguridad, tales como un cifrado transparente de comunicación y la terminación de TLS (seguridad de la capa de transporte), además de la autenticación y autorización del servicio de llamado o el usuario final. Le recomendamos que revise el posible uso de Istio, linkerd o Envoy antes de implementar sus propios **SIDECARS PARA SEGURIDAD DE ENDPOINTS**.

Los enfoques tradicionales a la seguridad empresarial, con frecuencia, enfatizan bloquear los componentes y hacer que el ritmo de cambios sea más lento. Sin embargo, sabemos que mientras más tiempo tenga un atacante para poner en peligro un sistema, mayor será el daño potencial. Las tres Rs de la seguridad empresarial —rotar, reparar y repavimentar— aprovechan la automatización de la infraestructura y la entrega continua para eliminar las oportunidades de ataque. Rotar las credenciales, aplicar parches tan pronto estén disponibles y reconstruir sistemas de un estado conocido y seguro —todo en cuestión de minutos u horas— hacen que sea más difícil que los atacantes sean exitosos. Es posible aplicar la técnica de **LAS TRES RS DE SEGURIDAD** gracias a que han surgido arquitecturas nativas de nube modernas. Cuando se implementan las aplicaciones como contenedores, y se construyen y realizan pruebas mediante un proceso completamente automatizado, un parche de seguridad es solo otra pequeña versión que puede enviarse al proceso, mediante un clic. Por supuesto, para aplicar las mejores prácticas de sistemas distribuidos, los desarrolladores deben diseñar sus aplicaciones para que sean resistentes a interrupciones no esperadas de servidores. Esto es similar al impacto de la implementación de Chaos Monkey en su ambiente.

Kafka se está convirtiendo en una solución de mensajes muy popular, y junto a este, Kafka Streams está en la vanguardia del gran interés por arquitecturas de transmisión. Desafortunadamente, cuando comienzan a incorporar Kafka en el núcleo de sus datos y plataformas de las aplicaciones, vemos que algunas organizaciones recurren a la **RECREACIÓN DE ENTORNOS ESB CON KAFKA**. Esto lo hacen mediante la centralización de componentes del ecosistema de Kafka —tales como conectores y procesadores de secuencias— en lugar de permitir que estos componentes compartan el proceso con el producto o equipos de servicios. Esto nos recuerda los graves problemas de antipatrones de ESB, en donde cada vez más lógica, orquestación y transformación penetran en el ESB gestionado centralmente, lo que crea una dependencia importante en un equipo centralizado. Estamos enfatizando esto para evitar implementaciones a futuro de este patrón defectuoso.

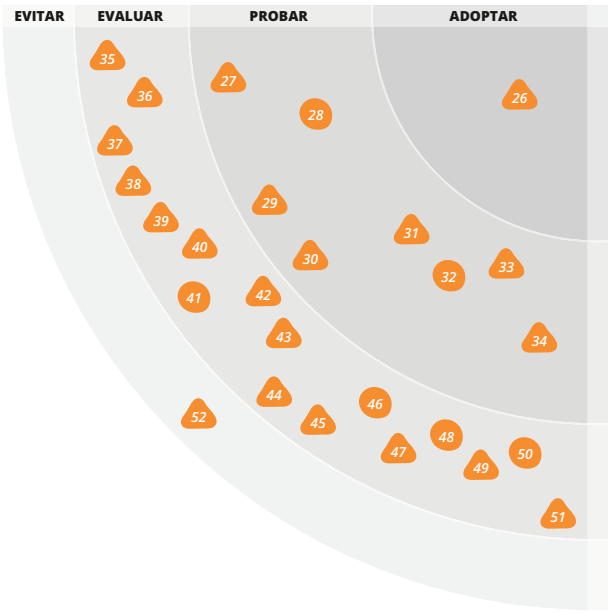
PLATAFORMAS

ADOPTAR
26. Kubernetes

PROBAR
27. .NET Core
28. AWS Device Farm
29. Flood IO *NUEVO*
30. Google Cloud Platform *NUEVO*
31. Keycloak
32. OpenTracing
33. Unity más allá de los videojuegos
34. WeChat *NUEVO*

EVALUAR
35. Azure Service Fabric *NUEVO*
36. Cloud Spanner *NUEVO*
37. Corda *NUEVO*
38. Cosmos DB *NUEVO*
39. DialogFlow
40. GKE *NUEVO*
41. Hyperledger
42. Kafka Streams
43. Language Server Protocol *NUEVO*
44. LoRaWAN *NUEVO*
45. MapD *NUEVO*
46. Mosquitto
47. Netlify *NUEVO*
48. PlatformIO
49. TensorFlow Serving *NUEVO*
50. Plataformas de voz
51. Contenedores de Windows *NUEVO*

EVITAR
52. Gateways API demasiado ambiciosas



Desde la última vez que mencionamos **KUBERNETES** en el Radar, se ha convertido en la solución predeterminada de la mayoría de nuestros clientes al implantar contenedores en un grupo (cluster) de máquinas. Las alternativas no captaron el interés de la mayoría y, en algunos casos, nuestros clientes incluso cambiaron su "motor" a kubernetes. Kubernetes se ha convertido en la plataforma de orquestración de contenedores preferida para plataformas públicas en la nube, incluso Azure Container Service de Microsoft y Google Cloud (ver el blip de GKE). Y hay muchos productos útiles que enriquecen el ecosistema de kubernetes de rápido crecimiento. Las plataformas que intentan esconder kubernetes bajo una capa de abstracción, sin embargo, todavía deben demostrar su validez.

Hemos notado una adopción más frecuente de **.NET CORE**, el marco de software de código abierto entre plataformas. .NET Core permite el desarrollo e despliegue de aplicaciones .NET en Windows, macOS y Linux. Con el lanzamiento de .NET Standard 2.0 que aumenta la cantidad de APIs estándar en plataformas .NET, la ruta de migración a .NET Core se ha vuelto más clara. Temas relacionados con el soporte de librerías de .NET Core se están facilitando y ahora están disponibles las herramientas entre plataformas de primera clase, lo que permite un desarrollo productivo en plataformas que no son Windows. Se presentan imágenes de Blessed Docker para facilitar la integración con servicios .NET Core en un ambiente de contenedores. Las direcciones positivas en la comunidad y la retroalimentación de nuestros proyectos indican que .NET Core está listo para una uso extendido.

Las pruebas de cargas se facilitaron cuando maduraron herramientas tales como Gatling y Locust. A la vez, las infraestructuras elásticas de nube facilitan la simulación de una gran cantidad de instancias de cliente. Nos encanta ver que Flood y otras plataformas de nube dan más pasos más allá al aprovechar estas tecnologías. **FLOOD IO** es un servicio de pruebas de cargas de

SaaS que ayuda a distribuir y ejecutar scripts de pruebas en cientos de servidores en la nube. Nuestro equipo considera que es fácil migrar las pruebas de rendimiento a Flood al volver a utilizar los scripts de Gatling.

