Radar Tecnológico

Una guía con opiniones sobre las tecnologías de vanguardia

/thoughtworks

Volume 25

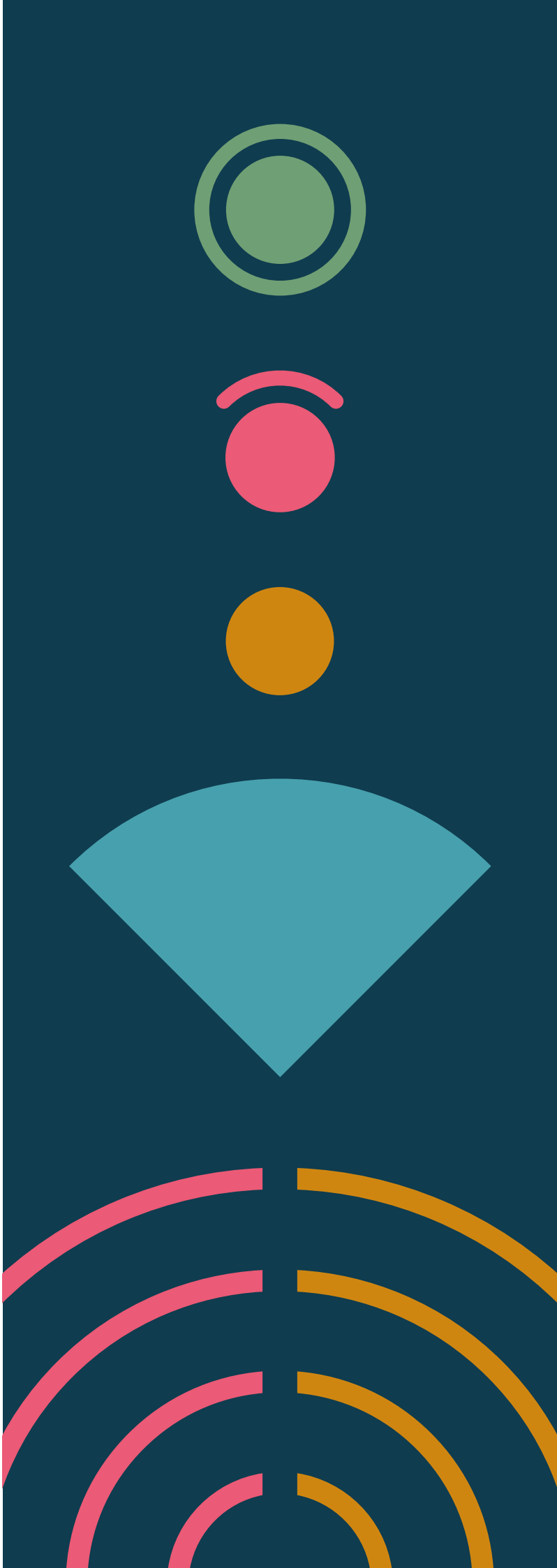
Sobre el Radar

Nuestros Thoughtworkers son personas apasionas por la tecnología. La construimos, la investigamos, la probamos, la hacemos de código abierto, escribimos sobre ella y tratamos de mejorarla constantemente, para todas las personas. Nuestra misión es liderar la excelencia tecnológica y revolucionar las TI. Creamos y compartimos el Radar Tecnológico de Thoughtworks en apoyo de esa misión. La Junta Asesora de Tecnología de Thoughtworks, un grupo de líderes tecnológicos de alto nivel de Thoughtworks, son quienes crean el Radar. Se reúnen periódicamente para debatir la estrategia tecnológica global de Thoughtworks y las tendencias tecnológicas que tienen un impacto significativo en nuestro sector.

El Radar captura los resultados de las discusiones de la Junta Asesora de Tecnología en un formato que provee valor a un amplio rango de personas interesadas, desde personas desarrolladoras hasta CTOs. El contenido pretende ser un resumen consiso.

Te animamos a explorar estas tecnologías. El Radar es de naturaleza gráfica y agrupa los items en técnicas, herramientas, plataformas y lenguajes & frameworks. Cuando items del Radar llegan a aparecer en múltiples cuadrantes, elegimos el que parezca más apropiado. Después agrupamos estos items en cuatro anillos para reflejar nuestra opinión actual sobre ellos.

Para más información sobre el Radar, consulta **thoughtworks.com/es/radar/faq**



Un vistazo al Radar

El Radar se dedica a rastrear cosas interesantes, a las que nos referimos como blips. Organizamos los blips en el Radar utilizando dos elementos de categorización: cuadrantes y anillos. Los cuadrantes representan los diferentes tipos de blips. Los anillos indican en qué fase del ciclo de vida de la adopción creemos que deberían estar.

Un blip es una tecnología o técnica que desempeña un papel en el desarrollo de software. Los blips son cosas que están "en movimiento", es decir, que su posición en el Radar está cambiando, lo que suele indicar que cada vez tenemos más confianza en ellos a medida que avanzan por los anillos.

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

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.

Evaluar: Vale la pena explorar con el objetivo de comprender cómo afectará a su empresa.

Resistir: Proceder con precaución.

