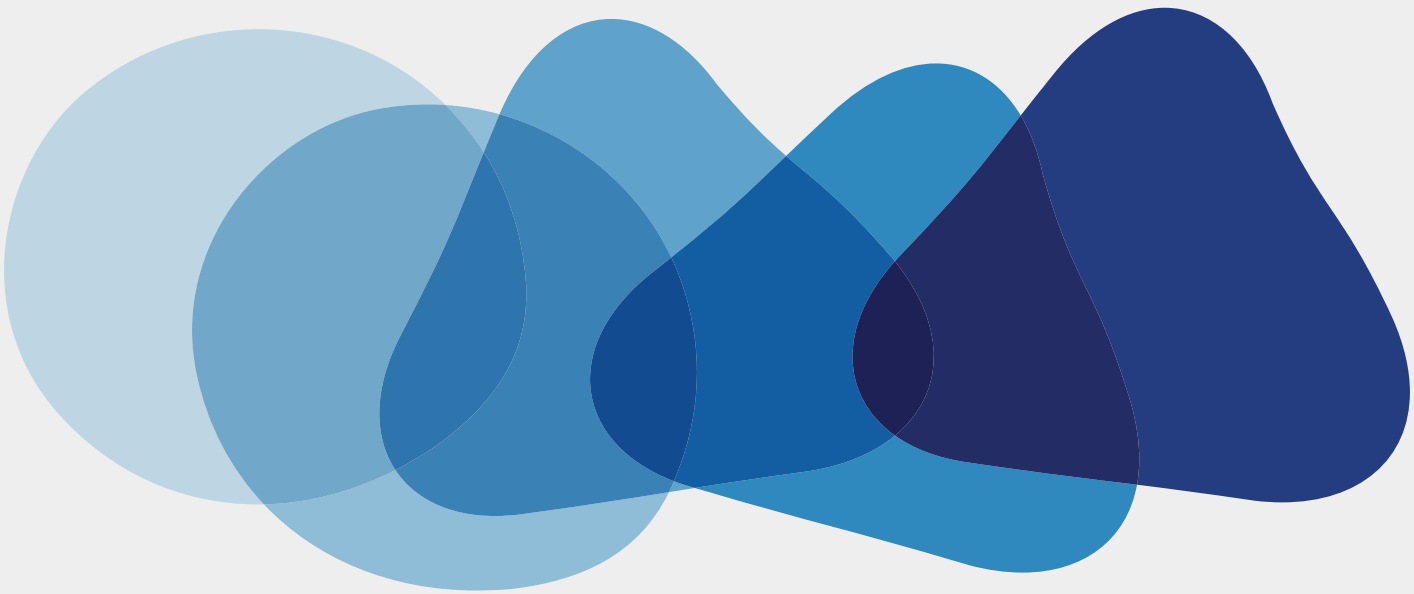
ThoughtWorks®

TECHNOLOGY RADAR *VOL.18*

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

Traducido por: Abraham Matus, Andrés Salazar, Carlos Andrés Oquendo, Carlos Ortiz, Carlos Valarezo, Daniel Aguilar, Daniel Santibañez, Danilo Burbano, David Corrales, Elena García Peña, Iván Ortega, Javier Villalobos, Jennifer Carrillo, Jonathan Morocho, José Puebla, Leonardo Venuti, Marcos Mercuri, María José Lalama, Mayrilis Castillo, Michael Soza, Orlando Hidalgo, Pamela Nuñez, Paola Jiménez y Tex Albuja

Esta edición del Radar Tecnológico de ThoughtWorks está basado en la reunión del *Technology Advisory Board*, que tuvo lugar en Sydney en Marzo del 2018

¿QUÉ HAY DE NUEVO?

Temas destacados en esta edición:

MIENTRAS LOS NAVEGADORES CRECEN LOS SERVIDORES SE REDUCEN

El navegador continúa expandiendo sus capacidades como destino para el despliegue de la lógica de la aplicación. A medida que las plataformas se ocupan de más preocupaciones transversales y requisitos no funcionales, se nota la tendencia a reducir la complejidad en la lógica del back-end. La introducción de WebAssembly abre nuevas opciones de lenguajes para crear la lógica de aplicaciones web y acerca el procesamiento al hardware (y a la GPU). Web Bluetooth permite a los navegadores manejar funcionalidades previamente reservadas para aplicaciones nativas, y vemos cada vez más estándares abiertos como CSS Grid Layout y CSS Modules suplantando bibliotecas personalizadas. La búsqueda de mejores experiencias para el usuario fomenta la tendencia de mover funcionalidades al navegador y como resultado, muchos servicios de back-end se reducen y se hacen menos complejos.

AUMENTO EN LA COMPLEJIDAD DE LA NUBE

Pese a que AWS sigue liderando el mercado con un conjunto sorprendente de nuevos servicios, vemos a Google Cloud Platform (GCP) y a Microsoft Azure madurar como alternativas viables. El uso de capas de abstracción como Kubernetes y prácticas como entrega continua facilitan la transición entre nubes al soportar cambios evolucionarios más sencillos. Sin embargo, las estrategias para la nube se tornan más complejas con la aparición de Polycloud (que permite a las organizaciones elegir múltiples proveedores en base a capacidades diferenciadas) y de mayores preocupaciones por regulaciones y de privacidad. Por ejemplo, varios países de la Unión Europea requieren por ley que los datos sean gestionados de manera local, convirtiendo a la jurisdicción donde se almacenan y las políticas de tenencia subyacentes en una nueva dimensión de diferenciación para los evaluadores de servicios en la nube. Las opciones para los ambientes de cómputo también se están incrementando con AWS Fargate que ofrece contenedores como servicio (CaaS) como un punto medio intrigante entre funciones como servicio (FaaS, Functions as a Service) y la gestión de clusters más durables. Aunque los recursos de la nube siguen madurando dentro de las organizaciones, un incremento inevitable en la complejidad siempre acompaña a la construcción de soluciones reales con estos nuevos elementos.

CONFIAR EN LOS EQUIPOS PERO VERIFICAR

La seguridad se mantiene como una preocupación importante para virtualmente todo desarrollo de software. Notamos un cambio en el enfoque tradicional de "limitar todo globalmente" hacia uno sutilmente diferente y más localizado. Ahora, muchos sistemas administran la confianza dentro de dominios mas pequeños y utilizan mecanismos modernos para crear una confianza transitiva entre sistemas disparejos. La filosofía de "nunca confiar en nada" fuera del dominio y "nunca verificar nada" dentro del dominio, ha cambiado a "confiar pero verificar" en todo lugar, es decir, asumir interacciones bien intencionadas con otras partes del sistema y verificar la confianza a nivel local. Esto permite que los equipos disfruten de altos niveles de control sobre su propia infraestructura, equipamiento y aplicaciones, permitiendo una alta visibilidad y, cuando sea necesario, una alta protección para el acceso. Herramientas como Scout2 y técnicas como BeyondCorp reflejan esta madura perspectiva respecto a la confianza. Damos la bienvenida a este cambio hacia una autonomía local, especialmente cuando las herramientas y la automatización permiten una cooperación similar o mejor.

LAS cosas EVOLUCIONAN

El ecosistema del Internet de las Cosas (IoT, Internet of Things) continúa evolucionando a un ritmo intenso y constante, e incluye factores críticos de éxito como la seguridad y buenas prácticas de ingeniería. Vemos crecimiento en todo el ecosistema de IoT, desde los sistemas operativos embebidos hasta los estándares de conectividad y, con más fuerza, en la administración de dispositivos y procesamiento de datos en la nube. Vemos madurez en las herramientas y en los frameworks que respaldan las buenas prácticas de ingeniería, como la entrega continua, despliegue y una serie de otros elementos necesarios para su posible uso generalizado. Además de los principales proveedores de servicios en la nube, que incluyen Google IoT Core, AWS IoT, y Microsoft Azure IoT Hub, empresas como Alibaba y Aliyun están invirtiendo fuertemente en soluciones PaaS (Platform as a Service) para IoT. Nuestros blips sobre EMQ y Mongoose OS brindan una idea de las capacidades principales del ecosistema actual de IoT y demuestran que, de hecho, las **cosas** están evolucionado muy bien.

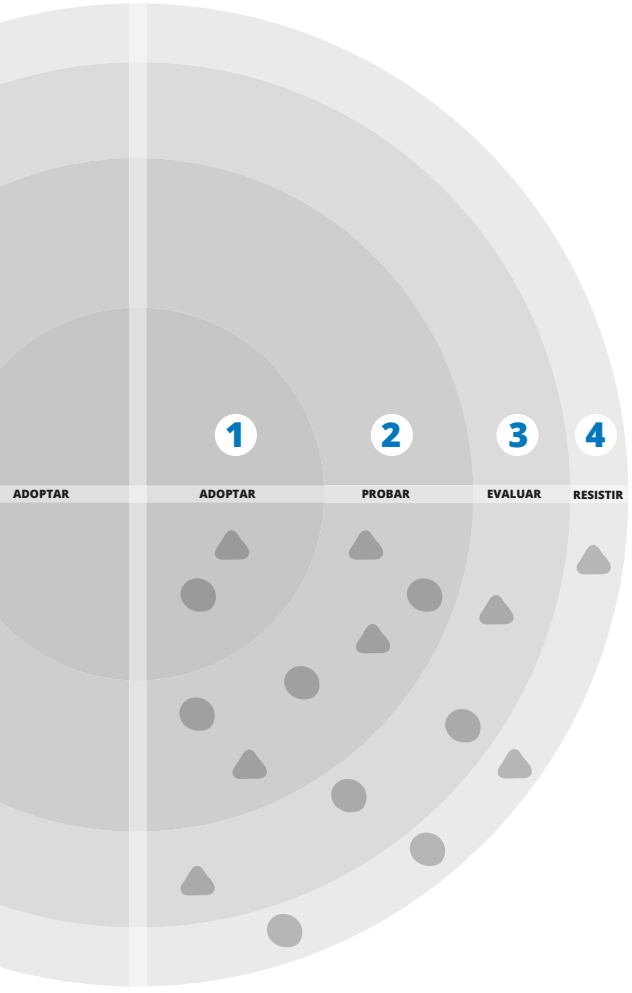
SOBRE EL RADAR

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 del Radar, visita thoughtworks.com/es/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 podría afectar a su empresa.