Debido a que **GOOGLE CLOUD PLATFORM** (GCP) se ha extendido a nivel de regiones geográficas disponibles y la madurez de los servicios, los clientes globalmente pueden considerarla, de forma seria, para su estrategia de nube. En algunas áreas, GCP ha conseguido tener las mismas características que su competidor, Amazon Web Services, mientras que en otras áreas se ha diferenciado, especialmente en plataformas de aprendizaje de máquina accesibles, herramientas de ingeniería de datos y Kubernetes viable como una solución de servicios (GKE). En la práctica, nuestros equipos solamente elogiaron la experiencia de desarrollo cuando utilizaron las herramientas de GCP y APIs.

Google Cloud Platform (GCP) se ha expandido a nivel de regiones geográficas y madurez de los servicios, y ahora es considerada seriamente en estrategias de nube de forma global.

(Google Cloud Platform)

En un microservicio, o cualquier otra arquitectura distribuida, una de las necesidades más comunes es asegurar los servicios o APIs con opciones de autenticación y autorización. En este punto, es útil **KEYCLOAK**. Keycloak es una solución de manejo de identidad y acceso de código abierto que facilita la seguridad de las aplicaciones o microservicios casi sin ningún código. Admite autenticación única, autenticación con redes sociales y protocolos estándar, tales como OpenID Connect, OAuth 2.0 y SAML sin mayores ajustes. Nuestros equipos han estado usando esta herramienta y planean continuar usándola en el futuro previsible. Pero requiere un poco de trabajo para configurarlo. Debido a que la configuración ocurre en la inicialización y en tiempo de ejecución por medio de APIs, es necesario escribir scripts para garantizar que se puedan repetir los despliegues.

En versiones previas del Radar, mencionamos que Unity se ha convertido en la plataforma preferida para el desarrollo de aplicaciones de VR y AR, porque provee las abstracciones y las herramientas de una plataforma madura y a la vez es más accesible que su alternativa principal, Unreal Engine. Con las inclusiones recientes de ARKit para iOS y ARCore para Android, las dos plataformas móviles principales ahora tienen los SDK nativos robustos para construir aplicaciones de realidad aumentada. Pero, pensamos que muchos equipos, especialmente aquellos sin una gran experiencia en la creación de juegos, se beneficiarán de utilizar una abstracción tal como Unity. Es por esto que se llama **UNITY MÁS ALLÁ DE LOS VIDEOJUEGOS**. Esto permite el enfoque en un SDK para los desarrolladores que no conocen mucho acerca de la tecnología. También ofrece una solución para una gran cantidad de dispositivos, especialmente Android, que no se incluyen en los SDK nativos.

WECHAT, visto con frecuencia como un equivalente a WhatsApp, se está convirtiendo en la plataforma de negocios de facto en China. Muchas personas no lo saben pero WeChat también es una de las plataformas de pago en línea más populares. Con el CMS incorporado en la aplicación y la gestión de miembros, los negocios pequeños están ahora realizando sus transacciones por completo en WeChat. Por medio de la característica 'Cuenta de Servicios', *Service Account*, , las organizaciones grandes pueden crear una interfaz entre su sistema interno y sus empleados. Debido a que más del 70% de los chinos están usando WeChat, hay que considerar la importancia de esta plataforma cuando los negocios quieren expandirse al mercado de ese país.

AZURE SERVICE FABRIC es una plataforma de sistemas distribuidos construida para microservicios y contenedores. Es comparable a orquestadores de contenedores, tales como Kubernetes, pero también funciona con anteriores servicios simples. Se puede utilizar en una gama de variedades, desde servicios sencillos en su lenguaje escogido hasta contenedores o servicios de Docker que se crearon usando un SDK. Desde su lanzamiento hace un par de años, ha agregado continuamente más funcionalidades, incluyendo un soporte de contenedores de Linux. Kubernetes es un modelo en cuanto a herramientas de orquestación de contenedores, pero Service Fabric es la opción seleccionada para aplicaciones de .NET. Lo estamos usando en algunos proyectos en ThoughtWorks y nos gusta lo que hemos visto hasta ahora.

CLOUD SPANNER es un servicio de base de datos relacional completamente administrado, que ofrece un alto grado de disponibilidad y consistencia sin afectar la latencia. Google ha estado trabajando en una base de datos distribuida globalmente llamada Spanner, por algún tiempo. Se ha lanzado el servicio al mundo exterior con el nombre de Cloud Spanner. Puede expandir la instancia de la base de datos desde un nodo hasta miles de nodos en todo el globo, sin preocuparse por la consistencia de los datos. Al aprovechar TrueTime, un reloj distribuido y altamente disponible, Cloud Spanner brinda una fuerte consistencia para lecturas y snapshots. Puede utilizar SQL estándar para leer datos de Cloud Spanner, pero para operaciones de escritura debe utilizar su API de RPC. A pesar de que no todos los servicios requieren una base de datos distribuida de escala global, la disponibilidad general de Cloud Spanner es un gran cambio en la manera en que pensamos sobre bases de datos. Y su diseño está influyendo en productos de código abierto tales como CockroachDB.

Después de una exploración exhaustiva, R3, un participante importante en el espacio de *blockchain*, se dio cuenta que esta no cumplía bien con su propósito, por lo que crearon **CORDA**. Corda es una plataforma de tecnología de libro de cuentas distribuida (*distributed ledger technology,* DLT) , enfocada en el campo financiero. R3 tiene una propuesta muy clara de valor y saben que su problema requiere de un enfoque pragmático sobre la tecnología. Esto coincide con nuestra experiencia. Las soluciones actuales de *blockchain* pueden no ser opciones razonables para ciertos casos de negocio, debido al costo de minado de datos y la falta de eficiencia operativa. A pesar de que la experiencia de desarrollo que hemos tenido con Corda no ha estado libre de tropiezos, ya que Los APIs están todavía inestables luego de la versión 1.0, y esperamos ver cómo el espacio de DLT madurará aún más.

COSMOS DB es el servicio de base de datos de multimodelo y distribuido globalmente, que pertenece a Microsoft y estuvo disponible en general a inicios de este año. Si bien la mayoría de bases de datos sin SQL ofrecen consistencia adaptable, Cosmos DB lo convierte en un ciudadano de primera clase y ofrece cinco modelos de consistencia distintos. Vale la pena destacar que también admite varios modelos, tales como valores clave, documentos, familia de columnas y gráficos, y todos estos se mapean a su modelo interno de datos, llamado secuencia de registro de átomo (ARS, por sus siglas en inglés). Uno de los aspectos interesantes de Cosmos DB es que ofrece acuerdos de nivel de servicios (SLA, por sus siglas en inglés) sobre su latencia, productividad, consistencia y disponibilidad. Con su amplia gama de aplicaciones, ha fijado un estándar estricto que otros proveedores de nube deben igualar.

En paralelo con el reciente surgimiento de chatbots y plataformas de voz, hemos visto una propagación de herramientas y plataformas que proveen un servicio para extraer la intención del texto y administrar los flujos de conversación a los que podemos acceder. **DIALOGFLOW** (antes API.ai), que Google compró, es un tipo de oferta de "comprensión de lenguaje natural como servicio" que compite con wit.ai y Amazon Lex, entre otros participantes de esta área.