Nuevo **Desplazado adentro/afuera** **Ningún cambio**

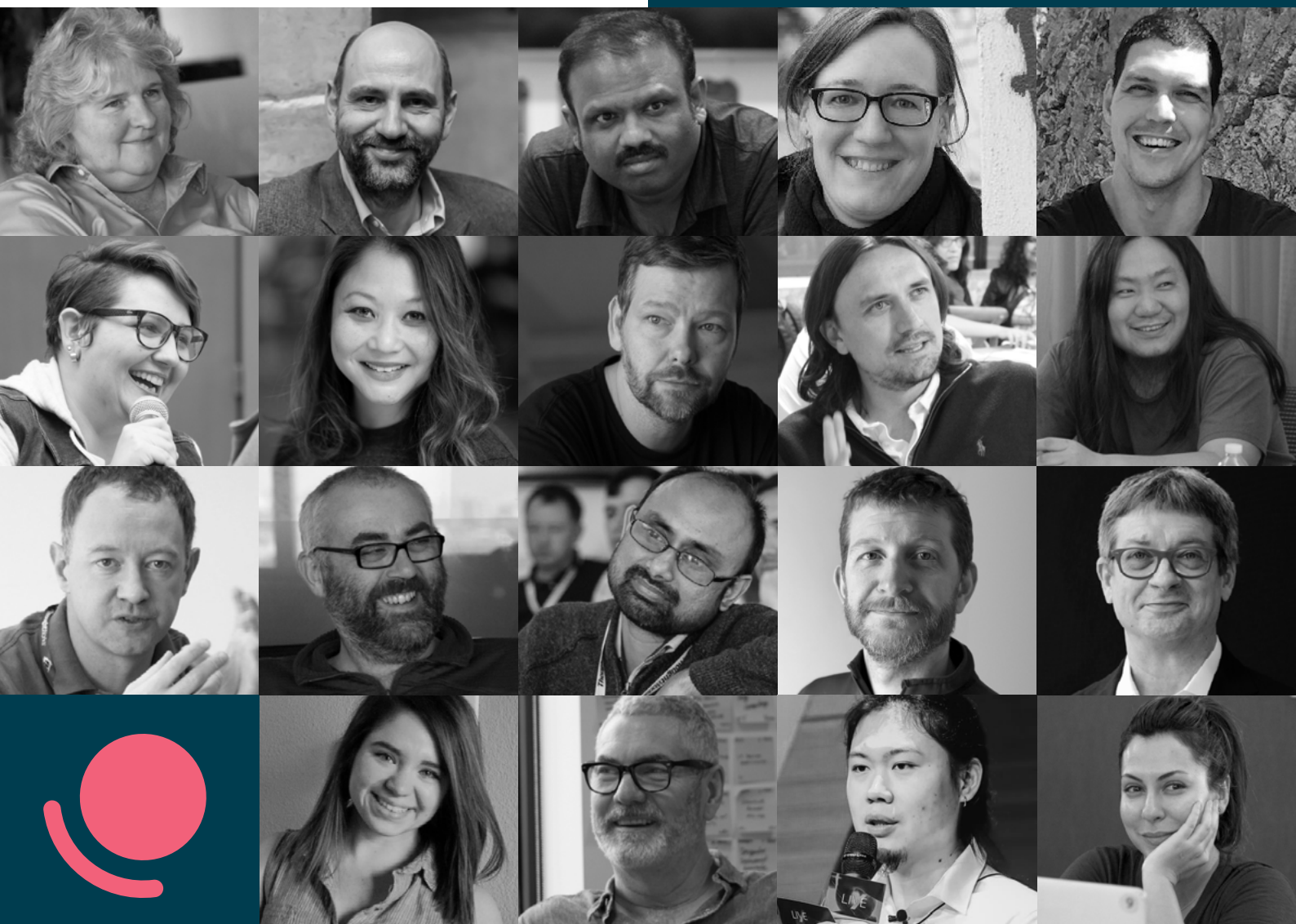
Nuestro Radar está orientado al futuro. Para dar paso a nuevos artículos, desvanecemos los que no se han movido recientemente, lo cual no es un reflejo de su valor, sino de nuestro limitado espacio en el Radar.

Contribuyentes

El Junta Asesora de Tecnología (TAB) es un grupo de 18 tecnologistas senior de Thoughtworks. El TAB se reúne dos veces al año en persona y quincenalmente por teléfono. Su función principal es la de ser un grupo asesor para Rebecca Parsons, CTO de Thoughtworks.

El TAB actúa como un solo individuo que puede examinar los temas que influyen en la tecnología y a tecnologistas de Thoughtworks. Con la actual pandemia mundial, hemos vuelto a crear este volumen del Radar Tecnológico a través de un evento virtual.

Rebecca Parsons (CTO)
Martin Fowler (Chief Scientist)
Bharani Subramaniam
Birgitta Böckeler
Brandon Byars
Camilla Falconi Crispim
Cassie Shum
Erik Doernenburg
Fausto de la Torre
Hao Xu
Ian Cartwright
James Lewis
Lakshminarasimhan Sudarshan
Mike Mason
Neal Ford
Perla Villarreal
Scott Shaw
Shangqi Liu
Zhamak Dehghani



Temas

Adaptando Kafka

Discutimos una serie de temas en esta edición del Radar (algunos de los cuales eventualmente no lograron llegar a la versión final) donde los equipos están empleando herramientas para adaptarse hacía/desde Kafka. Algunas de estas herramientas permiten interfaces más tradicionales para Kafka (como **ksqlDB**, **Confluent Kafka REST Proxy** y Nakadi), mientras otras están diseñadas para proporcionar servicios adicionales como frontends GUI y complementos de orquestación. Sospechamos que parte de la razón subyacente de esta abundancia de herramientas es la complejidad aguda subyacente de algunas de las partes de Kafka combinada con la creciente presencia en organizaciones que necesitan adaptarla a las arquitecturas y procesos existentes. Algunos equipos terminan tratando a **Kafka como un bus de servicio empresarial de nueva generación** - un ejemplo del tema *La engañosa pendiente de la conveniencia* - pero otros equipos usan Kafka para proporcionar acceso universal a los eventos del negocio a medida que ocurren. Estas organizaciones reconocen que a veces es más fácil tener una infraestructura centralizada con adaptación en los bordes y tratan de evitar la expansión con un diseño y gobernanza cuidadosos. En cualquier caso, muestra que Kafka continúa hacia el estatus de estándar de facto para la mensajería de publicación/suscripción asíncrona en volumen.

La engañosa pendiente de la conveniencia

Un antipatrón tan antiguo como el Radar es la tendencia de los equipos para tomar ciertos comportamientos en sus ecosistemas como convenientes (pero inadecuados) puntos de unión que conllevan a un empeoramiento de los mismos y a un aumento de la deuda técnica a largo plazo. Algunos ejemplos son, usar una base de datos como punto de integración, usar **Kafka** como coordinador global, implementar lógica de negocio con código de infraestructura, etc.
El desarrollo moderno de software ofrece a los desarrolladores muchas oportunidades para ocultar este comportamiento, y muchos equipos inexpertos o simplemente desconsiderados se encuentran con estos problemas por no tomar en serio las consecuencias a largo plazo de este tipo de cohesión inapropiada. Estructuras de equipo inapropiadas y otras variaciones de la **Ley de Conway** tampoco ayudan. A medida que los sistemas de software se vuelven más complejos, los equipos de desarrollo deben prestar especial atención a crear y mantener un diseño y arquitectura adecuados y no tomar estas decisiones a conveniencia. A menudo, pensar en la forma de probar un enfoque concreto aleja al equipo de tomar alguna de estas decisiones problemáticas. El software tiende a convertirse en algo complejo cuando se le deja al libre albedrío. Un diseño cuidadoso y, quizás más importante, realizar tareas de gobierno de forma continuada, asegura que la presión en el calendario de entregas o cualquier otra de las muchas fuerzas disruptivas no obligan a los equipos a tomar convenientes (pero inadecuadas) decisiones.