4 RESISTIR

Proceder con cautela

▲ NUEVO O MODIFICADO

● SIN CAMBIO

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 mas 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
3. Función de aptitud de arquitectura
4. Patrón de burbuja autónoma
5. Chaos Engineering
6. Eventos con alcance de dominio *NUEVO*
7. Administración de identidades alojada como servicio *NUEVO*
8. Micro frontends
9. Pipelines para infraestructura como código
10. Polycloud

EVALUAR
11. BeyondCorp *NUEVO*
12. Mocks móviles embebidos *NUEVO*
13. Ethereum para aplicaciones descentralizadas
14. Transmisión de eventos como fuente de la verdad
15. GraphQL para agregar recursos del lado del servidor *NUEVO*
16. Escáner de configuraciones de infraestructura *NUEVO*
17. Jupyter para Pruebas Automatizadas *NUEVO*
18. Nivel de registro de eventos por petición *NUEVO*
19. Ingeniería de Seguridad del Caos *NUEVO*
20. Service mesh
21. Sidecars para seguridad de endpoints
22. Las tres Rs de la seguridad

RESISTIR
23. Uso genérico de la nube *NUEVO*
24. Recreación de entornos ESB con Kafka

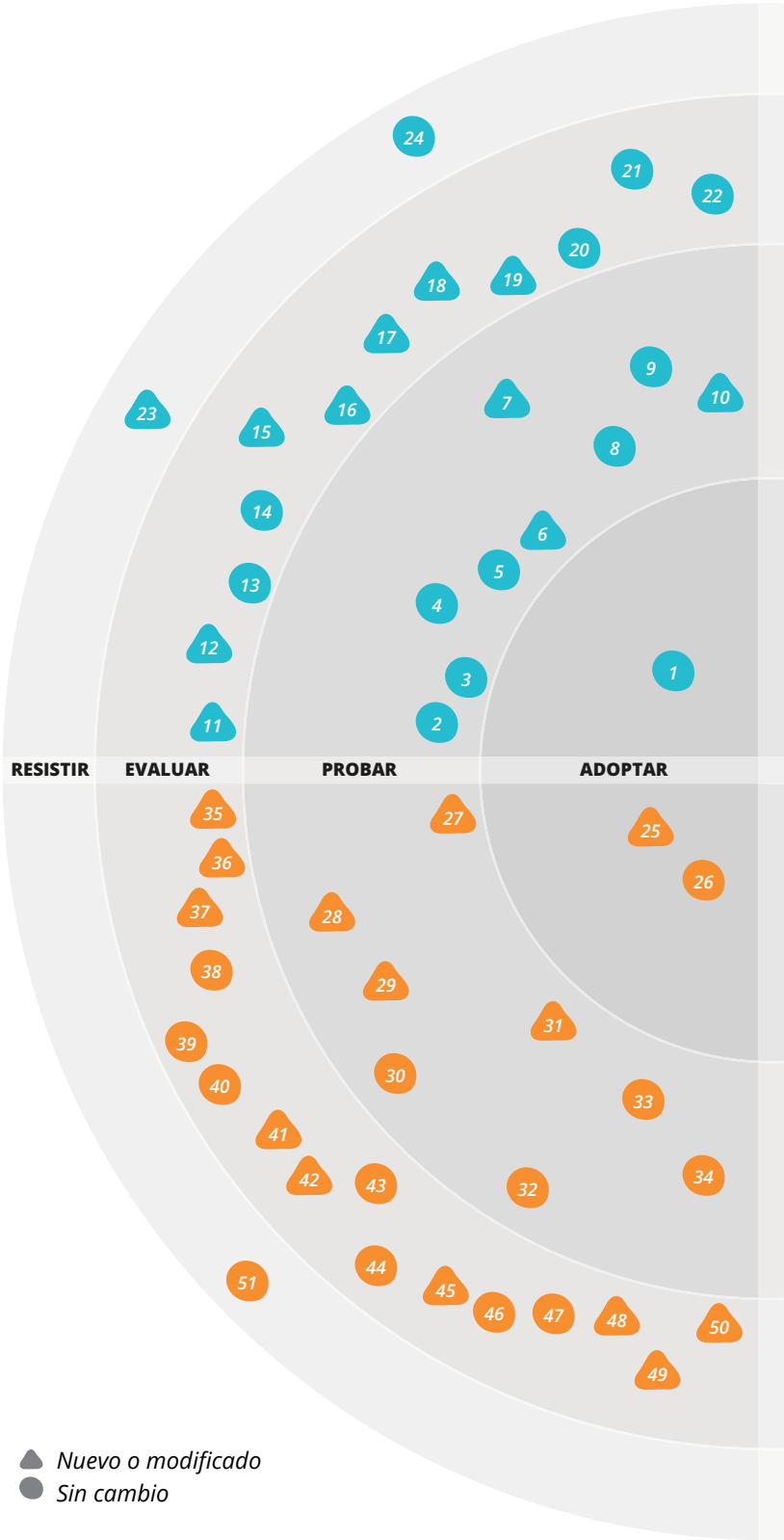
PLATAFORMAS

ADOPTAR
25. .NET Core
26. Kubernetes

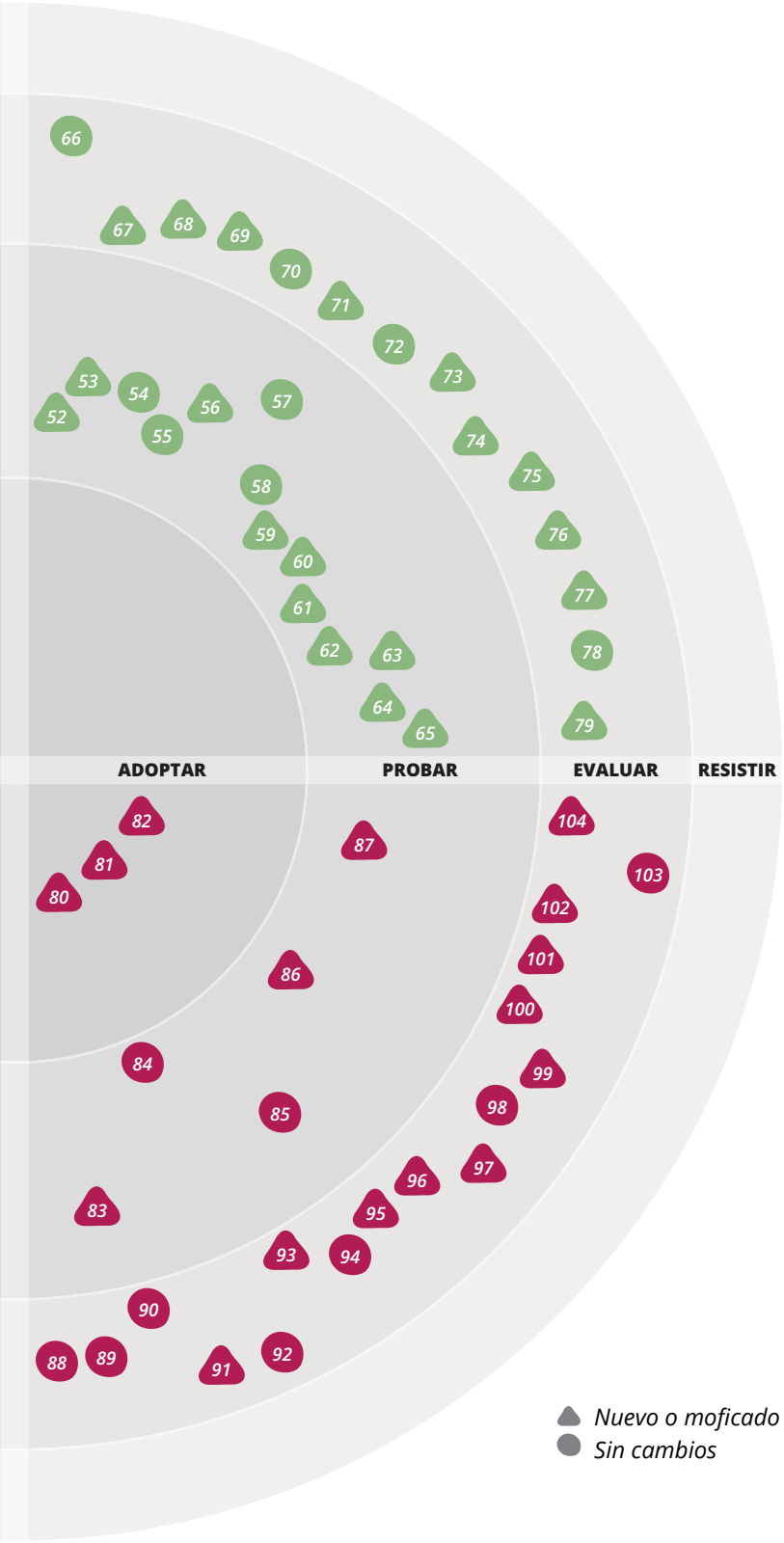
PROBAR
27. Azure
28. Contentful *NUEVO*
29. EMQ *NUEVO*
30. Flood IO
31. GKE
32. Google Cloud Platform
33. Keycloak
34. WeChat

EVALUAR
35. AWS Fargate *NUEVO*
36. Azure Service Fabric
37. Azure Stack *NUEVO*
38. Cloud Spanner
39. Corda
40. Cosmos DB
41. Godot *NUEVO*
42. Interledger *NUEVO*
43. Language Server Protocol
44. LoRaWAN
45. Mongoose OS *NUEVO*
46. Netlify
47. TensorFlow Serving
48. TICK Stack *NUEVO*
49. Web Bluetooth *NUEVO*
50. Windows Containers

RESISTIR
51. Gateways API demasiado ambiciosas



EL RADAR



HERRAMIENTAS

ADOPTAR

PROBAR
52. Appium Test Distribution *NUEVO*
53. BackstopJS *NUEVO*
54. Buildkite
55. CircleCI
56. CVXPY *NUEVO*
57. gopass
58. Headless Chrome para pruebas front-end
59. Helm *NUEVO*
60. Jupyter
61. Kong API Gateway
62. kops
63. Patroni *NUEVO*
64. WireMock *NUEVO*
65. Yarn