Si bien el ecosistema de desarrollo de software está convergiendo en Kubernetes como la plataforma de orquestación para contenedores, la ejecución de grupos de Kubernetes sigue siendo compleja por las operaciones. **GKE** (el motor de contenedores de Google) es una solución gestionada de Kubernetes para desplegar aplicaciones en contenedores que alivia la sobrecarga operativa de ejecución y mantenimiento de grupos de Kubernetes. Nuestros equipos han tenido una buena experiencia con GKE, y la plataforma se han encargado del trabajo pesado de aplicar parches de seguridad, monitorear y reparar, por su cuenta, los nodos, y gestionar las redes de grupos múltiples y de varias regiones. Según nuestra experiencia, el enfoque de API primero de Google en la exposición de la capacidad de la plataforma, además de aplicar estándares sectoriales, tales como Oauth para la autorización de servicios, mejora la experiencia del desarrollador. Es importante considerar que GKE está siendo desarrollado rápidamente, y a pesar de los esfuerzos de los desarrolladores en alejar a los consumidores de los cambios subyacentes, no has impactado temporalmente. Estamos esperando una mejora continua sobre la madurez de la infraestructura como código con Terraform de GKE y herramientas similares.

KAFKA STREAMS es una librería ligera para construir aplicaciones que manejan transmisión (o *streams*) de datos. Fue diseñada con el objetivo de simplificar lo suficiente el procesamiento de transmisión de datos para hacer un modelo de programación de aplicaciones más accesible para servicios asincrónicos. Puede ser una buena alternativa en escenarios donde se desee aplicar un modelo de procesamiento de transmisión de datos para solucionar el problema, sin la complejidad

de ejecutar un *cluster* (que normalmente es necesario cuando se usan marcos de trabajo integrales de procesamiento de transmisión de datos). Los nuevos desarrollos incluyen el procesamiento de transmisión de "una sola vez" en un *cluster* de Kafka. Esto se logró al introducir idempotencia en los productores de Kafka y permitir escrituras atómicas en múltiples particiones utilizando la nueva API de Transacciones.

Gran parte de la potencia de los IDE sofisticados proviene de su capacidad de análisis de un programa con el fin de crear un árbol de sintaxis abstracta (AST) para luego utilizarlo en el análisis y manipulación del programa. Estas características de soporte tales como la opción de completar automáticamente, encontrar los componentes de llamadas y la refactoración. Los servidores de lenguajes extraen esta capacidad hacia un proceso que le permite a cualquier editor de textos acceder a un API, para trabajar con el AST. Microsoft ha liderado la creación del protocolo **LANGUAGE SERVER PROTOCOL** (LSP), con base en sus proyectos de servidor OmniSharp y TypeScript Server. Cualquier editor que utilice este protocolo puede trabajar con los lenguajes que tengan un servidor compatible con LSP. Esto significa que podemos seguir usando nuestros editores favoritos sin renunciar a los modos de edición de texto enriquecido en muchos lenguajes. Esta opción alegra mucho a nuestros adictos a Emacs.

LORAWAN es una red de baja potencia y área amplia, creada para consumo de baja potencia y comunicación en distancias largas utilizando baja tasa de bits. Permite la comunicación entre dispositivos y gateways, que después reenvían los datos a aplicaciones o servidores, entre otros. Un uso típico es un conjunto distribuido de sensores, o para los dispositivos de Internet de las cosas (IoT, según las siglas en inglés), para los cuales una larga vida útil de batería y comunicaciones de largo alcance son requeridos. Las direcciones de LoRaWAN resuelven dos de los problemas clave de intentar utilizar un wifi normal para tales aplicaciones, que son el consumo de alcance y de potencia. Hay varias implementaciones, y una notable es The Things Network, que es gratuito y de código abierto.

MAPD es una base de datos analítica de columnas en memoria con soporte para SQL está construída para ejecutarse en la GPU. Discutimos sobre si la carga de trabajo de la base de datos está en realidad regida por I/O o computacionalmente, pero hay instancias donde puede resultar util la combinación del paralelismo de la GPU con el gran ancho de banda de la VRAM. MapD gestiona transparentemente los datos usados más frecuentemente en la VRAM (como las columnas que son parte de condiciones de agrupamiento, filtros, cálculos y uniones) y almacena el resto de datos en la memoria principal. Con esta configuración de gestión de la memoria, MapD logra un rendimiento importante en consultas, sin necesitar de índices. A pesar de que hay otros proveedores de bases de datos para GPU, MapD lidera este segmento al haber liberado recientemente el código fuente del motor central de su base de datos y por medio de la GPU Open Analytics Initiative. Si su carga de trabajo analítica es computacionalmente pesada y está pensando en aprovechar el paralelismo de la GPU y de la memoria principal, recomendamos evaluar MapD.

Los servidores de leguaje extraen funciones como autocompletar, encontrar callers and refactorizar una API que permite que cualquier editor de texto funcione con el árbol de sintaxis abstracta del lenguaje.

(Language Server Protocol)

Nos gustan las herramientas sencillas que resuelven un problema muy bien y **NETLIFY** encaja perfectamente en esta descripción. Con esta herramienta se puede crear sitios web con contenido estático, almacenarlos en GitHub y hacer que el sitio web se ejecute y esté disponible fácil y rápidamente. Hay una interfaz de línea de comandos (CLI) para controlar el proceso, el uso de redes de entrega de contenido (CDN) es soportado, puede funcionar con herramientas como Grunt, y, lo que es más importante, Netlify soporta HTTPS.

Los modelos de aprendizaje de máquina están comenzando a introducirse en las aplicaciones diarias de negocios. Cuando existen suficientes datos de entrenamiento, estos algoritmos pueden abordar problemas que antes requerían modelos estadísticos complejos o heurísticas. Ahora que pasamos de usos experimentales a usos en producción, requerimos una forma confiable para alojar e implantar los modelos que permitan el acceso remoto y expansión según la cantidad de consumidores. **TENSORFLOW SERVING** aborda parte ese problema al exponer una interfaz remota de gRPC a un modelo exportado. De esta manera, se permite la implantación de un modelo entrenado mediante una variedad de maneras. TensorFlow Serving

también acepta un flujo de modelos para incorporar actualizaciones continuas de entrenamiento y sus creadores mantienen un Dockerfile para facilitar el proceso de implementación. Supuestamente, la selección de gRPC guardará coherencia con el modelo de ejecución de TensorFlow. Sin embargo, somos algo cautelosos cuando se trata de protocolos que requieren generación de código y vinculación nativa (binding).

A medida que pasamos del uso experimental a la producción, necesitamos una manera confiable de alojar e implementar modelos de aprendizaje automático a los que se pueda acceder de forma remota y ampliar el número de consumidores.