Conway es aún la ley

Muchos arquitectos citan la Ley de Conway, la observación de 1960 de que las estructuras comunicacionales de los equipos influyen en el diseño, para justificar cambios en la organización de un equipo, y descubrimos a través de varios blips nominamos en esta edición que la estructura del equipo de una organización sigue siendo un habilitador clave cuando se maneja bien y un impedimento serio cuando se maneja mal. Los ejemplos que discutimos incluyen la necesidad de pensar en el producto en torno a equipos de plataforma en lugar de tratarlos como tomadores de pedidos; topologías de equipo y el creciente reconocimiento de la carga cognitiva del equipo en relación con la eficacia; y el nuevo framework desarrollado alrededor de la productividad de un programador llamado SPACE. Las organizaciones gastan enormes montos en herramientas, pero muchas obtienen mejores ganancias de productividad al prestar atención a las personas que crean el software y lo que las hace efectivas dentro de una organización en particular.

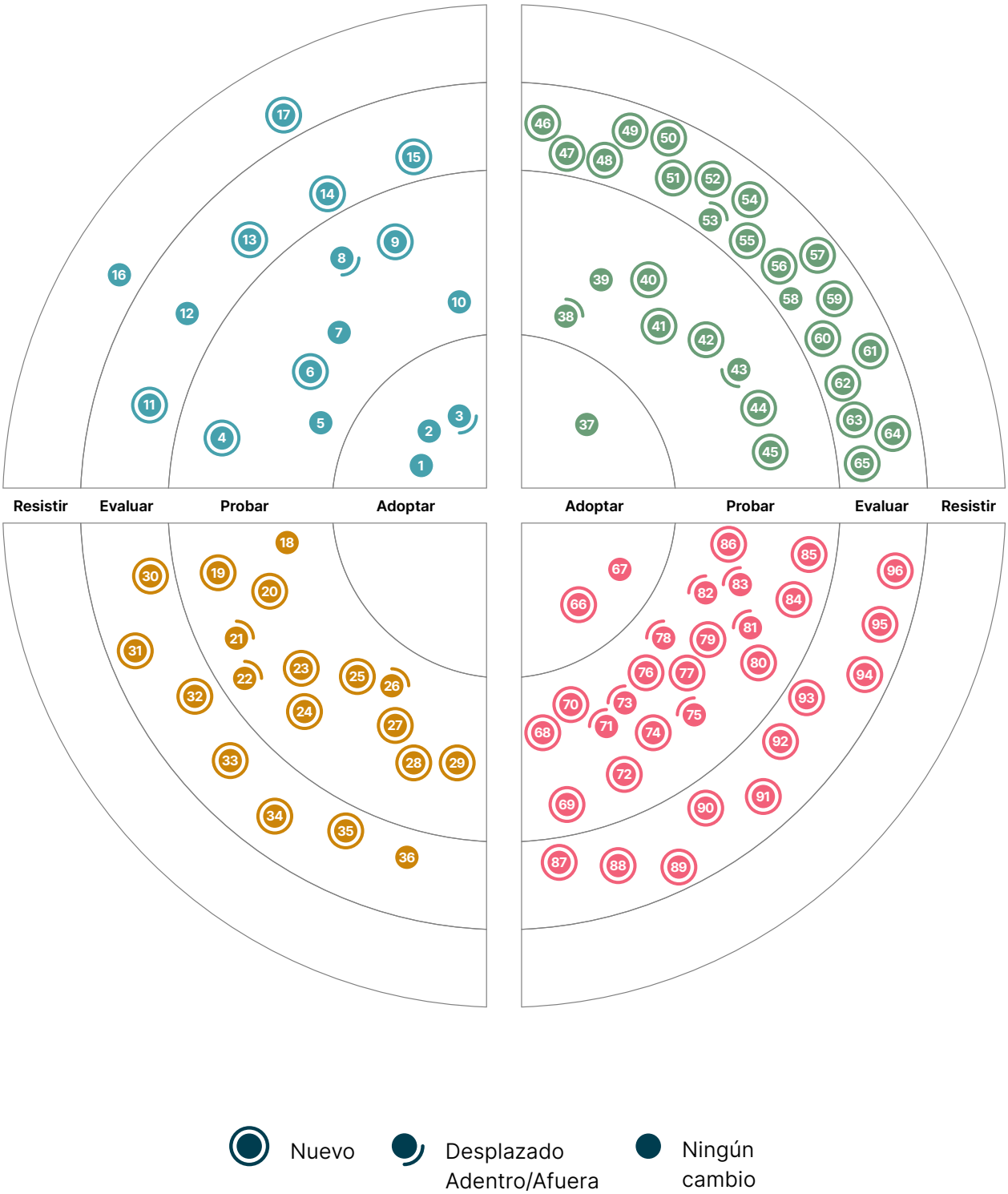
Tecnología inteligente que no deberíamos necesitar

Muchas personas en el mundo del software valoran las soluciones inteligentes para los problemas complejos, pero a menudo esas soluciones inteligentes resultan de una complejidad accidental autoinfligida. Numerosos ejemplos de este fenómeno existen hoy, incluyendo la práctica común pero desafortunada de producir código de orquestación o coordinación en una ubicación inapropiada. Por ejemplo, vemos herramientas de gestión de flujo de trabajo inteligentes como Airflow o Prefect que se utilizan con entusiasmo para gestionar pipelines de datos complejos a través de la orquestación. Encontramos una gran cantidad de herramientas que solucionan los problemas por monorepos, como Nx y muchas más. Los equipos a menudo no se dan cuenta que están duplicando o triplicando la complejidad innecesaria sin dar un paso atrás para mirar el panorama completo y preguntarse si la solución actual es peor que el problema. En lugar de saltar a más tecnología para resolver un problema, los equipos deben realizar un análisis de la causa raíz, abordar la complejidad esencial subyacente y corregir el rumbo. Data mesh es un ejemplo de un enfoque que aborda las suposiciones organizacionales y técnicas subyacentes que han dado lugar a herramientas y pipelines de datos demasiado complejos.