EVALUAR
66. Apex
67. ArchUnit *NUEVO*
68. cfn-nag *NUEVO*
69. Conduit *NUEVO*
70. Cypress
71. Dependabot *NUEVO*
72. Flow
73. Headless Firefox *NUEVO*
74. nsp *NUEVO*
75. Parcel *NUEVO*
76. Scout2 *NUEVO*
77. Sentry *NUEVO*
78. Sonobuoy
79. Swashbuckle for .NET Core *NUEVO*

RESISTIR

LENGUAJES Y FRAMEWORKS

ADOPTAR
80. AssertJ
81. Enzyme
82. Kotlin

PROBAR
83. Apollo *NUEVO*
84. CSS Grid Layout
85. CSS Modules
86. Hyperledger Composer *NUEVO*
87. OpenZeppelin *NUEVO*

EVALUAR
88. Componentes de Arquitectura de Android
89. Atlas and BeeHive
90. Clara rules
91. Flutter *NUEVO*
92. Gobot
93. Hyperapp *NUEVO*
94. PyTorch
95. Rasa *NUEVO*
96. Reactor *NUEVO*
97. RIBs *NUEVO*
98. Solidity
99. SwiftNIO *NUEVO*
100. Tensorflow Eager Execution *NUEVO*
101. TensorFlow Lite *NUEVO*
102. troposphere *NUEVO*
103. Truffle
104. WebAssembly *NUEVO*

RESISTIR

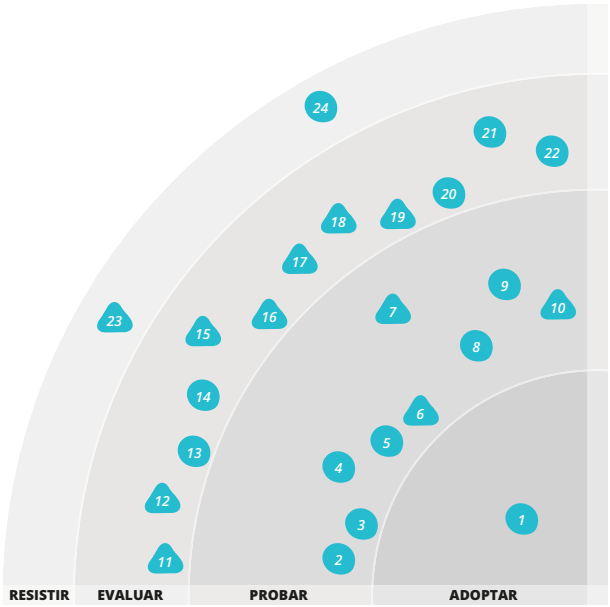
TÉCNICAS

ADOPTAR
1. Registros Livianos de Decisiones de Arquitectura

PROBAR
2. Aplicación de gestión de productos a plataformas internas
3. Función de aptitud de arquitectura
4. Patrón de burbuja autónoma
5. Chaos Engineering
6. Eventos con alcance de dominio *NUEVO*
7. Administración de identidades alojada como servicio *NUEVO*
8. Micro frontends
9. Pipelines para infraestructura como código
10. Polycloud

EVALUAR
11. BeyondCorp *NUEVO*
12. Mocks móviles embebidos *NUEVO*
13. Ethereum para aplicaciones descentralizadas
14. Transmisión de eventos como fuente de la verdad
15. GraphQL para agregar recursos del lado del servidor *NUEVO*
16. Escáner de configuraciones de infraestructura *NUEVO*
17. Jupyter para Pruebas Automatizadas *NUEVO*
18. Nivel de registro de eventos por petición *NUEVO*
19. Security Chaos Engineering *NUEVO*
20. Service mesh
21. Ingeniería de Seguridad del Caos
22. Las tres Rs de la seguridad

RESISTIR
23. Uso genérico de la nube *NUEVO*
24. Recreación de entornos ESB con Kafka



Es importante recordar que la encapsulación aplica a los eventos y arquitecturas dirigidas por eventos de la misma forma en que aplica a otras áreas del software. Específicamente, enfoquemonos sobre el alcance de un evento y si esperamos que sea consumido dentro de la misma aplicación, el mismo dominio o a lo largo de toda la organización un **EVENTO CON ALCANCE DE DOMINIO** será consumido dentro del mismo dominio en el que está publicado, por lo que esperamos que el consumidor tenga acceso a un cierto contexto, recursos o referencias para poder actuar sobre el evento. Si el consumo sucede más ampliamente dentro de una organización, los contenidos del evento bien podrían necesitar ser diferentes y debemos procurar no dejar "escapar" detalles de la implementación, de los que, posteriormente, otros dominios podrían llegar a depender.

La administración de identidades es un componente crítico de infraestructura. Los usuarios externos de aplicaciones móviles requieren ser autenticados, las personas desarrolladoras necesitan acceso a componentes de la infraestructura de entrega, y los microservicios pueden requerir auto identificarse con otros microservicios. En estos casos, hay que preguntarse si la administración de identidades debería ser alojada por uno mismo (*self-hosted)*. En base a nuestra experiencia, es preferible una solución de **ADMINISTRACIÓN DE IDENTIDADES ALOJADA COMO SERVICIO** (*SaaS, Software as a Service*). Nos parece que los proveedores alojados de primer nivel como Auth0 y Okta pueden proveer mejores tiempos de operación y acuerdos de nivel de servicio de seguridad. Dicho esto, en ocasiones alojar por uno mismo una solución es una decisión realista, especialmente para empresas que tienen los recursos y la disciplina operacional para hacerlo de forma segura. Las soluciones de identidad empresariales más grandes típicamente ofrecen un rango más extenso de capacidades como derechos centralizados (*entitlements),* reporte de gobernanza y separación de administración de tareas, entre otras. Sin embargo, estas preocupaciones son típicamente más relevantes para las identidades de empleados, especialmente en empresas reguladas con sistemas legados.

Las organizaciones se están sintiendo más cómodas con la estrategia **POLYCLOUD** - en lugar de atarse a un solo proveedor, prefieren colocar diferentes tipos de cargas de trabajo en diversos proveedores según

su propia estrategia. Algunas empresas escogen al proveedor que ofrezca el mejor servicio para un área en particular, por ejemplo: para colocar servicios estándar escogen AWS, pero prefieren Google para aplicaciones orientadas a datos y aprendizaje de máquina (*machine learning*), mientras que para aplicaciones Microsoft Windows se usa Azure. Para algunas organizaciones, esta es una decisión cultural y comercial. Las empresas de ventas a por menor, por ejemplo, con frecuencia se niegan a almacenar sus datos en Amazon y distribuyen la carga a diferentes proveedores en función de sus datos. Esto es diferente a una estrategia agnóstica a la nube, que se enfoca en la portabilidad entre los proveedores, lo que es costoso y obliga a pensar en aspectos que se convierten en el mínimo común denominador. En cambio, Polycloud se centra en usar el servicio que se ajuste mejor de la oferta de cada proveedor de servicios en la nube.

Algunas organizaciones están deshaciéndose de las intranets con confianza implícita y tratando a toda comunicación como si estuviese siendo transmitida por el Internet público.

(BeyondCorp)

Previamente en el Radar hemos discutido el surgimiento de las empresas sin límites. Ahora, algunas organizaciones están deshaciéndose de las intranets con confianza implícita y tratando a toda comunicación como si estuviese siendo transmitida por el Internet público. Un conjunto de prácticas, colectivamente etiquetadas como **BEYONDCORP**, han sido descritas por los ingenieros de Google en una serie de publicaciones. Colectivamente, estas prácticas, que incluyen a dispositivos administrados, redes 802.1x y *proxies* de acceso estándar que protegen servicios individuales, hacen a este un enfoque viable para la seguridad de redes en grandes empresas.

Durante el desarrollo de aplicaciones móviles, nuestros equipos normalmente no cuentan con un servidor externo para probar las aplicaciones. Configurar un mock del canal (*over-the-wire mock)* puede encajar bien en este escenario. El desarrollo de mocks HTTP y su compilación dentro del binario móvil para pruebas – **MOCKS MÓVILES EMBEBIDOS** – permite a los equipos probar sus aplicaciones móviles cuando están desconectadas y sin dependencias externas. Esta técnica puede requerir la creación de una librería dogmática basada en la librería de red usada por la aplicación móvil y el uso de la librería subyacente.

Un patrón que aparece una y otra vez al construir arquitecturas del tipo microservicios es como manejar la agregación de múltiples recursos en el lado del servidor. En años recientes, vimos emerger varios patrones como Backend for frontends (BFF) y herramientas como Falcor para manejar esto. En cambio nuestros equipos comenzaron a utilizar **GRAPHQL PARA AGREGAR RECURSOS DEL LADO DEL SERVIDOR**. Esto difiere de la manera usual de utilizar GraphQL donde los clientes consultan directamente a un servidor GraphQL. Al utilizar esta técnica, los servicios continuan exponiendo APIs RESTful pero por debajo los servicios agregados utilizan resolutores (*resolvers*) GraphQL como implementación para incorporar recursos de otros servidores. Esta técnica simplifica la implementación interna de servicios agregados o BFFs al usar GraphQL.

Por algún tiempo hemos recomendado incrementar la responsabilidad del equipo de entrega sobre todo el conjunto de tecnologías utilizadas, incluida la infraestructura. Esto significa que el propio equipo de entrega asume una mayor responsabilidad para configurar la infraestructura de una manera protegida, segura y conforme a regulaciones. Al adoptar estrategias para la nube, la mayoría de las organizaciones adoptan por defecto una configuración estrechamente bloqueada y administrada centralmente para reducir el riesgo, pero esto también crea importantes cuellos de botella en la productividad. Un enfoque alternativo es permitir a los equipos administrar su propia configuración y usar un **ESCÁNER DE CONFIGURACIONES DE INFRAESTRUTURA** para garantizar que la configuración se establezca de manera protegida y segura. Watchmen es una herramienta interesante creada para proporcionar, usando reglas, garantía de las configuraciones de las cuentas de AWS que son operadas y custodiadas de forma independiente por los equipos de entrega. Scout2 es otro ejemplo de escaneo de configuración que soporta verificación del cumplimiento de regulaciones seguro.

Vemos algunos reportes interesantes del uso de **JUPYTER PARA PRUEBAS AUTOMATIZADAS**. La habilidad para mezclar codigo, comentarios y salidas en el mismo documento nos recuerda a FIT, FitNesse y

Concordion . Este enfoque flexible es particularmente útil si las pruebas son pesadas en datos o dependen de algún análisis estadístico, como las pruebas de rendimiento. Python provee todo el poder que se necesita, pero como las pruebas crecen en complejidad, un modo para administrar conjuntos de cuadernos (*notebooks*) sería útil.