(TensorFlow Serving)

Microsoft se está actualizando en el mundo de los contenedores con sus **CONTENEDORES DE WINDOWS**. Al momento de escribir, Microsoft provee dos imágenes Windows OS como contenedores de Docker, Windows Server 2016 Server Core y Windows Server 2016 Nano Server. Aunque todavía hay espacio para mejores en los contenedores de Windows, por ejemplo, la disminución el gran tamaño de las imágenes y la sofisticación del soporte al ecosistema y la documentación; nuestros equipos han comenzado a usarlos en escenarios donde otros contenedores han estado funcionando correctamente, como los agentes de compilación.

Seguimos preocupados por la lógica del negocio y la orquestación de procesos implementadas en middleware, especialmente donde se requiere habilidades de expertos y herramientas solamente para crear puntos de escalamiento y control. Los proveedores en el mercado altamente competitivo de gateways API continúan esta tendencia al agregar características para diferenciar a sus productos. Esto da como resultado productos de **GATEWAYS API DEMASIADO AMBICIOSAS** cuya funcionalidad —además de lo que es esencialmente un proxy invertido— fomenta diseños que continúan siendo difíciles de probar y desplegar. Las gateways API proveen utilidad con algunos problemas específicos — como la autenticación o la limitación del tráfico— pero cualquier dominio (lógica del negocio) debe quedarse en las aplicaciones o servicios.

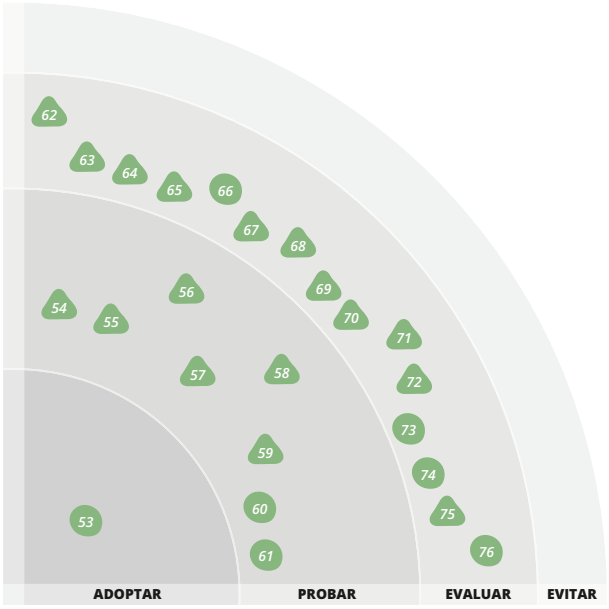
HERRAMIENTAS

ADOPTAR
53. fastlane

PROBAR
54. Buildkite *NUEVO*
55. CircleCI *NUEVO*
56. gopass *NUEVO*
57. Headless Chrome para pruebas front-end *NUEVO*
58. jsoniter *NUEVO*
59. Prometheus
60. Scikit-learn
61. Framework sin servidores

EVALUAR
62. Apex *NUEVO*
63. assertj-swagger *NUEVO*
64. Cypress *NUEVO*
65. Flow *NUEVO*
66. InSpec
67. Jupyter *NUEVO*
68. Kong API Gateway *NUEVO*
69. kops *NUEVO*
70. Lighthouse *NUEVO*
71. Rendertron *NUEVO*
72. Sonobuoy *NUEVO*
73. spaCy
74. Spinnaker
75. Spring Cloud Contract *NUEVO*
76. Yarn

EVITAR



A nuestros equipos les gusta mucho **BUILDKITE**, una herramienta de alojamiento de CI/CD, por su sencillez y fácil instalacion. Con Buildkite, uno proporciona sus propias máquinas para compilar y construir la aplicación, en sitio o en la nube, e instalar una aplicación agente liviana que conecte el agente de compilación con el servicio alojado. En muchos casos, tener este nivel de control sobre la configuración de los agentes es positivo al comparar con el uso de agentes alojados.

CIRCLECI es un motor de integración continua ofrecido como SaaS y en sitio. CircleCI ha sido la herramienta de CI como SaaS de nuestros equipos de desarrollo, que necesitaban pipelines de compilación y despliegue de baja fricción y fácil configuración. La versión de CircleCI 2.0 soporta flujos de tareas de compilación, con flujos fan-in y fan-out y puertas manuales, así como, desarrollo para móviles. Esto permite que los desarrolladores puedan ejecutar pipelines localmente y sean fácilmente integrados con Slack y otros sistemas de notificación y alarma. Recomendamos investigar más acerca de las prácticas de seguridad de CircleCI, tal como lo harían con cualquier otro producto SaaS que alojan los activos de su empresa.

Un descendiente de pass, gopass agrega características como soporte para administración de destinatarios y múltiples tiendas de contraseñas en un solo árbol; una funcionalidad de búsqueda interactiva; soporte para contraseñas de un solo usp basado en tiempo (TOTP); y almacenamiento de datos binarios.

(gopass)

GOPASS es una solución de gestión de contraseñas para equipos, basado en GPG y en Git. Es un descendiente de pass y agrega opciones tales como: soporte para gestión de destinatarios y varios almacenes de contraseñas en un solo árbol,

funcionalidad de búsqueda interactiva, soporte para contraseñas de una sola vez y basadas en tiempo (TOTP) y almacenamiento de datos binarios. La migración de su almacén de contraseñas es muy sencilla, porque gopass es compatible en gran medida con el formato usado por pass. Esto también significa que se puede lograr integrar con los procesos de aprovisionamiento mediante una sola llamada al secreto guardado.

A partir de mediados de 2017, los usuarios de Chrome han tenido la opción de ejecutar el navegador en modo sin encabezado. Esta opción es especialmente adecuada para ejecutar pruebas de front-end en el navegador, sin la sobrecarga del despliegue de acciones en una pantalla. Anteriormente, esto era algo solo disponible en PhantomJS pero Headless Chrome (sin encabezado) está reemplazando rápidamente al enfoque de WebKit impulsado por JavaScript. Las pruebas con Headless Chrome deben ejecutarse mucho más rápido y el comportamiento será más parecido a un navegador real, pero nuestros equipos se han dado cuenta que utiliza más memoria que PhantomJS. Con todas estas ventajas, **HEADLESS CHROME PARA PRUEBAS FRONT-END** posiblemente se convertirá en el estándar de facto.

Si está buscando un codificador/decodificador JSON de gran rendimiento en Go y Java, revise la librería de código abierto **JSONITER**. La librería es compatible con el paquete de codificación estándar JSON en Go.

Hemos visto mejoras continuas y un aumento en la adopción de Prometheus, la herramienta de monitoreo y de base de datos de series de tiempo, que fue desarrollada originalmente por Soundcloud.

(Prometheus)

Hemos visto mejoras continuas y un aumento en la adopción de **PROMETHEUS**, la herramienta de monitoreo y de base de datos de series de tiempo, que fue desarrollada originalmente por Soundcloud. Prometheus soporta principalmente el modelo de extracción de HTTP, pero también permite generar alertas, que son una parte activa de su conjunto de herramientas operacionales. Cuando se redactó este artículo, Prometheus 2.0 estaba en la versión previa al lanzamiento y continúa evolucionando. Los desarrolladores de Prometheus han enfocado sus esfuerzos en las bases de datos de series de tiempo principales y la variedad de parámetros disponibles. Grafana se ha convertido en la herramienta preferida de visualización de panel para los usuarios de Prometheus y la herramienta se envía con soporte para Grafana. Nuestros equipos también consideran que el monitoreo de Prometheus complementa de buena manera la indexación y la capacidad de búsqueda de Elastic Stack.