Menos plataformas tecnológicas en el Radar

Hemos identificado un importante descenso en el número de puntos en el Radar relacionados con plataformas en esta edición, cuya causa atribuimos al incremento de la estabilización de ciertos estándares de la industria: la mayoría de empresas han elegido ya sus proveedores en la nube, y prácticamente se ha estandarizado el uso de Kubernetes para la orquestación de contenedores y el uso de Kafka como herramienta de mensajería de alto rendimiento. ¿Significa esto que las plataformas ya no importan? ¿O es que estamos experimentando el equivalente a un ciclo empresarial de períodos de expansión y contracción alternos? - Hemos visto periodos similares de rápida innovación seguidos de estancamiento (lo que Stephen Jay Gould llamó “equilibrio interrumpido”) en tecnologías de bases de datos, por ejemplo. Quizás la industria ha entrado en un período de relativa calma a la par que las organizaciones asimilan los movimientos tectónicos de la nube y esperan la siguiente ola de innovación disruptiva.

El Radar



El Radar

Técnicas

Adoptar

1. Cuatro métricas clave
2. Equipos de productos de ingeniería de plataformas
3. Arquitectura Confianza Cero

Probar

4. Protocolos Bilingües CBOR/JSON
5. Data mesh
6. Documentación viva en sistemas legados
7. Micro frontends para aplicaciones móviles
8. Programación en grupo en remoto
9. Tablero remoto de un equipo
10. Carga cognitiva del equipo

Evaluar

11. Anclajes espaciales para Realidad Aumentada
12. Hotwire
13. Patrón de operación para recursos no clusterizados
14. Acercamiento espontáneo a distancia
15. Lista de Materiales de Software

Resistir

16. La revisión por pares equivale a pull request
17. Producción de datos en entornos de prueba

Plataformas

Adoptar

—

Probar

18. Backstage
19. ClickHouse
20. Kafka REST Proxy de Confluent
21. GitHub Actions
22. K3s
23. Mambu
24. MirrorMaker 2.0
25. Guardián OPA para Kubernetes
26. Pulumi
27. Sealed Secrets
28. Vercel
29. Weights & Biases

Evaluar

30. Azure Cognitive Search
31. Babashka
32. ExternalDNS
33. Konga
34. Milvus 2.0
35. Thought Machine Vault
36. XTDB

Resistir

—

El Radar

Herramientas

Adoptar
37. fastlane

Probar
38. Airflow
39. Batect
40. Berglas
41. Contrast Security
42. Dive
43. Lens
44. Nx
45. Wav2Vec 2.0

Evaluar
46. cert-manager
47. Cloud Carbon Footprint
48. Code With Me
49. Comby
50. Conftest
51. Cosign
52. Crossplane
53. gopass
54. Micoo
55. mob
56. Comandos modernos de Unix
57. Mozilla Sops
58. Operator Framework
59. Pactflow
60. Prefect
61. Proxyman
62. Regula
63. Sourcegraph
64. Telepresence
65. Vite

Resistir
—

Lenguajes y Frameworks

Adoptar
66. Jetpack Compose
67. React Hooks

Probar
68. Arium
69. Chakra UI
70. DoWhy
71. Gatsby.js
72. Jetpack Hilt
73. Kotlin Multiplataforma Móvil
74. lifelines
75. Mock Service Worker
76. NgRx
77. pydantic
78. Quarkus
79. React Native Reanimated 2.0
80. React Query
81. Tailwind CSS
82. TensorFlow Lite
83. Three.js
84. ViewInspector
85. Vowpal Wabbit
86. Zap

Evaluar
87. Headless UI
88. InsightFace
89. Kats
90. ksqlDB
91. Polars
92. PyTorch Geometric
93. Qiankun
94. React Three Fiber
95. Tauri
96. Transloco

Resistir
—

Técnicas

Adoptar

1. Cuatro métricas clave
2. Equipos de productos de ingeniería de plataformas
3. Arquitectura Confianza Cero

Probar

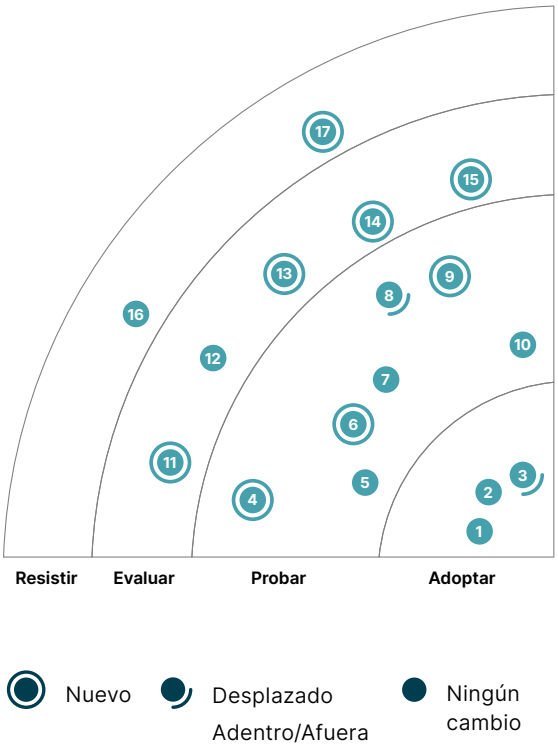
4. Protocolos Bilingües CBOR/JSON
5. Data mesh
6. Documentación viva en sistemas legados
7. Micro frontends para aplicaciones móviles
8. Programación en grupo en remoto
9. Tablero remoto de un equipo
10. Carga cognitiva del equipo

Evaluar

11. Anclajes espaciales para Realidad Aumentada
12. Hotwire
13. Patrón de operación para recursos no clusterizados
14. Acercamiento espontáneo a distancia
15. Lista de Materiales de Software

Resistir

16. La revisión por pares equivale a pull request
17. Producción de datos en entornos de prueba



1.Cuatro métricas clave

Adoptar

Para medir el rendimiento en la entrega de software, cada vez más organizaciones recurren a las cuatro métricas clave definidas por el programa **DORA research**: el tiempo de espera de un cambio, la frecuencia de despliegue, el tiempo medio de restauración (MTTR) y tasa de fallo de cambio. Esta investigación y sus análisis estadísticos han demostrado una clara relación entre el alto rendimiento de entrega y estas métricas, lo cual proporciona un magnífico indicador de cómo un equipo, o incluso toda una organización, lo está haciendo.