Un problema con la observabilidad en una arquitectura de microservicios altamente distribuida es la elección entre registrar todos los eventos (*logs*), y usar gran cantidad de espacio de almacenamiento, o registrar muestras aleatorias de eventos, y potencialmente perder eventos importantes. Recientemente hemos notado una técnica que ofrece un balance entre estas dos soluciones. Definir el **NIVEL DE REGISTRO DE EVENTOS POR PETICIÓN** mediante un parámetro enviado a través de un encabezado de seguimiento. Al usar un framework de seguimiento, posiblemente basado en el estándar OpenTracing, se puede pasar un identificador de correlación entre servicios en una sola transacción. Se puede incluso inyectar otros datos, como el nivel deseado para el registro de eventos, en la transacción de inicio, y pasarlos junto a la información de seguimiento. Esto garantiza que los datos adicionales recolectados correspondan a una sola transacción de usuario a medida que fluye por el sistema. Esta es también una técnica útil para la depuración, ya que los servicios podrían estar pausados o modificados en función de la transacción.

Deliberadamente introducimos falsos positivos en las redes de producción y otros elementos de la infraestructura, por ejemplo, dependencias en tiempo de construcción, para verificar si los procedimientos establecidos son capaces de identificar fallas de seguridad bajo condiciones controladas

(Security Chaos Engineering)

Hemos hablado previamente en el Radar acerca de la técnica denominada Ingeniería del Caos y también sobre el conjunto de herramientas Simian Army de Netflix, que hemos usado para ejecutar experimentos que prueban la resistencia de la infraestructura de producción. La **INGENIERIA DE SEGURIDAD DEL CAOS** amplía el alcance de esta técnica al ámbito de la seguridad. Deliberadamente introducimos falsos positivos en las redes de producción y otros elementos de la infraestructura, por ejemplo, dependencias en tiempo de construcción, para verificar si los procedimientos establecidos son capaces de identificar fallas de seguridad bajo condiciones controladas. Si bien es útil, esta técnica debe ser usada con ciudado para evitar que nuestros equipos se desensibilicen ante los problemas de seguridad.

Vemos a más organizaciones prepararse para usar múltiples nubes, no por beneficiarse de fortalezas o características particulares, sino para evitar a toda costa atarse a un único proveedor.

(Uso genérico de la nube)

Los principales proveedores de servicios de nube continuan agregando nuevas funcionalidades a un ritmo rápido, y bajo el nombre Polycloud hemos sugerido utilizar nubes de multiples proveedores en paralelo, para mezclar y combinar servicios basados en las fortalezas de las ofertas de cada proveedor. Vemos a más organizaciones prepararse para usar múltiples nubes, no por beneficiarse de fortalezas o características particulares, sino para evitar a toda costa atarse a un único proveedor. Esto, por supuesto, conduce hacia el **USO GENÉRICO DE LA NUBE**, al usar solo características que son entregadas por todos los proveedores, lo que nos recuerda el escenario de hace 10 años en el que las empresas evitaban usar muchas características avanzadas de las base de datos relacionales en un esfuerzo por mantenerse neutrales con respecto a los proveedores. El problema de la dependencia hacia un proveedor es real. Sin embargo, en vez de tratarlo de forma radical, recomendamos mirarlo desde la perspectiva de los costos de salida y compararlos con los beneficios de usar características específicas de una cierta nube.

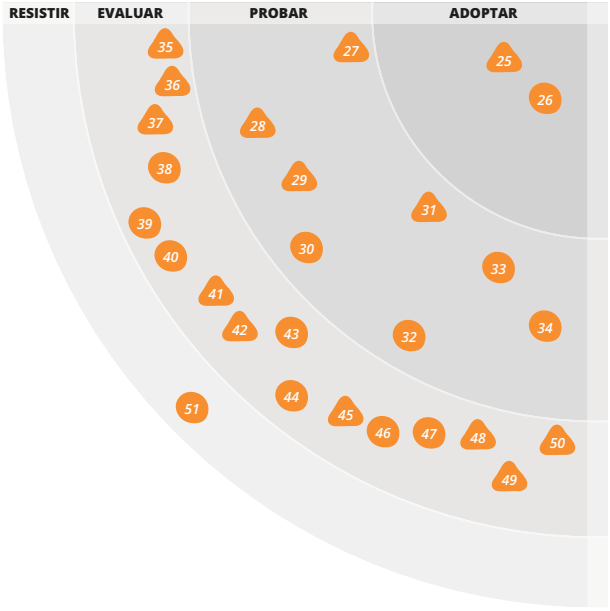
PLATAFORMAS

ADOPTAR
25. .NET Core
26. Kubernetes

PROBAR
27. Azure
28. Contentful *NUEVO*
29. EMQ *NUEVO*
30. Flood IO
31. GKE
32. Google Cloud Platform
33. Keycloak
34. WeChat

EVALUAR
35. AWS Fargate *NUEVO*
36. Azure Service Fabric
37. Azure Stack *NUEVO*
38. Cloud Spanner
39. Corda
40. Cosmos DB
41. Godot *NUEVO*
42. Interledger *NUEVO*
43. Language Server Protocol
44. LoRaWAN
45. Mongoose OS *NUEVO*
46. Netlify
47. TensorFlow Serving
48. TICK Stack *NUEVO*
49. Web Bluetooth *NUEVO*
50. Windows Containers

RESISTIR
51. Gateways API demasiado ambiciosas



Nuestros equipos han confirmado que **.NET CORE** ha alcanzado un nivel de madurez tal que lo convierte en la opción predeterminada para aplicaciones de servidor .NET. El framework de código abierto .NET Core permite el desarrollo y despliegue de aplicaciones .NET en Windows, macOS y Linux con herramientas multiplataforma de primera clase. Microsoft proporciona imágenes Docker certificadas, que simplifican el despliegue de aplicaciones .NET Core en un entorno basado en contenedores. Indicaciones positivas de la comunidad y la retroalimentación obtenida de nuestros proyectos indican que .NET Core es el futuro para el desarrollo en .NET

Microsoft ha seguido mejorando **AZURE** y hoy no hay mucho que separe la experiencia fundamental de las nubes de los mayores proveedores de servicios, como Amazon, Google y Microsoft. Los proveedores de servicios en la nube parecen estar de acuerdo y buscan diferenciarse en aspectos como características, servicios y estructura de costos. Microsoft es el proveedor de servicios que muestra un interés real en los requisitos legales de las compañías europeas. Ellos tienen una estrategia matizada y plausible, que incluye ofertas como Azure Alemania y Azure Stack, que brinda alguna seguridad a las compañías europeas con respecto a la GDPR y a posibles cambios de legislación en los Estados Unidos.

Los sistemas de gestión de contenido (CMS) sin interfaz gráfica (*headless)* se están volviendo en componentes comunes de las plataformas digitales. **CONTENTFUL** es un CMS *headless* moderno que nuestros equipos han integrado exitosamente en sus flujos de trabajo de desarrollo. Particularmente nos gusta su enfoque en las APIs y que implementa CMS como código. Este admite poderosas primitivas de modelado de contenido como código y *scripts* de evolución de modelos de contenido, que permiten tratarlo como otro esquema de almacenamiento de datos y aplicar prácticas de diseño

evolutivo de base de datos al desarrollo de CMS. Otras características notables que nos han gustado incluyen la inclusión de dos CDN por defecto para entregar recursos digitales (*media assets*) y documentos JSON, un buen soporte para la localización y la capacidad (aunque con cierto esfuerzo) de integrarse con Auth0.

EMQ es un broker MQTT multiplataforma y escalable de código abierto. Está escrito en Erlang/OTP para un mayor rendimiento, manejando millones de conexiones concurrentes. Soporta múltiples protocolos, incluyendo MQTT, MQTT Sensor Networks, CoAP así como WebSockets, lo que lo hace adecuado tanto para IoT como para dispositivos móviles. Hemos comenzado a usar EMQ en nuestros proyectos y hemos disfrutado de su facilidad de instalación y uso, su habilidad para enrutar mensajes a diferentes destinos, incluyendo a Kafka y PostgreSQL, así como su enfoque prioritario en las APIs para su monitoreo y configuración.

A medida que el ecosistema de desarrollo de software converge en Kubernetes como la plataforma de orquestación para contenedores, ejecutar *clusters* de Kubernetes sigue siendo complejo desde un punto de vista operacional. Google Kubernetes Engine (**GKE**) es una solución administrada para desplegar aplicaciones en contenedores que alivia la sobrecarga operacional de ejecutar y mantener *clusters* de Kubernetes. Nuestros equipos han tenido una buena experiencia usando GKE, con la plataforma a cargo de tareas difíciles como aplicar parches de seguridad, monitorear y reparar automáticamente los nodos y manejar redes entre múltiples *clusters* y regiones. En nuestra experiencia, la decisión de Google de enfocarse primero en las APIs para exponer capacidades de las plataformas, así como en usar estándares de la industria como OAuth para la autorización de servicios, mejora la experiencia para los desarrolladores. Es importante considerar que GKE está bajo desarrollo constante, con muchas de sus APIs en estado beta que, a pesar de los mejores esfuerzos de los desarrolladores para abstraer a los consumidores de los cambios subyacentes, pueden tener un impacto negativo. Esperamos mejoras continuas sobre la madurez de la Infraestructura como código con Terraform on GKE y herramientas similares.

AWS FARGATE es un ingreso reciente al espacio de Docker como Servicio y actualmente está limitada a la región US-East-1. Para equipos utilizando AWS Elastic Container Service (ECS), AWS Fargate es una buena alternativa donde no hay que gestionar, provisionar ni configurar ninguna instancia o *clusters* de EC2 subyacentes. Fargate permite definir tareas (ECS or EKS – ECS para Kubernetes) como un tipo Fargate, y se ejecutarán en la infraestructura AWS Fargate. Si te gusta el enfoque en funcionalidad de negocio que AWS Lambda provee, Fargate es lo más cercano que se puede tener cuando las aplicaciones no pueden ser desplegadas como funciones individuales.