APEX es una herramienta para fácilmente construir, implementar y gestionar funciones AWS Lambda. Con Apex, usted puede escribir funciones en lenguajes que AWS no admite de forma nativa, tales como Golang, Rust y otros. Esto se logra con el componente (shim) Node.js, que crea un proceso hijo y procesa eventos por medio de stdin y stdout. Apex tiene muchas características que mejoran la experiencia para el desarrollador, y específicamente nos gusta que se puedan realizar pruebas de funciones, de forma local, y ejecutar un simulacro de los cambios, antes de que se apliquen a recursos de AWS.

Una librería AssertJ, **ASSERTJ-SWAGGER** le permite validar el cumplimiento de la implementación de API con su contrato específico. Nuestros equipos utilizan assertj-swagger para detectar problemas cuando la implementación del endpoint de API cambia sin actualizar sus especificaciones de Swagger o deja de publicar la documentación actualizada.

La corrección de fallas punto a punto de un CI puede ser una experiencia complicada, especialmente en el modo sin encabezado. **CYPRESS** es una herramienta útil que ayuda a desarrolladores a construir pruebas punto a punto, de forma fácil, y registra todos los pasos de la prueba en un video de formato MP4. En vez de reproducir el problema en modo sin encabezado, los desarrolladores pueden ver el video de prueba con el fin de solucionarlo. Cypress es una plataforma potente y no solo una estructura de pruebas. Actualmente, hemos integrado su CLI con CI sin encabezado en nuestros proyectos.

FLOW es un inspector de tipo estático de JavaScript que permite agregar inspectores de escritura en la base de códigos, de forma progresiva. A diferencia de Typescript, que es otro lenguaje, Flow puede agregarse de forma gradual a una base de código de JavaScript que admite las ediciones 5ta, 6ta y 7ma de ECMAScript. Sugerimos añadir Flow a su proceso de integración

continua, y comience con el código que más le preocupa. Flow permite ver el código con más claridad, aumenta la fiabilidad de refactorización y detecta errores de escritura, de forma temprana durante la compilación.

En los últimos dos años, hemos notado un aumento constante en la popularidad de cuadernos de analítica. Estas son aplicaciones inspiradas en matemática que combinan texto, visualización y código en un documento computacional interactivo. En una edición previa, mencionamos a GorillaREPL, una variante de Clojure de estas aplicaciones. Pero el interés creciente en el aprendizaje de máquinas, junto con el surgimiento de Python como lenguaje de programación predilecto para los desarrolladores de este campo, ha centrado una atención específica en los cuadernos de Python. Entre estos, **JUPYTER** parece que está logrando la mayor popularidad en equipos de ThoughtWorks.

Kong es una gateway API de código abierto construido y patrocinado por Mashape, quienes también proveen una solución empresarial que integra Kong con sus herramientas patentadas para API de análisis y desarrollo. Estas pueden ser desplegadas en una variedad de configuraciones, como una gateway borde de API o un proxy de API interno. OpenResty, a través de sus módulos de Nginx, proporciona una base sólida y eficaz con complementos Lua para extensiones. Kong puede usar PostgreSQL para despliegues de una sola región o Cassandra para configuraciones multiregiones. Nuestros desarrolladores han disfrutado del alto rendimiento de Kong, su enfoque de API primero (que habilita la automatización de su configuración) y su facilidad de despliegue como contenedor. La **GATEWAY API KONG**, a diferencia de las gateways API demasiado ambiciosas, tiene un conjunto más pequeño de características pero implementa las funcionalidades esenciales de una gateway API como control del tráfico, seguridad, registros, monitoreo y autenticación. Estamos emocionados de evaluar Kong en una configuración de sidecar en un futuro próximo.

KOPS es una herramienta de línea de comando para crear y gestionar la producción de alta disponibilidad de grupos (cluster) de Kubernetes. Inicialmente se enfocaba en AWS y ahora tiene apoyo experimental para otros proveedores. Puede ayudarlo a comenzar las cosas rápidamente y a pesar de que unas pocas opciones (tales como actualizaciones continuas) todavía no se han desarrollado por completo, nos ha impresionado la comunidad.

LIGHTHOUSE es una herramienta escrita por Google para evaluar el cumplimiento de estándares de Progressive Web App por parte de aplicaciones web. La versión de Lighthouse 2.0 de este año agrega métricas de performance y revisiones de accesibilidad al paquete básico. Esta funcionalidad adicional se ha incluido ahora en las herramientas de desarrollador de Chrome estándar, bajo la pestaña de auditoría. Lighthouse 2.0 es otro beneficiario más del modo headless de Chrome. Provee una alternativa a Pa11y y herramientas similares para ejecutar revisiones de accesibilidad en un pipeline de integración continua, porque puede ejecutarse desde la línea de comando o por sí solo como una aplicación Node.js.

Un problema permanente de aplicaciones web llenas de JavaScript es cómo lograr que la parte dinámica de esas páginas esté disponible a motores de búsqueda. Históricamente, los desarrolladores han recurrido a una variedad de trucos, incluyendo un procesamiento en el lado del servidor con React, servicios externos o contenido previo al procesamiento. Ahora el nuevo modo headless de Google Chrome agrega un nuevo truco a la caja de herramientas, **RENDERTRON**, una solución de procesamiento headless de Chrome. Rendertron envuelve una instancia de Chrome headless en un contenedor de Docker, listo para desplegarse como un servidor HTTP independiente. Los Bots que no procesan JavaScript pueden dirigirse a este servidor para que lo procese por ellos. A pesar de que los desarrolladores no pueden siempre desplegar su propio proxy Chrome headless y la maquinaria de enrutamiento asociada, Rendertron simplifica el proceso de configuración y despliegue. Además provee un ejemplo de código de middleware para detectar y encaminar bots.

Un problema permanente de aplicaciones web llenas de JavaScript es cómo lograr que la parte dinámica de esas páginas esté disponible para motores de búsqueda. Los Bots que no procesan JavaScript pueden dirigirse a un servidor Rendertron para que los procese por ellos.

(Rendertron)

SONOBUOY es una herramienta de diagnóstico para ejecutar pruebas de cumplimiento de un endpoint a otro en cualquier grupo (cluster) de Kubernetes de forma no destructiva. El equipo de Heptio, que fue fundado por dos creadores de proyectos de Kubernetes, desarrollaron esta herramienta para garantizar que una amplia gama de distribuciones y configuraciones de Kubernetes cumplan con las mejores prácticas, y a la vez sigan la estandarización de código abierto de interoperabilidad de grupos. Estamos experimentando con Sonobuoy para que se ejecute como nuestro proceso de versión de infraestructura como código, además de realizar un monitoreo continuo de nuestras instalaciones de Kubernetes, para validar el comportamiento y estado de todo el grupo.