Aún somos partidarios de estas métricas, sin embargo hemos aprendido algunas lecciones desde la primera vez que empezamos a monitorearlas. Cada vez vemos más enfoques de medición erróneos en herramientas que ayudan a los equipos a medir estas métricas basándose puramente en sus pipelines de entrega continua (CD). En particular, cuando se trata de métricas de estabilidad (MTTR y tasa de fallo de cambio), únicamente los datos recogidos de las pipelines de entrega continua (CD) no proveen suficiente información para determinar qué es un fallo en el despliegue con un impacto real en el usuario. Las métricas de estabilidad sólo tienen sentido si incluyen datos sobre los incidentes reales que degradan el servicio para los usuarios.

Y como todas las métricas, recomendamos tener siempre presente el objetivo principal que está detrás de la medición y usarlas para reflexionar y aprender. Por ejemplo, antes de estar semanas construyendo sofisticadas herramientas con tableros, considere incluir y revisar regularmente el **DORA quick check** en las retrospectivas del equipo. Esto da al equipo la oportunidad de reflexionar en que **capabilities** podrían trabajar para mejorar sus métricas, lo que puede ser mucho más eficaz que las herramientas que ya están hechas y sobre-detalladas.

2. Equipos de productos de ingeniería de plataformas

Adoptar

Seguimos viendo a los equipos de producto de ingeniería de plataformas como una opción sensata, con la idea principal de que no son más que otro **equipo de producto**, aunque centrado en los clientes internos de la plataforma. Por lo tanto, es fundamental tener clientes y productos claramente definidos y utilizar las mismas disciplinas de ingeniería y formas de trabajo que cualquier otro equipo de producto (centrado en el exterior); los equipos de plataforma no son especiales en este sentido. Advertimos que no hay que limitarse a cambiar el nombre de los equipos internos existentes por el de "equipos de plataforma" sin cambiar los métodos de trabajo ni las estructuras organizativas. Seguimos siendo grandes seguidores del uso de los conceptos de **Team Topologies** cuando pensamos en la mejor manera de organizar los equipos de plataforma. Consideramos que los equipos de productos de ingeniería de plataformas son un enfoque estándar y un importante facilitador de la TI de alto rendimiento.

3. Arquitectura Confianza Cero

Adoptar

Seguimos sabiendo de empresas que descubren que su seguridad ha sido gravemente comprometida debido a una dependencia excesiva del perímetro de red "seguro". Una vez que se viola este perímetro externo, los sistemas internos demuestran estar mal protegidos y los atacantes pueden implementar rápida y fácilmente herramientas de extracción de datos automatizadas y

ataques de ransomware que, con demasiada frecuencia, permanecen sin ser detectados durante largos períodos. Esto nos lleva a recomendar la arquitectura de confianza cero (ZTA) como un estándar sensato.

ZTA es un cambio de paradigma en la arquitectura y la estrategia de seguridad. Se basa en la suposición de que un perímetro de red ya no es representativo de un límite seguro y que no se debe otorgar confianza implícita a los usuarios o servicios basándose únicamente en su ubicación física o de red. La cantidad de recursos, herramientas y plataformas disponibles para implementar aspectos de ZTA sigue creciendo e incluye la imposición de **políticas como código** basadas en los principios de privilegio mínimo y lo más granular posible y en el monitoreo continuo y la mitigación automatizada de amenazas; usando una **malla de servicios** para imponer el control de seguridad entre aplicación-servicio y servicio-servicio; implementando **atestación binaria** para verificar el origen de los binarios; e incluyendo **enclaves seguros** además del cifrado tradicional para hacer cumplir los tres pilares de la seguridad de los datos: en tránsito, en reposo y en memoria. Para más información, consulte la publicación **NIST ZTA** y el artículo de Google sobre **BeyondProd**.

4. Protocolos Bilingües CBOR/JSON

Probar

Aunque ha existido por un tiempo, estamos viendo cada vez más casos donde el usar la especificación **CBOR** para el intercambio de datos, tiene sentido, especialmente en entornos que contienen múltiples tipos de aplicaciones que se comunican entre sí: servicio a servicio, navegador a servicio, etc. Algo que hemos encontrado útil con **Borer** — una implementación a Scala de un codificador/decodificador CBOR — es la capacidad de los clientes para negociar el contenido entre la representación binaria y el antiguo formato JSON. Es muy útil tener una versión de texto visible en un navegador, así como el formato binario conciso. Prevemos que los protocolos bilingües CBOR/JSON cobrarán popularidad con el aumento continuo de IoT, Edge Computing y otras situaciones en las que el entorno es muy restringido.

5. Data mesh

Probar

Cada vez más, vemos una falta de coincidencia entre lo que las organizaciones basadas en datos quieren lograr y lo que permiten las arquitecturas de datos y las estructuras organizativas actuales. Las organizaciones quieren integrar la toma de decisiones basada en datos, machine learning y la analítica en muchos aspectos de sus productos y servicios y en cómo operan internamente; esencialmente, quieren aumentar todos los aspectos de su panorama operativo con inteligencia basada en datos. Sin embargo, todavía nos queda mucho camino por recorrer antes de que podamos integrar datos analíticos, acceder a ellos y cómo se administran en los dominios y operaciones comerciales. Hoy en día, todos los aspectos de la gestión de datos analíticos se externalizan fuera de los dominios comerciales operativos al equipo de datos y a los monolitos de gestión de datos: lagos de datos y almacenes de datos. **Data mesh** es un enfoque sociotécnico descentralizado para eliminar la dicotomía de datos analíticos y operaciones comerciales. Su objetivo es integrar el intercambio y el uso de datos analíticos en cada dominio comercial operativo y cerrar la brecha entre los planos operativo y analítico. Se basa en cuatro principios: propiedad de los datos de dominio, datos como

producto, plataforma de datos de autoservicio y gobernanza federada computacional.

Nuestros equipos han estado implementando la **arquitectura de data mesh**; han creado nuevas abstracciones arquitectónicas, como el cuanto de producto de datos para encapsular código, la política de datos como una unidad autónoma de intercambio de datos analíticos incrustada en dominios operativos; y han creado capacidades de plataforma de datos de autoservicio para administrar el ciclo de vida de los cuantos de productos de datos de manera declarativa, como se describe en ***Data Mesh***. A pesar de nuestros avances técnicos, todavía estamos experimentando fricciones con el uso de las tecnologías existentes en una topología de data mesh, sin mencionar la resistencia de los dominios comerciales a aceptar el uso compartido y el uso de datos como una responsabilidad de primera clase en algunas organizaciones.