AZURE SERVICE FABRIC es una plataforma de sistemas distribuidos construida para micro servicios y contenedores. Puede actuar como una Plataforma como Servicio (PaaS) con sus servicios confiables, o como un orquestador de contenedores por su habilidad de manejar contenedores. Sin embargo, lo que distingue a Service Fabric son los modelos de programación como Reliable Actors, construidos sobre servicios confiables. Cuando se trata de casos de uso del Internet de las Cosas (IoT), por ejemplo, Reliable Actors ofrece algunas ventajas convincentes como la capacidad de manejo de estados y replicación, además de la confiabilidad y los beneficios de la plataforma al estar en Service Fabric. Manteniendo su enfoque continuo en el software de código abierto (OSS), Microsoft realizará la transición de Service Fabric a un open development process on Github proceso de desarrollo abierto en Github. Todo esto hace que valga la pena probar Azure Service Fabric, especialmente para las organizaciones que han invertido en el Framework .NET.

Microsoft agrega una oferta interesante como punto medio entre las nubes públicas completas y la simple virtualización interna

(Azure Stack)

La computación en la nube trae beneficios significativos sobre soluciones de virtualización auto alojadas pero algunas veces los datos simplemente no pueden estar fuera de las instalaciones de una organización, usualmente por temas de latencia o disposiciones legales. Para las compañías europeas el actual clima político también plantea más preocupaciones sobre poner datos en las manos de empresas basadas en Estados Unidos. Con **AZURE STACK**, Microsoft agrega una oferta interesante como punto medio entre las nubes públicas completas y la simple virtualización interna: una versión ligera del software que ejecuta la nube global de Microsoft Azure combinada con hardware básico de proveedores como HP y Lenovo,

provee a la organización de la experiencia esencial de Azure pero dentro de las propias instalaciones. Por defecto, el soporte se divide entre Microsoft y los proveedores de hardware (y prometen cooperar), pero los integradores de sistemas también pueden ofrecer soluciones Azure Stack completas.

A medida que la realidad aumentada *(AR, augmented reality)* y la realidad virtual (*VR, virtual reality*) siguen ganando adherentes, continuamos explorando herramientas con las cuales podemos crear mundos virtuales inmersivos. La experiencia positiva que tuvimos con Unity, uno de los motores para juegos más importante, nos hizo incluirlo en ediciones anteriores del Radar. Todavía nos gusta Unity pero estamos emocionados por **GODOT** que debuta en el campo. Godot es software de código abierto y, a pesar que no tiene tantas características como los motores comerciales grandes, tiene un diseño más moderno y es más organizado. Al ofrecer soporte para C# y Python, a la larga se reduce aun más la barrera para desarrolladores fuera de la industria de los juegos. Godot versión 3, lanzado a inicios del año, agrega soporte para realidad virtual y el soporte para realidad aumentada está en el horizonte.

Muchas personas puede que conozcan a la "Internet del dinero" debido a Bitcoin. IDe hecho, esta idea ya aparece en las etapas iniciales de la Web. HTTP incluso reservó un código de estado para pagos digitales.

(Interledger)

Muchas personas puede que conozcan a la "Internet del dinero" debido a Bitcoin. De hecho, esta idea ya aparece en las etapas iniciales de la Web. HTTP incluso reservó un código de estado para pagos digitales. La parte desafiante de esta idea es transferir valor entre diferentes libros de transacciones (*ledgers*) en entidades diferentes. La tecnología Blockchain promueve esta idea mediante la construcción de un libro de transacciones distribuido. El reto actual es cómo lograr interoperabilidad entre diferentes libros de transacciones de *Blockchain* e interoperabilidad con libros de transacciones centralizados tradicionales. **INTERLEDGER** es un protocolo para conectar a diferentes libros de transacciones. Este protocolo utiliza conectores y un mecanismo criptográfico como HTLC para enrutar pagos seguros entre libros de transacciones. No es difícil unirse a la red de pagos mediante sus herramientas. Interledger fue concebido por Ripple y ahora se encuentra en desarrollo continuo por un grupo comunitario de la W3C.

Por el acelerado crecimiento de dispositivos embebidos conectados y un mayor acceso al hardware, **MOONGOSE OS** llena una notable brecha para los desarrolladores de software embebido: la brecha entre el *firmware* Arduino adecuado para el prototipado y los SDKs nativos de los microcontroladores. Mongoose OS es un sistema operativo para microcontroladores que viene con un conjunto de librerias y un framework de desarrollo para soportar, por defecto, las aplicaciones típicas del Internet de las Cosas (IoT) con conectividad hacia servidores genéricos MQTT y plataformas populares en la nube de IoT como Google Cloud IoT Core y AWS IoT. De hecho, Google recomienda un Mongoose starter kits para su Cloud IoT Core. Tuvimos una experiencia transparente utilizando Mongoose OS en nuestros proyectos embebidos de construcción de ambientes de trabajo conectados. Nos gustó especialmente la seguridad incorporada a nivel de dispositivos individuales y las actualizaciones OTA de *"firmware"*, entre otras características. Al momento de esta redacción, solamente un número limitado de microcontroladores y tableros son soportados mientras sigue el desarrollo de microcontroladores basados en ARM, una arquitectura más popular.

TICK STACK es una plataforma compuesta por componentes de código abierto que hacen que la recopilación, el almacenamiento, la representación gráfica y la alerta de series de datos *on-time*, como métricas y eventos, sea fácil. Los componentes de TICK Stack son: Telegraph, un agente de servidor para recoger y reportar métricas. InfluxDB, una base de datos de alto rendimiento de series de tiempo. Chronograf, una interfaz de usuario para la platforma; y Kapacitorm, un motor de procesamiento de datos que puede procesar, transmitir y poner en lotes datos desde InfluxDB. A diferencia de Prometheus, que se basa en un modelo *pull,* TICK Stack está basado en el modelo *push* para recolectar datos. El corazón del sistema es el componente InfluxDB, que es una de las mejores bases de datos para series de tiempo. Esta solución está respaldada por InfluxData y necesita la versión empresarial para habilitar funcionalidades como el agrupamiento de bases de datos, y continúa siendo una buena opción para monitoreo. Nosotros la estamos usando en algunos lugares en producción y hemos tenido buenas experiencias.

WEB BLUETOOTH nos permite controlar cualquier dispositivo Bluetooth de baja energía (*Low Energy*) directamente desde el navegador. Esto nos permite cubrir escenarios que anteriormente solo podían ser resueltos con una aplicación nativa. La especificación es publicada por el *Web Bluetooth Community Group* y describe un API para descubrir y comunicarse con dispositivos a través del estándar inalámbrico Bluetooth 4. En este momento, Chrome es el único navegador importante que soporta esta especificación. Con Physical Web y Web Bluetooth, ahora tenemos otras formas de hacer que los usuarios interactúen con dispositivos sin que tengan que instalar otra aplicación en su teléfono. Este es un tema emocionante que vale la pena observar.

Microsoft se está poniendo al día en el espacio de contenedores con **WINDOWS CONTAINERS** el permitir ejecutar aplicaciones de Windows como contenedores en entornos basados en Windows. Al momento de escribir estas líneas, Microsoft proporciona dos imágenes del sistema operativo Windows como contenedores Docker, Windows Server 2016 Server Core and Windows Server 2016 Nano Server, que se pueden ejecutar como Windows Server Container en Docker. Nuestro equipo empezó a utilizar los contenedores de Windows en escenarios donde agentes de compilación y contenedores similares se encontraban funcionando satisfactoriamente. Microsoft es consciente que debe realizar mejoras como reducir el tamaño de las imágenes y enriquecer el ecosistema soportado y la documentación.

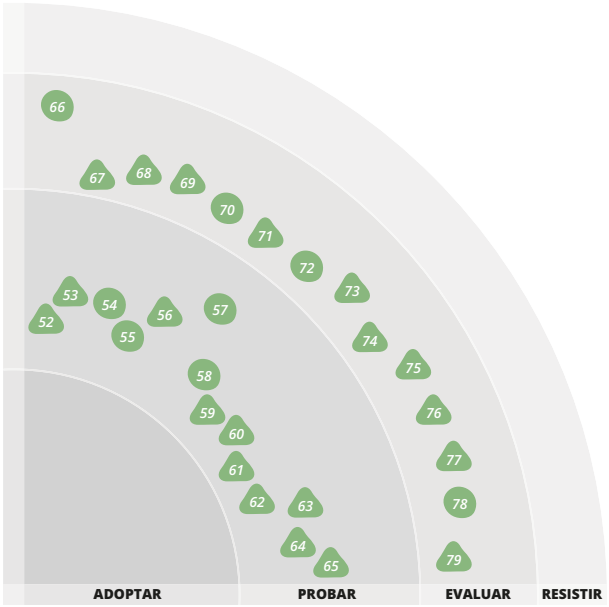
HERRAMIENTAS

ADOPTAR

PROBAR
52. Appium Test Distribution *NUEVO*
53. BackstopJS *NUEVO*
54. Buildkite
55. CircleCI
56. CVXPY *NUEVO*
57. gopass
58. Headless Chrome para pruebas front-end
59. Helm *NUEVO*
60. Jupyter
61. Kong API Gateway
62. kops
63. Patroni *NUEVO*
64. WireMock *NUEVO*
65. Yarn

EVALUAR
66. Apex
67. ArchUnit *NUEVO*
68. cfn-nag *NUEVO*
69. Conduit *NUEVO*
70. Cypress
71. Dependabot *NUEVO*
72. Flow
73. Headless Firefox *NUEVO*
74. nsp *NUEVO*
75. Parcel *NUEVO*
76. Scout2 *NUEVO*
77. Sentry *NUEVO*
78. Sonobuoy
79. Swashbuckle for .NET Core *NUEVO*

RESISTIR



Hemos destacado Appium en ediciones pasadas del Radar. Es uno de los frameworks de automatización de pruebas para aplicaciones móviles más populares. A medida que el conjunto de pruebas va aumentando, ser capaz de ejecutar las pruebas en paralelo contra una serie de dispositivos, es clave para tener ciclos de retroalimentación cortos. **APPIUM TEST DISTRIBUTION** soluciona este problema de manera muy efectiva, con su habilidad para ejecutar pruebas en paralelo, así como para ejecutar las mismas pruebas en diferentes dispositivos. Entre otras cosas, se distingue por su capacidad para añadir y eliminar dispositivos en los cuales las pruebas se ejecutan sin requerir ninguna configuración manual y también por su habilidad para ejecutar las pruebas en dispositivos remotos. En ThoughtWorks lo hemos usado en algunos proyectos en los últimos años y nos ha funcionado muy bien.