Si está implementando servicios de Java usando el framework Spring, podría querer considerar **SPRING CLOUD CONTRACT** para pruebas de contrato dirigidas por el cliente. El ecosistema actual de esta herramienta admite la verificación de las llamadas de cliente y la implementación del servidor contra contrato. En comparación con Pact, un conjunto de herramientas de contrato de código abierto y dirigidas por el cliente, no cuenta con la intermediación de los contratos ni el soporte de otros lenguajes de programación. Sin embargo, se integra bien con el ecosistema de Spring, por ejemplo en cuanto al enrutamiento de mensajes con la integración de Spring.

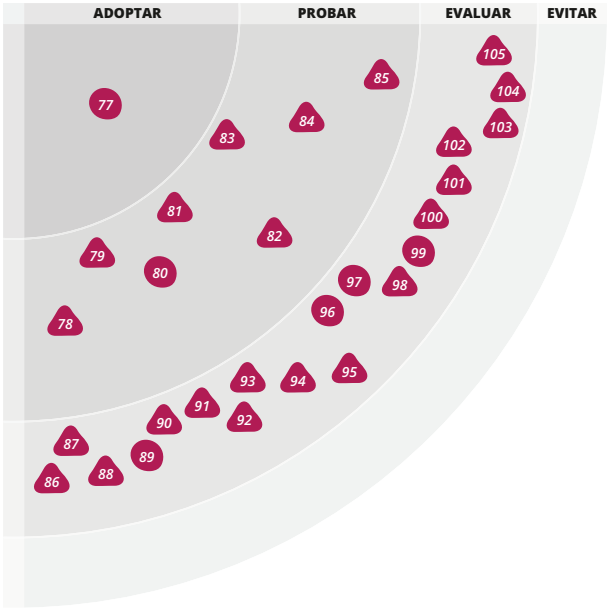
LENGUAJES Y FRAMEWORKS

ADOPTAR
77. Python 3

PROBAR
78. Angular
79. AssertJ *NUEVO*
80. Avro
81. CSS Grid Layout *NUEVO*
82. Módulos CSS *NUEVO*
83. Jest *NUEVO*
84. Kotlin
85. Spring Cloud

EVALUAR
86. Componentes de Aquitectura de Android *NUEVO*
87. ARKit/ARCore *NUEVO*
88. Atlas y BeeHive *NUEVO*
89. Caffe
90. Clara rules *NUEVO*
91. CSS-en-JS *NUEVO*
92. Digdag *NUEVO*
93. Druid *NUEVO*
94. ECharts *NUEVO*
95. Gobot *NUEVO*
96. Instana
97. Keras
98. LeakCanary *NUEVO*
99. PostCSS
100. PyTorch *NUEVO*
101. single-spa *NUEVO*
102. Solidity *NUEVO*
103. TensorFlow Mobile *NUEVO*
104. Truffle *NUEVO*
105. Weex *NUEVO*

EVITAR



En ediciones anteriores del Radar, hemos estado dubitativos de dar fuertes recomendaciones acerca de **ANGULAR** esencialmente porque era nuevo, y en general un monótono marco de trabajo que solamente compartía nombre con AngularJS, un viejo marco de trabajo que amábamos en el pasado. Mientras tanto, Angular, ahora con la versión 5, ha mejorado constantemente proporcionando compatibilidad con versiones anteriores a lo largo del camino. Varios de nuestros equipos tienen aplicaciones con Angular en producción y según dice, les gusta lo que ven. Por esta razón, estamos cambiando a Angular al anillo de Ensayar en este Radar, para mostrar que algunos de nuestros equipo ya lo consideran una opción sólida. Sin embargo, la mayoría de nuestros equipos todavía prefieren React, Vue o Ember sobre Angular.

ASSERTJ es una librería Java que provee una interfaz fluida para afirmaciones, lo que facilita que se exprese la intención en el código de pruebas. AssertJ presenta mensajes de error legibles, afirmaciones suaves, colecciones mejoradas y admite excepciones. Hemos notado que algunos equipos lo usan de forma predeterminada en vez de JUnit combinado con Hamcrest.

CSS es la opción preferida para distribuir páginas de web, incluso cuando no ofrecía mucho apoyo explícito en la creación de diseños. Flexbox ayudó en diseños más sencillos y de una sola dimensión, pero los desarrolladores, en general, se valían de librerías y juegos de herramientas para diseños más complejos. **CSS GRID LAYOUT** es un sistema de diseño basado en cuadrillas bidimensionales que tiene un mecanismo

para dividir el espacio disponible en columnas y filas, mediante el uso de comportamientos de fijación de tamaños predecibles. Grid no necesita librerías y funciona bien con Flexbox y otros elementos de despliegue de CSS. Sin embargo, como IE11 es admitido solo parcialmente, éste hace caso omiso de usuarios que todavía dependen de un navegador de Microsoft en Windows 7.

CSS Grid Layout es un sistema de diseño basado en cuadrillas bidimensionales que tiene un mecanismo para dividir el espacio disponible en columnas y filas, mediante el uso de comportamientos de fijación de tamaños predecibles.

(CSS Grid Layout)

La mayoría de bases de códigos grandes de CSS requieren esquemas complejos de nomenclatura para evitar conflictos de nombres en el espacio de nombres globales. Los **MÓDULOS CSS** abordan estos problemas al crear un solo alcance para todos los nombres de clase en un solo archivo de CSS. Este archivo se importa a un módulo de JavaScript, en donde las clases de CSS son referidas como cadenas. Luego, en el proceso de creación de la versión (Webpack, Browserify, etc.), los nombres de clases se reemplazan con cadenas únicas generadas. Este es un cambio significativo en las responsabilidades. Anteriormente, una persona tenía que gestionar el espacio de nombres globales, para evitar los conflictos de nombres de clases. Ahora la responsabilidad consiste en las herramientas de versiones. Hay una pequeña desventaja que hemos detectado en los módulos CSS. Las pruebas funcionales están, en general, fuera del alcance local y, por tanto, no pueden referirse a las clases por el nombre definido en el archivo de CSS. Recomendamos utilizar IDs o atributos de datos.

Nuestros equipos están encantados con los resultados del uso de **JEST** para pruebas de front-end. Provee una experiencia sin configuración y tiene opciones que vienen ya instaladas tales como mocks y cobertura de código. Puede aplicar este marco de pruebas no solo a aplicaciones de React, sino también a frameworks de JavaScript. Una de las opciones destacadas, con frecuencia, de Jest es las pruebas de capturas de interfaz de usuario. Las pruebas de capturas de interfaz serían un buen complemento para la capa superior de la pirámide de pruebas, pero recuerde que las pruebas unitarias son todavía la base sólida.

Jest es una herramienta de pruebas front-end de "configuración cero" con características fuera de la caja, tales como mocking y cobertura de código, dirigidas a React otros grameworks de JavaScript.