6. Documentación viva en sistemas legados

Probar

La **documentation viva**, que proviene desde la comunidad de behavior-driven development (BDD), es frecuentemente considerada un privilegio aplicable para aquellas bases de código que incluyen especificaciones ejecutables. Nosotros encontramos que esta técnica puede ser aplicada también a sistemas legados. La falta de conocimiento de negocio es un obstáculo común que encuentran los equipos cuando están realizando modernización de los sistemas. El código es usualmente la única fuente confiable de verdad debido a las rotaciones de personal y la falta de actualización de la documentación existente. Por lo tanto, es muy importante restablecer la asociación entre la documentación y el código y esparcir el conocimiento del negocio en el equipo cuando se toma a cargo un sistema legado. En práctica, podríamos primero acudir al código y profundizar nuestro conocimiento del negocio simplemente a través de una limpieza o refactorización segura del mismo. Durante el proceso, necesitaremos agregar anotaciones al código para ser capaces de generar automáticamente documentación viva más tarde. Esto es muy diferente a hacer BDD en proyectos green-field, pero es un buen inicio en sistemas legados. Basados en la documentación generada podríamos tratar de convertir algunas de las especificaciones en pruebas de automatización ejecutables de alto nivel. Haz esto iterativamente y eventualmente puedes obtener una documentación viva en sistemas legados que está estrechamente asociada con el código y es parcialmente ejecutable.

7. Micro frontends para aplicaciones móviles

Probar

Desde su introducción en el Radar de 2016, hemos visto una amplia adopción de los **micro frontends** para interfaces de usuario (UIs) web. Sin embargo, recientemente, hemos visto proyectos que extienden este estilo de arquitectura para incluir también micro frontends para aplicaciones móviles. Cuando la aplicación se vuelve suficientemente grande y compleja, es necesario distribuir su desarrollo entre múltiples equipos. Esto presenta el reto de mantener la autonomía de dichos equipos, mientras se integra su trabajo en una única aplicación. Algunos equipos escriben sus propios frameworks para habilitar este estilo de desarrollo, y, en el pasado, hemos mencionado a **Atlas and Beehive** como posibles maneras de simplificar el problema de integrar el desarrollo de apps en múltiples equipos. Más recientemente, hemos visto a equipos utilizando **React Native** para lograr lo mismo. Cada micro frontend de React Native se mantiene en su propio repositorio, donde puede ser compilado, probado y desplegado por separado. El equipo responsable de la aplicación completa

puede después consolidar todos estos micro frontends, desarrollados por diferentes equipos, en una sola aplicación terminada.

8. Programación en grupo en remoto

Probar

Seguimos viendo a muchos equipos trabajando y colaborando de forma remota; para estos equipos la programación en grupo es una técnica que merece la pena probar. La programación en grupo en remoto permite a equipos colaborar rápidamente en un problema o pieza de código sin la limitación física de poder acomodar solamente a cierto número de personas alrededor de una estación de trabajo en pareja. Los equipos pueden colaborar de manera rápida en un problema o pieza de código usando sus herramientas de videoconferencia sin tener que conectar una gran pantalla, reservar una sala de reuniones física o encontrar una pizarra blanca.

9. Tablero remoto de un equipo

Probar

Con el creciente uso de equipos remotos distribuidos, una de las cosas que escuchamos es que las personas echan de menos el tablero físico del equipo. Este es un lugar único en el que se pueden mostrar todas las tarjetas de historias y tareas, con su correspondiente estado y progreso, que actúa como un irradiador de información y punto de encuentro para el equipo. A menudo el tablero era un punto de integración con los datos reales almacenados en varios sistemas diferentes. A medida que los equipos se han vuelto remotos, han tenido que volver a buscar en cada uno de los sistemas de origen, y lograr una mirada “de un vistazo” de un proyecto se ha vuelto muy difícil. Un tablero remoto de un equipo es una técnica simple para reintroducir el tablero del equipo virtualmente. Si bien puede haber un costo extra para mantenerlo actualizado, creemos que los beneficios para el equipo valen la pena. Para algunos equipos, la actualización del tablero físico formaba parte de las “ceremonias” diarias que el equipo realizaba en conjunto, y lo mismo se puede hacer con un tablero remoto.

10. Carga cognitiva del equipo

Probar

La arquitectura de un sistema imita la estructura organizativa y su comunicación. No es una gran novedad que debamos ser intencionales sobre cómo interactúan los equipos - véase, por ejemplo, **Inverse Conway Maneuver**. La interacción de los equipos es una de las variables que determinan la rapidez y la facilidad con la que los equipos pueden aportar valor a sus clientes. Nos alegramos de encontrar una forma de medir estas interacciones; utilizamos la **evaluación** del autor de **Team Topologies**, brinda un mejor entendimiento de lo fácil o difícil que les resulta a los equipos construir, probar y mantener sus servicios. Al medir la carga cognitiva del equipo , podemos asesorar de mejor manera a nuestros clientes sobre cómo cambiar la estructura de sus equipos y hacer evolucionar sus interacciones.

11. Anclajes espaciales para Realidad Aumentada

Evaluar

Muchas aplicaciones de realidad aumentada (RA) necesitan conocer la ubicación y orientación del dispositivo del usuario. Por defecto, se utilizan soluciones basadas en GPS, pero también vale la pena considerar los anclajes espaciales , una nueva técnica para resolver este requisito. Los anclajes espaciales funcionan con la imagen grabada por la cámara del dispositivo, utilizando las

características de la imagen y su posición relativa en el espacio 3D para reconocer una ubicación del mundo real. Para esta ubicación, se crea un ancla correspondiente en el espacio RA. Aunque los anclajes espaciales no pueden reemplazar todos los anclajes basados en marcadores y GPS, brindan más precisión que la mayoría de las soluciones basadas en GPS y son más resistentes a diferentes ángulos de visión que los anclajes basados en marcadores. Nuestra experiencia actualmente se limita a Cloud Anchors for Android, de Google, que funcionó bien para nosotros. De forma algo inusual, Google también ofrece Cloud Anchors for iOS y con Azure Spatial Anchors Microsoft soporta aún más plataformas.