Hemos estando disfrutando de **BACKSTOPJS** para pruebas de regresión visual de aplicaciones web. Los *viewports* configurables y la habilidad de ajustar tolerancias son particularmente útiles, así como la herramienta de comparación visual que facilita la detección de variaciones menores. Puede ser fácilmente administrado por *scripts (scriptability)* y también tiene la opción de ejecutarse en Headless Chrome, PhantomJS and SlimerJS. Lo encontramos particularmente útil al momento de ejecutarlo contra guías vivas de estilos de componentes.

Es sorprendente la cantidad de problemas que pueden ser expresados como problemas de optimización matemática y a menudo de optimización convexa que pueden ser solucionados eficientemente.
(CVXPY)

Es sorprendente la cantidad de problemas que pueden ser expresados como problemas de optimización matemática y a menudo de optimización convexa, que pueden ser solucionados eficientemente. **CVXPY** es un lenguaje de código abierto integrado con Python que sirve para el modelado de problemas de optimización convexa. Es mantenido por la comunidad académica

de la Universidad de Stanford y ofrece un paquete que incluye varios solucionadores de código abierto y comerciales. La documentación incluye muchos ejemplos que deberían alentar a los desarrolladores a usuarlo. Es particularmente útil para realizar prototipos de soluciones, aunque puede ser necesario usar solucionadores con licencia comercial, como Gurobi o IBM CPLEX. De todas formas, en muchos casos, es suficiente por si mismo. Sin embargo, el mismo grupo ha escrito muchos paquetes de extensión como DCCP y software relacionado como CVXOPT basados en avances recientes en optimización.

HELM es un gestor de paquetes para Kubernetes. El conjunto de recursos de Kubernetes que definen una aplicación se empaqueta en *charts*. Estos *charts* pueden describir un único recurso, como un pod de Redis, o todos los elementos necesarios para una aplicación web, como servidores HTTP, bases de datos y caches. Helm, por defecto, cuenta con un repositorio saneado de aplicaciones Kubernetes, que son mantenidas en el repositorio oficial de *charts*. También es fácil configurar un repositorio privado de *charts* para uso interno. Helm tiene dos componentes: un cliente de linea de comandos llamado Helm, y un componente de cluster llamado Tiller. Proteger un cluster de Kubernetes es un tema amplio y con muchos matices, por lo que recomendamos encarecidamente configurar Tiller en un entorno con controles de acceso basados en roles (RBAC). Hemos utilizado Helm en varios proyectos de clientes y sus sistemas de gestión de dependencias, de plantillas y el mecanismo de bloqueo (*hook mechanism*) han simplificado mucho la gestion del ciclo de vida de aplicaciones en Kubernetes.

En los últimos años, hemos notado un incremento constante en la popularidad de los cuadernos de análisis. Estas son aplicaciones inspiradas en la Matemática que combinan texto, visualización y código en un archivo computacional.

(Jupyter)

En los últimos años, hemos notado un incremento constante en la popularidad de los cuadernos de análisis (*analytics notebooks*). Estas son aplicaciones inspiradas en Mathematica que combinan texto, visualizaciones y código en un documento computacional vivo. Un mayor interés en el aprendizaje de máquina (*machine learning),* junto con el posicionamiento de Python como el lenguaje de programación predilecto para los practicantes de este campo, ha centrado la atención en los cuadernos de Python, de los cuales **JUPYTER** parece estar ganando preferencia entre los equipos de ThoughtWorks. Las personas parece que siguen encontrando usos creativos para Jupyter más allá de usarlo como una simple herramienta de análisis. Por ejemplo, ver Jupyter para pruebas automatizadas.

Kong es una puerta de enlace (*gateway*) de código abierto para APIs que es también un producto empresarial que se integra con un API de analíticas propietario y un portal para desarrolladores. Kong puede ser desplegado en una variedad de configuraciones, como una puerta de enlace externa para APIs, como un proxy para APIs internas, o incluso como *sidecar* en una configuración de malla de servicios (*service mesh*). OpenResty, mediante sus módulos para Nginx, provee bases fuertes y de alto rendimiento con complementos en Lua a modo de extensiones. Kong puede usar PostgreSQL para despliegues en regiones individuales o Cassandra para configuraciones de regiones múltiples. Nuestros equipos de desarrollo han disfrutado del alto rendimiento de Kong, su enfoque prioritario sobre las APIs (lo que permite la automatización de su configuración) y su facilidad de despliegue como un contenedor. A diferencia de las puertas de enlace para APIs demasiado ambiciosas, **KONG API GATEWAY** tiene un conjunto menor de características pero implementa el conjunto esencial de capacidades de una puerta de enlace para APIs, como el control de tráfico, seguridad, registro de eventos, monitoreo y autenticación.

KOPS es una herramienta de línea de comandos para crear y administrar *clusters* de Kubernetes de alta disponibilidad para producción. Kops se ha convertido en nuestra herramienta más usada para auto administrar *clusters* en AWS, no solamente debido al rápido crecimiento de su comunidad de código abierto. Adicionalmente, soporta instalar, actualizar y administrar *clusters* de Kubernetes en Google Cloud. Sin embargo, nuestra experiencia con kops en Google es muy limitada debido a nuestra preferencia por GKE, la oferta administrada de Kubernetes. Recomendamos usar kops en *scripts* reutilizables para crear infrastructura como código. Estamos interesados en ver como kops continúa evolucionando para soportar *clusters* de Kubernetes tales como EKS el servicio

administrado de Kubernetes propio de Amazon.

PATRONI es una plantilla para utilizar PostgreSQL en alta disponibilidad. Nacido de la necesidad de proveer recuperación automática de fallos para PostgreSQL, Patroni es un controlador basado en Python que explota las capacidades de un almacén de configuraciones distribuido (como etcd, ZooKeeper, o Consul) para administrar el estado del *cluster* PostgreSQL. Patroni soporta modelos de replicación sincrónicos y de *streaming*, y proporciona un conjunto rico de APIs REST para la configuración dinámica del *cluster* PostgreSQL. Si se necesita conseguir alta disponibilidad en una instalación distribuida de PostgreSQL, hay que considerar muchos casos borde y nos gusta el hecho que Patroni provee una plantilla para solventar la mayoría de los casos más comunes.

Una de las claves para las arquitecturas basadas en micro servicios es la capacidad de evolucionar los servicios de manera independiente. Por ejemplo, cuando dos servicios dependen uno del otro, el proceso de pruebas para uno de ellos usualmente involucra usar un doble (*stubs* o *mocks*) del otro. Estos pueden ser escritos a mano, pero como sucede al crear dobles para las pruebas unitarias, un framework siempre ayuda a los desarrolladores a enfocarse concretamente en los escenarios de pruebas. Conocíamos de **WIREMOCK** desde hace tiempo, pero preferíamos correr las pruebas con mountebank. Durante el año pasado, WireMock ha mejorado mucho y lo recomendamos como una buena alternativa.

YARN es un gestor de paquetes para JavaScript que es rápido, confiable y seguro. Utilizando un archivo de bloqueo y un algoritmo determinista, Yarn garantiza que una instalación que funcionó en un sistema funcionará de la misma manera en cualquier otro. Al encolar eficientemente las peticiones, Yarn maximiza la utilización de la red y, como resultado, hemos visto descargas de paquetes más rápidas. Yarn sigue siendo nuestra herramienta preferida para la gestión de paquetes de JavaScript a pesar de las últimas mejoras en npm (versión 5).

ARCHUNIT es una librería Java para pruebas para validar características de arquitectura como dependencias entre paquetes y clases, verificación de anotaciones e incluso inconsistencias entre capas. Un punto positivo es el hecho de ArchUnit se ejecuta como parte de las pruebas unitarias existentes, aunque está disponibles únicamente para arquitecturas Java. El conjunto de pruebas de ArchUnit puede ser incorporado al ambiente de integración continua o al *pipeline* de despliegue, facilitando la implementación de funciones de aptitud como se lo hace en las arquitecturas evolutivas.

La nube y la entrega continua tienen un efecto dramático en la seguridad de la infraestructura. Cuando se aplica infrastructura como código, la infraestructura completa (que incluye redes, cortafuegos y cuentas) está definida en scripts y archivos de configuración, y con los servidores y ambientes fénix (*phoenix servers and environments*) la infraestructura se vuelve a crear en cada despliegue, a menudo varias veces al día. En este tipo de escenarios, probar la infraestructura luego de que es creada no es suficiente ni tampoco factible. Una herramienta que puede ayudar a enfrentar este problema es **CFN-NAG**. Esta herramienta escanea las plantillas de CloudFormation usadas con AWS en búsqueda de patrones que indicarían que la infraestructura es insegura, y lo hace antes de que ésta sea creada. Ejecutar herramientas como cfg-nag en un *pipeline* de construcción es rápido y puede detectar una serie de problemas antes de que alcancen los ambientes en la nube.

CONDUIT es unat malla de servicios ligera para Kubernetes. Conduit adopta la arquitectura "fuera de proceso" (*out-of-process*) con un proxy del plano de datos (*data plane proxy*) escrito en Rust y un plano de control (*control plane*) en Go. El proxy del plano de datos se ejecuta como un sidecar para todo el tráfico TCP en el cluster de Kubernetes y el plano de control corre en un espacio de nombres (*namespace*) separado, exponiendo APIs REST para controlar el comportamiento del proxy de plano de datos. Al procesar todas las peticiones, Conduit provee un conjunto muy valioso de métricas para el monitoreo y la observabilidad de las interacciones en la malla de servicios para tráfico HTTP, HTTP/2 y gRPC. Pese a que Conduit es relativamente nuevo en este espacio, nosotros lo recomendamos ya que es simple de instalar y operar.

Mantener dependencias actualizadas es un trabajo, pero es importante realizar actualizaciones frecuente e incrementalmente. Queremos que el proceso sea lo más automático y simple posible. Nuestros equipos con frecuencia tienen *scripts* rudimentarios que automatizan parte del proceso. Sin embargo, ahora integramos algunas ofertas comerciales que

hacen ese trabajo. **DEPENDABOT** es un servicio que se integra con los repositorios de GitHub y automáticamente busca nuevas versiones de las dependencias de un proyecto. Cuando sea necesario, Dependabot abrirá un *pull request* con dependencias actualizadas. Usando características del servidor de integración continua, se puede, de manera automática, probar la compatibilidad de las actualizaciones y combinar las que sean compatibles en *master*. Hay alternativas a Dependabot, incluyendo Renovate para proyectos en JavaScript y Depfu para proyectos en JavaScript y Ruby. Sin embargo, nuestros equipos recomiendan Dependabot debido a su soporte multilenguaje y a su facilidad de uso.