(Jest)

El anuncio del soporte de Android de primera clase brindó un impulso adicional a la rápida evolución del lenguaje **KOTLIN**, y estamos siguiendo de cerca el avance de Kotlin/Native; por ejemplo, la capacidad de LLVM-backed para compilar nativos ejecutables. Seguridad de referencias a nulos, clases de datos y la facilidad de crear DSLs son algunos de los beneficios que hemos aprovechado junto con la librería Anko para el desarrollo de Android. A pesar de las desventajas de una compilación inicial lenta y la dependencia en IntelliJ para soporte de IDE de primera clase, recomendamos intentar usar este lenguaje moderno, conciso e innovador.

SPRING CLOUD continúa evolucionando y agregando funcionalidades nuevas e interesantes. Soporte para vincular con Kafka Streams, por ejemplo, en el proyecto de spring-cloud-streams es relativamente fácil crear aplicaciones manejadas por mensajes con conectores para Kafka y RabbitMQ. Los equipos que lo están usando aprecian la sencillez que brinda cuando se utilizan infraestructuras complejas, a veces, tales como ZooKeeper, y permite el soporte de problemas comunes que necesitamos abordar cuando creamos sistemas distribuidos, gracias al rastreo con spring-cloud-sleuth, por ejemplo. Se aplican las advertencias usuales, pero lo estamos aplicando con éxito en varios proyectos.

Históricamente, los ejemplos en la documentación para Android de Google no tenían arquitectura ni estructura. Esto cambió cuando se introdujeron los **COMPONENTES DE ARQUITECTURA DE ANDROID** (*Android Architecture Components*), un conjunto de bibliotecas de clases que ayudan a desarrollar aplicaciones de Android con una mejor arquitectura. Abordan puntos que generaron problemas por algún

tiempo en el desarrollo de Android como la gestión de ciclos de vida, la paginación, bases de datos SQLite y la persistencia de datos cuando hay cambios de configuración. No hace falta utilizar todas las bibliotecas de clases juntas, sino que es posible escoger las que más se necesitan e integrarlas a los proyectos.

Los Componentes de Arquitectura de Android son un conjunto de bibliotecas de clases que ayudan a desarrollar aplicaciones de Android con una mejor arquitectura.

(Componentes de Arquitectura de Android)

Hemos notado muchas actividades en realidad aumentada para móviles. Gran parte es impulsada por **ARKIT Y ARCORE**, las librerías nativas de Realidad Aumentada (AR) usadas por Apple y Google, respectivamente. Estas librerías están logrando que las tecnologías de AR para móviles se vuelvan más comunes. Sin embargo, el reto será que las empresas encuentren casos de uso que no sean sensacionalistas y provean soluciones reales que en verdad mejoren la experiencia del usuario.

Una estrategia multi-apps es realmente controversial, particularmente en este tiempo cuando cada vez menos usuarios están descargando nuevas aplicaciones. En lugar de introducir una nueva aplicación y luchar contra los números de descarga, los multi-equipos tienen que entregar funcionalidades a través de una sola aplicación que esté ya instalada, lo cual crea un reto arquitectónico. **ATLAS Y BEEHIVE** son soluciones de modularización para aplicaciones Android y iOS, respectivamente. Atlas y BeeHive permite a los multi-equipos trabajar en módulos físicamente aislados para montar o cargar dinámicamente estos módulos desde una aplicación de fachada. Ambos son proyectos de código abierto de Alibaba, desde que Alibaba encontró el mismo problema de disminución de descargas y retos arquitectónicos de una sola aplicación.

Nuestra primera pauta general al seleccionar un motor de reglas es que no es necesario. Hemos visto que demasiada gente se enfoca en un motor de reglas de caja negra que es difícil de probar, por motivos espurios, cuando un código personalizado habría sido una mejor solución. A pesar de tal afirmación, hemos logrado buenos resultados al utilizar las **CLARA RULES** para escenarios en los cuales tiene sentido aplicar un motor de reglas. Nos agrada que utilice códigos sencillos de Clojure para evaluar y expresar las reglas, lo que significa que se puede refactorizar, probar y controlar la fuente. En lugar de tratar de conseguir que la gente de negocio manipule directamente las reglas, impulsa la colaboración entre expertos de negocios y desarrolladores.

Hemos logrado buenos resultados usando las Clara Rules para escenarios donde tiene sentido aplicar un motor de reglas. Nos gusta que aplique códigos sencillos de Clojure para expresar y evaluar reglas, lo que implica que puede refactorizar, probar y controlar la fuente.

(Clara rules)

CSS EN JS es una técnica de escritura de estilos de CSS en el lenguaje de programación de JavaScript. Esto fomenta un patrón común de desarrollo del estilo con el componente de JavaScript al cual se aplica, al co-localizar las consideraciones de presentación y lógicas. Los nuevos participantes, que incluyen JSS, emotion y componentes estilizados dependen de herramientas para transferir el código de CSS en JS a hojas de estilo separadas de CSS, para que sean idóneas para el consumo de navegadores. Este es el enfoque de segunda generación al desarrollo de CSS en JavaScript y a diferencia de los enfoques previos, este no depende de estilos en línea. Esto significa que provee el beneficio de admitir todas las opciones de CSS, y se puede compartir CSS usando el ecosistema npm y la utilización de componentes en varias plataformas. Nuestros equipos piensan que los componentes estilizados funcionan bien con los marcos de trabajos basados en componentes, tales como React, y las pruebas unitarias de CSS con componentes estilizados de Jest. Este espacio es nuevo y cambia rápidamente. El enfoque requiere algo de esfuerzo para la depuración de los nombres de clase generados en el navegador. Podría no aplicar a algunos proyectos en donde la arquitectura de front-end no admite la reutilización de componentes y se requieren estilos globales.

DIGDAG es una herramienta para crear, ejecutar, calendarizar y monitorear *pipelines* de datos en la nube. Estos *pipelines* pueden definirse en YAML, ya sea usando el amplio conjunto de operadores predefinidos

(*out-of-the-box*) o creando operadores por su cuenta por medio del API. Digdag muchas de las opciones comunes de una solución de *pipeline* de datos, tales como gestión de dependencias, flujos de trabajo modulares para promover la reutilización, gestión segura de secretos y la compatibilidad con varios idiomas. La opción que más nos emociona es el soporte de *polycloud* (múltiples nubes), que le permite trasladar y unir datos entre AWS RedShift, S3 y Google BigQuery. A medida que más proveedores de la nube ofrezcan soluciones rivales de procesamiento de datos, pensamos que Digdag (y herramientas similares) será útil para aprovechar las mejores opciones para determinadas tareas.

DRUID es un grupo de conexiones de JDBC con muchas opciones de monitoreo. Tiene un analizador de SQL, que provee monitoreo semántico de declaraciones de SQL que se ejecutan en la base de datos. Las inyecciones o declaraciones sospechosas de SQL se bloquearán y registrarán directamente en la capa de JDBC. Además, se fusionan las consultas con base en la semántica. Este es un proyecto de fuente abierta de Alibaba y refleja las lecciones aprendidas por este al operar sus propios sistemas de bases de datos.