12. Hotwire

Evaluar

Después del exitoso lanzamiento de su aplicación de correo HEY como una aplicación del lado del servidor, Basecamp ha reportado estar migrando su producto insignia, Basecamp 3 a Hotwire este verano. Viendo cómo las organizaciones cada vez más seleccionan por defecto hacer Aplicaciones de Página Única para sus nuevos desarrollos web, nosotros nos emocionados por ver a Hotwire nadando contra corriente. A diferencia de las Aplicaciones de Página Única, las aplicaciones Hotwire conservan la mayoría de la lógica y navegación en el servidor, dependiendo de una mínima cantidad de Javascript en el navegador. Hotwire modulariza las páginas HTML dentro de un conjunto de componentes (llamados Turbo Frames) que pueden ser cargados de forma perezosa, provee contextos independientes y envía actualizaciones de HTML a esos contextos basándose en acciones del usuario. Las Aplicaciones de Página Única ofrecen una innegable capacidad de respuesta al usuario, pero la simpleza de la programación web tradicional del lado del servidor combinada con las herramientas de los navegadores modernos proveen una alternativa refrescante en cuanto al balance entre la efectividad del desarrollador y la capacidad de respuesta al usuario.

13. Patrón de operación para recursos no clusterizados

Evaluar

Estamos viendo un incremento en el uso del patrón de Kubernetes Operator para propósitos distintos a la gestión de aplicaciones desplegadas en el cluster. El uso del patrón de operación para recursos no clusterizados aprovecha las definiciones de recursos personalizadas y el mecanismo de programación basado en eventos implementado en el plano de control de Kubernetes para administrar actividades relacionadas aún fuera del clúster. Esta técnica se construye sobre la idea de Kube-managed cloud services y se extiende a otras actividades, como despliegue continuo o reacción a los cambios en repositorios externos. Una ventaja de esta técnica sobre una herramienta específicamente construida es que abre una amplia gama de herramientas que o bien vienen con Kubernetes o son parte del ecosistema más amplio. Puede usar comandos como **`diff`**, **`dry-run`** o **`apply`** para interactuar con los recursos personalizados del operador. El mecanismo de programación Kube hace el desarrollo más fácil al eliminar la necesidad de orquestar actividades en el orden correcto. Las herramientas de código abierto como Crossplane, Flux y ArgoCD toman ventaja sobre esta técnica y esperamos ver más emerger con el tiempo.

14. Acercamiento espontáneo a distancia

Evaluar

Estamos viendo una innovación continua en las herramientas de colaboración remota. La nueva función Huddles de Slack ofrece una experiencia similar a la de Discord, con llamadas de audio persistentes en las que los usuarios pueden entrar y salir en cualquier momento. Gather ofrece una forma creativa de emular una oficina virtual con avatares y vídeo. Los IDEs ofrecen funciones de colaboración directa para el emparejamiento y la depuración: ya hemos hecho saltar a Visual

Studio Live Share y ha incluido JetBrains Code With Me a la lista en esta edición. A medida que las herramientas siguen evolucionando en cuanto a modalidades de colaboración, además de las videoconferencias, cada vez vemos más equipos que participan en remote spontaneous huddling , recreando la espontaneidad de las conversaciones informales por encima de la intencionalidad de programar una reunión de Zoom o Microsoft Teams. No esperamos recrear por completo la complejidad de la comunicación cara a cara a través de las herramientas digitales, pero sí vemos una mejora en la eficacia de los equipos remotos al ofrecerles múltiples canales de colaboración en lugar de depender de una cadena de herramientas para todo.

15. Lista de Materiales de Software

Evaluar

En mayo del 2021, la Casa Blanca de los Estados Unidos publicó la Orden Ejecutiva sobre la Mejora de la Ciberseguridad de la Nación. El documento propone varios mandatos técnicos que se relacionan a ítems que hemos presentado en Radares anteriores, como es la Arquitectura de Confianza Cero y el Escaneo Automatizado de Cumplimiento usando la política de seguridad como código. Gran parte del documento está dedicado a mejorar la seguridad del software de la cadena de suministro. Un ítem en particular que llamó nuestra atención fue el requerimiento de que el software del gobierno debería contener un lista de materiales de software (SBOM) legible por máquina, definido como "un registro formal conteniendo los detalles y las relaciones de la cadena de suministro de varios componentes utilizados en la construcción de software." En otras palabras, debería detallar no solamente los componentes enviados sino también las herramientas y marcos utilizados para entregar el software. Esta orden ejecutiva tiene el potencial de marcar el comienzo de una nueva era de transparencia y apertura en el desarrollo de software. Esto indudablemente tendrá un impacto en aquellos de nosotros que producimos software como una forma de vida. Muchos, sino todos los productos de software producidos hoy contienen componentes open source o los utilizan en el proceso de desarrollo. A menudo, el consumidor no tiene forma de saber cuál versión de cuál paquete podría tener un impacto en la seguridad de su producto. En cambio, ellos deben confiar en las alertas de seguridad y parches proporcionados por el proveedor minorista. Esta orden ejecutiva asegurará que una descripción explícita de todos los componentes se ponga a disposición del consumidor, empoderándolos para implementar sus propios controles de seguridad, y como el SBOM es legible por máquina, esos controles pueden ser automatizados. Sentimos que este movimiento también representa un cambio hacia adoptar software open source y prácticamente considerando ambos, los riesgos y los beneficios de seguridad que este provee.

16. La revisión por pares equivale a pull request

Resistir

Algunas organizaciones parecen pensar que la revisión por pares equivale a pull request; han adoptado el punto de vista de que la única manera de lograr una revisión por pares del código es a través de un pull request. Hemos visto que este enfoque crea importantes cuellos de botella en el equipo, así como también degrada significativamente la calidad de la retroalimentación en los comentarios, ya que los revisores al verse sobrecargados comienzan simplemente a rechazar las solicitudes. Aunque se podría argumentar que esta es una forma de demostrar el "cumplimiento normativo" de la revisión del código, a uno de nuestros clientes se le dijo que esto no era válido, ya que no había pruebas de que el código fuera realmente leído por alguien antes de su aceptación. Los pull requests son sólo una forma de gestionar el flujo de trabajo de la revisión de código; instamos a la gente a considerar otros enfoques, especialmente cuando hay una necesidad de entrenar y pasar la retroalimentación cuidadosamente.