Para desarrollar aplicaciones *front-end*, hemos mencionado Headless Chrome como una alternativa mejor a Phantom JS para pruebas del *front-end* en una edición previa del Radar. Ahora sugerimos evaluar **HEADLESS FIREFOX** como una opción viable para esta área. De la misma manera que Headless Chrome, Firefox trabaja sin los componentes visibles de UI, ejecutando el conjunto de pruebas de UI mucho más rápido.

NSP es una herramienta de línea de comandos para identificar vulnerabilidades conocidas en aplicaciones basadas en Node.js. Al ejecutar el comando **`check`** en la raíz del proyecto Node.js, nsp genera el reporte de vulnerabilidades tomando en cuenta los avisos publicados. Nsp brinda una forma de personalizar el comando **`check`** para esconder todas las vulnerabilidades bajo un limite CVSS dado o terminar con código de error si al menos una de las vulnerabilidades encontradas tiene un puntaje CVSS sobre el valor provisto. Una vez que los resultados son guardados a través del comando **`gather`**, nsp puede ser utilizado sin conexión.

PARCEL es un empaquetador (*bundler*) de aplicaciones web similar a Webpack o Browserify. Webpack ya ha sido parte de nuestro Radar y sigue siendo una gran herramienta. Parcel se distingue de sus rivales por la experiencia para los desarrolladores y la velocidad. Brinda todas las caracteristícas estándar de empaquetado y una experiencia real de cero configuración, por lo que es muy fácil comenzar a usarla. Tiene tiempos de rápidos de empaquetado y supera a sus competidores en muchos puntos de referencia. Parcel ha ganado mucho interés de la comunidad y vale la pena tenerlo en cuenta.

SCOUT2 es una herramienta de auditoría de seguridad para ambientes AWS. En lugar de navegar manualmente por las páginas web, Scout2 puede obtener todos los datos de configuración de un ambiente AWS; incluso genera un reporte sobre la potencial superficie de ataque. Scout2 incluye reglas preconfiguradas y pueden ser fácilmente extendidas para soportar más servicios y casos de prueba. Ya que Scout2 solo realiza llamadas a APIs de AWS para obtener los datos de configuración e identificar brechas de seguridad, no es necesario completar y enviar el Formulario de solicitud de pruebas de intrusión/vulnerabilidad de AWS (*AWS Vulnerability / Penetration Testing Request Form*).

SENTRY es una herramienta de rastreo de errores que ayuda a monitorear y corregir errores en tiempo real. Las herramientas de gestión y rastreo de errores como Sentry se diferencian de las soluciones de registro de eventos tradicionales, como ELK Stack, por su enfoque en el descubrimiento, investigación y corrección de errores. Sentry ha estado presente por algún tiempo y es muy popular — las herramientas de rastreo de errores se están volviendo más útiles con su enfoque en el "tiempo promedio de recuperación". Sentry, con sus opciones de integración con GitHub, Hipchat, Heroku, Slack, etc., nos permite mirar más de cerca a nuestras aplicaciones. Puede proveer notificaciones de errores luego de un paso a producción, lo que nos permite rastrear si los cambios efectuados realmente corrigen un error y nos alerta si un problema vuelve debido a una regresión.

En el estado actual de los servicios de tecnología, exponer APIs RESTFul es cada vez mas adoptado y la documentación de las APIs es muy importante para los consumidores.

(Swashbuckle for .NET Core)

En el estado actual de los servicios de tecnología, exponer APIs RESTFul es cada vez mas adoptado y la documentación de las APIs es muy importante para los consumidores. En este mundo, Swagger ha sido muy ocupado por muchos equipos y quisiéramos destacar a **SWASHBUCKLE FOR .NET CORE**. Esta es una herramienta que genera documentación viva de las APIs en Swagger, basada en el código para proyectos .NET Core. Cuando se la usa, también puedes explorar y probar operaciones de la API a traves de su interfaz de usuarioCeps, ditabutum dion se dius; nosteris.

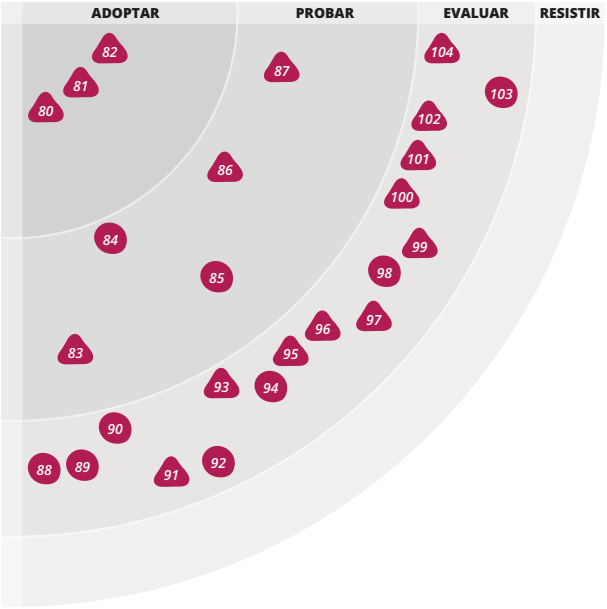
LENGUAJES & FRAMEWORKS

ADOPTAR
80. AssertJ
81. Enzyme
82. Kotlin

PROBAR
83. Apollo *NUEVO*
84. CSS Grid Layout
85. CSS Modules
86. Hyperledger Composer *NUEVO*
87. OpenZeppelin *NUEVO*

EVALUAR
88. Componentes de Arquitectura de Android
89. Atlas and BeeHive
90. Clara rules
91. Flutter *NUEVO*
92. Gobot
93. Hyperapp *NUEVO*
94. PyTorch
95. Rasa *NUEVO*
96. Reactor *NUEVO*
97. RIBs *NUEVO*
98. Solidity
99. SwiftNIO *NUEVO*
100. Tensorflow Eager Execution *NUEVO*
101. TensorFlow Lite *NUEVO*
102. troposphere *NUEVO*
103. Truffle
104. WebAssembly *NUEVO*

RESISTIR



ASSERTJ es una libreria de Java que proporciona una interfaz fluida para *assertions*, lo que hace que sea sencillo transmitir el propósito del código de prueba. AssertJ ofrece mensajes de error legibles, *soft assertions* y soporte mejorado para colecciones y excepciones. Muchos de nuestros equipos eligen AssertJ como su libreria de *assertions* predeterminada en lugar de usar JUnit combinado con Java Hamcrest.

ENZYME se ha convertido en el estándar para pruebas unitarias para componentes de interfaz de usuario de React. A diferencia de otras herramientas de pruebas basadas en capturas de pantalla (*snapshot tests*), Enzyme permite hacer pruebas sin efectuar la representación visual (*rendering*) en dispositivo, lo que permite hacer pruebas más detalladas y rápidas. Este es un factor que contribuye a nuestra capacidad para reducir de forma masiva la cantidad de pruebas funcionales que normalmente tenemos que hacer en aplicaciones React. En muchos de nuestros proyectos es utilizado dentro de un framework de pruebas unitarias como Jest.

KOTLIN ha experimentado una tasa acelerada de adopción y un rápido crecimiento de herramientas de soporte. Algunas de las razones detrás de su popularidad son: sintaxis concisa, seguridad contra valores nulos (*null safety*), facilidad de transición desde Java e interoperabilidad con otros lenguajes basados en la JVM en general, y que funciona como un excelente lenguaje introductorio para la programación funcional. Con JetBrains agregando la capacidad de compilar Kotlin a binarios nativos en múltiples plataformas, así como traducir (*transpile*) a JavaScript, creemos que tiene un potencial de un uso mucho más amplio por parte de la gran comunidad de desarrolladores de aplicaciones móviles y nativas. Aunque al momento de escribir estas líneas, algunas herramientas como el análisis estático y de cobertura de código no han alcanzado aún su madurez, dada nuestra experiencia en el uso de Kotlin en muchas aplicaciones en producción, creemos que Kotlin está listo para su adopción general.

Desde que fue introducido en el Radar, hemos visto una firme adopción de GraphQL, pparticularmente como una interface remota para Backend for Frontend (BFF). A medida que ganan más experiencia, nuestros equipos han llegado a un consenso sobre Apollo (un cliente GraphQL) como la manera preferida de acceder a datos GraphQL desde una aplicación React. A pesar que el proyecto **APOLLO**, también provee un framework para servidores y un gateway GraphQL, el cliente Apollo simplifica el problema de vincular los componentes de la Interfaz de Usuario a los datos servidos por cualquier backend GraphQL. Notablemente, Apollo es utilizado por Amazon AWS en su reciente lanzamiento del nuevo servicio AWS AppSync.

El proyecto Hyperledger ha crecido en colaboración y ahora tiene una serie de sub-proyectos. Soporta implementaciones de Blockchain para distintos propósitos; por ejemplo, Burrow está dedicado a construir un Ethereum con permisos, e Indy está más enfocado en la identidad digital. Entre estas plataformas, Fabric es la más madura. La mayor parte de las veces que la gente habla de adoptar Hyperledger, realmente están pensando en Hyperledger Fabric. Sin embargo, la abstracción programática de chaincode es relativamente de bajo nivel, dado que manipula el estado del libro de transacciones (*ledger*) directamente. Además, siempre se tarda bastante en configurar la infrastructura antes de escribir la primera linea de código de blockchain. **HYPERLEDGER COMPOSER**, que construye encima de Fabric, acelera el proceso de convertir ideas en software. Composer provee DSLs para modelar activos de negocio, definir controles de acceso y construir una red de negocio. Usando Composer se puede validar ideas rápidamente a través de un navegador sin configurar infraestructura. Hay que recordar que Composer no es Blockchain - y que debe desplegarse en Fabric.

La seguridad es la piedra angular en la economía de "blockchain".