ECHARTS es una librería liviana de gráficos y visualización con amplio soporte para distintos tipos de gráficos e interacciones. Debido a que ECharts se basa por completo en el API de Canvas, tiene un rendimiento increíble incluso cuando manipula más de 100K de datos. También tiene un funcionamiento optimizado para el uso con móviles. En conjunto con su proyecto hermano, ECharts-X, permite realizar trazados en 3D. ECharts es un proyecto de fuente abierta de Baidu.

La capacidad para compilar el Lenguaje de programación Go para objetivos bare-metal ha aumentado el interés entre los desarrolladores que utilizan este lenguaje en sistemas embebidos. **GOBOT** es un marco de trabajo para robótica, física informática, y el internet de las cosas (IoT), escrito en el lenguaje de programación Go y soportado en una variedad de plataformas. Hemos utilizado este marco de trabajo para proyectos de robótica experimentales donde la respuesta en tiempo real no ha sido un requerimiento y hemos creado controladores de software de código abierto con Gobot. Las APIs de Gobot HTTP habilita una integración simple de hardware con dispositivos móviles para crear aplicaciones más sofisticadas.

Nuestros equipos de aplicaciones móviles han estado emocionados por **LEAKCANARY**, una herramienta para detectar pérdida de memoria en Android y Java. Es sencillo incorporarlo y provee notificaciones con un rastreo claro de regreso hacia la causa de la pérdida. Si lo agrega a su juego de herramientas, puede ahora horas en la resolución de problemas, que son tediosas, para buscar el error de memoria agotada, en varios dispositivos.

PYTORCH es una versión completamente nueva del marco de aprendizaje de máquina Torch escrita en Python. A pesar de que es bastante nuevo e inmaduro comparado con Tensorflow, los desarrolladores piensan que es mucho más fácil trabajar con PyTorch. Debido a su orientación a objetos e implementación nativa de Python, se pueden expresar los modelos más claramente y de forma resumida, además de resolver errores durante la ejecución. A pesar de que muchos marcos han surgido recientemente, PyTorch tiene el respaldo de Facebook y una amplia gama de organizaciones socias, incluyendo NVIDIA, lo cual debe garantizar el apoyo continuo para arquitecturas CUDA. Los equipos de ThoughtWorks consideran que PyTorch es útil para experimentar y desarrollar modelos pero todavía dependen del desempeño de TensorFlow para capacitación según escala de producción y clasificación.

SINGLE-SPA es un metamarco de JavaScript que nos permite crear micro frontends usando diferentes marcos de trabajo que pueden coexistir en una misma aplicación. En general, no recomendamos utilizar más de un marco en una aplicación, pero a veces hay momentos en que no podemos evitarlo. Por ejemplo, Single-Spa puede aplicarse cuando se está trabajando en una aplicación legada y se desea experimentar con el desarrollo de una nueva funcionalidad, ya sea con una nueva versión del marco existente o con uno totalmente distinto. Debido a la duración corta de muchos marcos de trabajo de JavaScript, pensamos que hace falta una solución que permitiría cambios futuros de los marcos y la experimentación localizada, sin afectar toda la aplicación. Single-spa parece ser un buen inicio para lograrlo.

La programación para contratos inteligentes requiere un lenguaje más expresivo que el sistema de scripts de transacciones. **SOLIDITY** es el más popular de los nuevos lenguajes de programación diseñados para contratos inteligentes. Solidity es un lenguaje orientado a contratos, con tipado estático cuya sintaxis se asemeja a JavaScript. Provee la abstracción para redactar la lógica del negocio dentro de los contratos inteligentes. La cadena de herramientas

basada en Solidity está creciendo rápidamente. Actualmente, Solidity es la opción primaria de la plataforma Ethereum. Dada la naturaleza inmutable de los contratos inteligentes desplegados, se debe recalcar que las pruebas rigurosas y la auditoría de dependencias son esenciales.

TENSORFLOW MOBILE les permite a los desarrolladores incluir una amplia gama de técnicas de comprensión y clasificación en sus aplicaciones de iOS o Android. Esto es especialmente útil para la gama de datos de sensores disponibles en teléfonos celulares. Los modelos pre-entrenados de TensorFlow pueden cargarse en una aplicación móvil y aplicarse a datos de entrada tales como marcos de video en vivo, texto o archivos con audio. Los teléfonos celulares son una plataforma sorprendente y oportuna para la implementación de estos modelos computacionales. Los modelos de TensorFlows se exportan y cargan en archivos protobuf, que pueden producir algunos problemas para los encargados de la implementación. El formato binario de protobuf puede complicar la revisión de modelos y requiere que se vincule la versión de la librería de protobuf correcta a su aplicación móvil. Pero la ejecución del modelo local ofrece una alternativa atractiva a TensorFlow Serving sin la sobrecarga de comunicación causada por la ejecución remota.

TRUFFLE es un marco de trabajo que presenta una experiencia de desarrollo web moderna a la plataforma de Ethereum. Se encarga de la compilación de contratos inteligentes, los vínculos con librerías y la implantación, además de manejar los artefactos de redes de cadena de bloques. Uno de los motivos por los cuales nos encanta Truffle es que alienta a las personas a escribir pruebas para sus contratos inteligentes. Necesita enfocarse seriamente en las pruebas ya que la programación de contratos inteligentes se relaciona, con frecuencia, con el dinero. Gracias a su estructura de pruebas incorporada e integración con TestRPC, Truffle facilita la redacción de contratos usando TDD. Esperamos ver más tecnologías similares a Truffle para promover la integración continua en el área de cadenas de bloque.

WEEX es un marco de trabajo para crear aplicaciones móviles multi plataforma usando la syntaxis de componentes de Vue.js. Para aquellos que prefieren la sencillez de Vue.js, Weex es una opción factible de usarse en aplicaciones móviles nativas, pero también funciona bien en aplicaciones más complicadas. Hemos visto muchas aplicaciones exitosas con una gran complejidad que se basan en esta estructura, incluyendo TMall y Taobao, dos de las aplicaciones móviles más populares de China. Alibaba desarrolló Weex, y ahora se considera un proyecto incubador de Apache.

Sé el primero en saber cuando el Technology radar sea lanzado, y mantente al tanto de webinars y contenido exclusivo.

SUSCRÍBETE AHORA

thght.works/Sub-ES

ThoughtWorks®

ThoughtWorks es una consultora de tecnología y una comunidad de individuos apasionados guiados por propósitos. Ayudamos a nuestros clientes a incorporar la tecnología en el corazón de su negocio, y juntos creamos software relevante para ellos. Dedicados al cambio social positivo; nuestra misión es mejorar a la humanidad a través del software, y nos asociamos con varias organizaciones que luchan en la misma dirección.

Fundada hace más de 20 años, ThoughtWorks se ha convertido en una compañía de mñas de 4500 personas, incluyendo una división de productos que crea herramientas pioneras de software para equipos. ThoughtWorks cuenta con 42 oficinas en 15 países: Alemania, Australia, Brasil, Canadá, Chile, China, Ecuador, España, Estados Unidos, India, Italia, Reino Unido, Singapur, Sudáfrica y Turquía.

thoughtworks.com/es