(OpenZeppelin)

La seguridad es la piedra angular en la economía de *blockchain*. En la última publicación del Radar resaltamos la importancia de probar y auditar las dependencias de los contratos inteligentes (*smart contracts*). **OPENZEPPELIN** es un framework que ayuda a crear contratos inteligentes seguros en Solidity. El equipo detrás de OpenZeppelin resumió una serie de obstáculos y buenas prácticas para la seguridad de los contratos inteligentes e integró dichas experiencias en el código fuente. El framework está bien revisado y validado por la comunidad de código abierto. Recomendamos el uso OpenZeppelin en lugar de escribir implementaciones propias del *token* ERC20/ERC721. OpenZeppelin también se integra con Truffle.

FLUTTER es un framework multiplataforma que permite escribir aplicaciones móviles nativas en Dart. Se beneficia de Dart y puede ser compilado a código nativo y se comunica con la plataforma destino sin necesidad de puentes ni cambios de contexto, algo que puede causar cuellos de botella de rendimiento en frameworks como ReactNative o Weex. La función de recarga en tiempo real de Flutter es impresionante y proporciona retroalimentación visual muy rápida al editar el código. Actualmente, Flutter todavía se encuentra en fase beta, pero seguiremos monitoreándolo para ver como madura su ecosistema.

Dado el número de frameworks JavaScript que hemos presentado en el Radar a través de los años, nos preguntamos ¿de verdad necesitamos hablar de otro? Decidimos que **HYPERAPP** merece ser revisado por su enfoque minimalista. Tiene un tamaño muy pequeño, de menos de 1KB, y aún así cubre todas las funcionalidades esenciales para escribir una aplicación web. Esto solamente es posible con un diseño elegante que reduce todo hasta el mínimo absoluto, lo cual a su vez hace que el framework sea fácil de usar y entender. A pesar de ser relativamente nuevo, ha atraído a una comunidad bastante grande y recomendamos al menos considerarlo cuando se elija un framework para una aplicación nueva.

RASA es un debutante en el área de los *chatbots*. En lugar de usar un simple árbol de decisión, utiliza redes neuronales para mapear el significado y el estado interno a una respuesta. Rasa se integra con soluciones de procesamiento de lenguaje natural como spaCy, y a diferencia de otras soluciones que hemos mencionado en el Radar, Rasa es un software de código libre y puede ser auto alojado, lo que lo convierte en una solución viable cuando la propiedad de los datos es crítica. Nuestras experiencias utilizando Rasa Stack en una aplicación interna ha sido positiva.

REACTOR es una librería para construir aplicaciones no bloqueantes en la JVM, para la versión 8 y superiores, basada en la especificación de Reactive Streams. La programación reactiva enfatiza en cambiar de una lógica imperativa hacia un estilo de codificación asíncrono, no bloqueante y funcional, especialmente cuando se trabaja con recursos externos. Reactor

implementa la especificación de flujo reactivo y provee dos APIs publicadoras, Flux (0 a N elementos) y Mono (0 o 1 elementos), para modelar efectivamente el procesamiento de flujos en modalidad *push* (*push-based stream processing*). Reactor es muy adecuado para arquitecturas de microservicios y ofrece mecanismos de alta disponibilidad para manejar tráfico HTTP, WebSockets, TCP y UDP.

RIBS, acrónimo en inglés de *router*, *interactor* y *builder* (enrutador, interactor y constructor) es un framework de arquitectura móvil multiplataforma de Uber. La idea central de RIBs es desacoplar la lógica de negocio del árbol de vistas, y así asegurar que la app se conduce por la lógica de negocio. En realidad, esta es una aplicación de Clean Architecture en el desarrollo de aplicaciones móviles. Al aplicar patrones de arquitectura consistentes entre Android e iOS, RIBs proporciona una gestión clara de las sentencias y simplifica la escritura de pruebas. Recomendamos poner la lógica de negocio en los servicios de *backend* en lugar de filtrarla a las vistas, así que para una aplicación móvil complicada, RIBs puede ayudar a manejar esta complejidad.

Estamos a favor de los estilos de programación asíncronos y reactivos, especialmente para sistemas distribuidos que usan I/O en red. Las librerías reactivas a menudo se ubican sobre frameworks de comunicación no bloqueante de bajo nivel como Netty. Ultimamente **SWIFTNIO**, un framework de código abierto para comunicación en red no bloqueante de Apple nos ha llamado la atención. SwiftNIO es similar a Netty pero está escrito en Swift. Actualmente es soportado en macOS y Ubuntu e implementa HTTP como un protocolo de más alto nivel. Estamos emocionados de ver el uso de este framework y su integración con frameworks de aplicación de más alto nivel además de otros protocolos.

En la anterior entrega presentamos a PyTorch, un framework de modelado para aprendizaje profundo (deep-learning) que permite utilizar un estilo imperativo de programación. Ahora, **TENSORFLOW EAGER EXECUTION** provee este estilo imperativo en TensorFlow permitiendo la ejecución de sentencias fuera del contexto de una sesión. Esta mejora podría proporcionar la facilidad de depurar y controlar modelos más granular de PyTorch con la mayor popularidad y rendimiento de los modelos de TensorFlow. Esta característica es todavía nueva por lo que estamos ansiosos de ver cómo se desempeñará y será recibida por la comunidad de TensorFlow.

TENSORFLOW LITE es el sucesor designado de TensorFlow Mobile, el cual mencionamos en la edición anterior del Radar. Tal como Mobile, es una solución ligera adaptada y optimizada para dispositivos móviles (Android e iOS). Esperamos que el caso de uso estándar sea el despliegue de modelos pre-entrenados en aplicaciones móviles pero adicionalmente TensorFlow Lite soporta aprendizaje en el dispositivo, lo que abre las puertas para más áreas de aplicación.

Estamos probando **TROPOSPHERE** como una forma de definir infraestructura como código en AWS para nuestros proyectos donde AWS CloudFormation es usado en lugar de Terraform. troposphere es una librería escrita en Python que nos permite escribir código Python para generar descriptores de CloudFormation en formato JSON. Lo que nos gustó de troposphere es que facilita la detección temprana de errores en JSON, mediante la verificación de tipado y con pruebas unitarias, además de la composición de recursos en AWS aplicando el principio de no repetición *(DRY, Don't Repeat Yourself)*.

WebAssembly es un formato de compilación binaria diseñado para ejecutarse en el navegador con velocidades cercanas al código nativo. Incrementa el rango de lenguajes que se puede usar para escribir funcionalidad front-end.

(WebAssembly)

WEBASSEMBLY es un gran paso hacia adelante en las capacidades de un navegador como ambiente de ejecución de código. Es soportado por todos los navegadores más conocidos y cuenta con compatibilidad hacia versiones anteriores, es un formato de compilación binaria diseñado para ejecutarse en el navegador con velocidades cercanas al código nativo. Incrementa el rango de lenguajes que se puede usar para escribir funcionalidad *front-end*, con un enfoque temprano en C, C++ y Rust, y también es un objetivo de compilación LLVM. Cuando se ejecuta en un sandbox, puede interactuar con JavaScript y compartir los mismos permisos y modelos de seguridad. El utilizarlo con el nuevo compilador streaming de Firefox da como resultado una rápida inicialización de página. Aunque aún se encuentra en sus primeros días, el estándar W3C es definitivamente uno por el que se puede empezar a explorar.

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

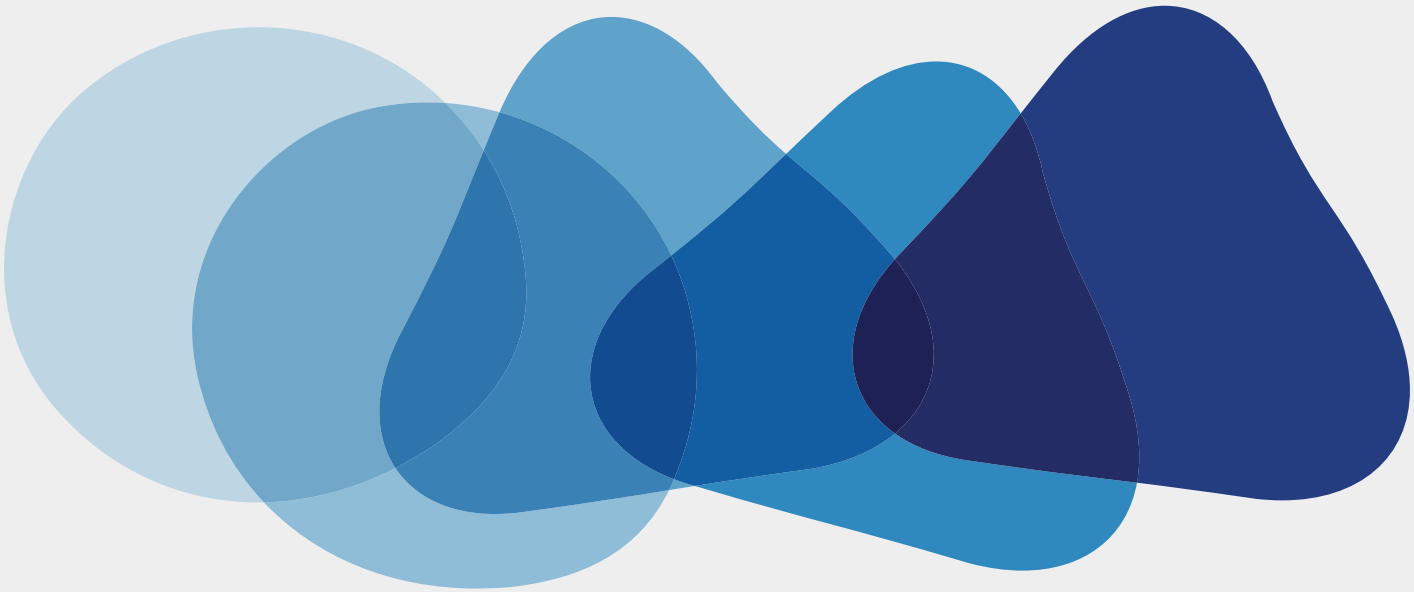
SUSCRÍBETE AHORA

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 más de 4500 personas, incluyendo una división de productos que crea herramientas pioneras de software para equipos. ThoughtWorks cuenta con 42 oficinas en 14 países: Australia, Brasil, Canada, Chile, China, Ecuador, Alemania, India, Italia, Singapur, España, Tailandia, el Reino Unido y los Estados Unidos.



thoughtworks.com/es/radar

#TWTechRadar