17. Producción de datos en entornos de prueba

Resistir

Seguimos percibiendo a la producción de datos en entornos de prueba como un área de preocupación. En primer lugar, muchos ejemplos de esto han resultado en daños a la reputación, por ejemplo, cuando se ha enviado una alerta incorrecta desde un sistema de prueba a toda una población de clientes. En segundo lugar, el nivel de seguridad, específicamente en torno a la protección de datos privados, tiende a ser menor para los sistemas de prueba. No tiene mucho sentido tener controles elaborados sobre el acceso a los datos de producción si esos datos se copian en una base de datos de prueba a la que pueden acceder todos los desarrolladores y QA. Aunque puede ocultar los datos, esto tiende a aplicarse sólo a campos específicos, por ejemplo, números de tarjetas de crédito. Por último, copiar datos de producción en sistemas de prueba puede infringir las leyes de privacidad, por ejemplo, cuando los sistemas de prueba se alojan o se accede a ellos desde un país o región diferente. Este último escenario es especialmente problemático con implementaciones complejas en la nube. Los datos falsos son un enfoque más seguro y existen herramientas para ayudar en su creación. Reconocemos que existen razones para copiar elementos

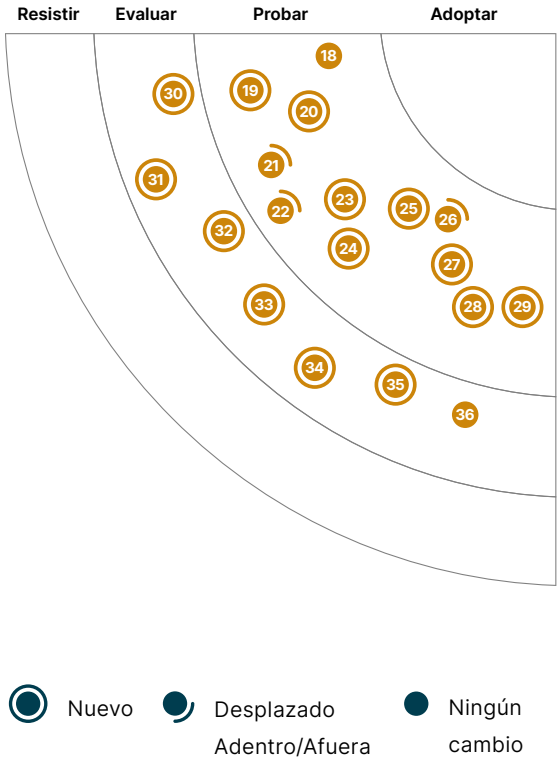
Plataformas

Adoptar
—

Probar
18. Backstage
19. ClickHouse
20. Kafka REST Proxy de Confluent
21. GitHub Actions
22. K3s
23. Mambu
24. MirrorMaker 2.0
25. Guardián OPA para Kubernetes
26. Pulumi
27. Sealed Secrets
28. Vercel
29. Weights & Biases

Evaluar
30. Azure Cognitive Search
31. Babashka
32. ExternalDNS
33. Konga
34. Milvus 2.0
35. Thought Machine Vault
36. XTDB

Resistir
—



18. Backstage

Probar

A medida que aumenta el interés por mejorar la experiencia y la eficiencia de los desarrolladores en las organizaciones, estamos viendo cómo aumenta la popularidad de Backstage, junto con la adopción de portales para desarrolladores. Estas organizaciones buscan apoyar y optimizar sus entornos de desarrollo. A medida que aumenta el número de herramientas y tecnologías, se hace cada vez más importante algún tipo de estandarización para la coherencia, de modo que los desarrolladores puedan centrarse en la innovación y el desarrollo de productos, en lugar de verse atascados con la reinvención de la rueda. Backstage es una plataforma de portal para desarrolladores de código abierto creada por Spotify. Se basa en plantillas de software, herramientas de infraestructura unificadas y documentación técnica coherente y centralizada. La arquitectura de plugins permite la extensibilidad y adaptabilidad al ecosistema de infraestructura de una organización. Estaremos atentos al nuevo Backstage Service Catalog, actualmente en fase alfa, que hace un seguimiento de la propiedad y los metadatos de todo el software del ecosistema de una organización.

19. ClickHouse

Probar

ClickHouse es una base de datos orientada a columnas para procesamiento analítico (OLAP) de código abierto para analíticas en tiempo real. Empezó como un proyecto experimental en el año 2009 y desde entonces ha madurado a una base de datos de análisis de alto rendimiento y con escalamiento lineal. Su eficiente motor para procesamiento de consultas, sumado a su compresión de datos lo hace apropiado para correr consultas interactivas sin necesidad de pre-agregación. Hemos usado ClickHouse y estamos bastante impresionados con su desempeño.

20. Kafka REST Proxy de Confluent

Probar

Kafka es una herramienta comúnmente usada en arquitecturas controladas por eventos, pero adaptarlo a entornos heredados introduce un error de impedancia. En algunos casos, hemos tenido éxito al minimizar la complejidad heredada usando Kafka REST Proxy de Confluent. Este proxy permite a los desarrolladores acceder a Kafka a través de una interfaz HTTP, que es útil en entornos que dificultan el uso del protocolo nativo de Kafka. Por ejemplo, pudimos consumir eventos emitidos a través de SAP simplemente haciendo que el equipo de SAP invocara un comando HTTP POST a través de una llamada de función remota de SAP preconfigurada, evitando la necesidad de poner en marcha una abstracción de Java alrededor de SAP (y un equipo para administrarla). El proxy tiene muchas funciones aunque, al igual que con cualquier herramienta adaptadora de este tipo, recomendamos precaución y una visión clara de los problemas involucrados. Creemos que el proxy es valioso cuando permite que los productores heredados envíen eventos, pero deberíamos ser cautelosos al crear consumidores de eventos a través del proxy a medida que la abstracción se vuelve más compleja. El proxy no cambia el hecho de que los consumidores de Kafka tienen estado, lo que significa que las instancias de consumidor creadas a través de la API REST están vinculadas a un proxy específico, y la necesidad de realizar una llamada HTTP para consumir mensajes de un flujo cambia la semántica estándar de evento de Kafka.

21. GitHub Actions

Probar

A pesar de nuestras advertencias la última vez que lo cubrimos, hemos visto entusiasmo continuo por GitHub Actions. Lo que dijimos antes sigue siendo cierto: GitHub Actions aún no es un reemplazo completamente establecido para flujos de CI/CD complejos. La herramienta no puede, por ejemplo: re-disparar una tarea simple de un flujo, llamar a otras acciones dentro de una acción compuesta,

o soportar una librería compartida. Adicionalmente, mientras que el ecosistema en el GitHub Marketplace ofrece ventajas obvias, entregarle acceso a tus build pipelines a GitHub Actions de terceros, te expone al riesgo de compartir secretos de formas inseguras (recomendamos seguir estos consejos de GitHub sobre security hardening). A pesar de estas preocupaciones, la conveniencia de poder crear los flujos de trabajo directamente en GitHub junto al código fuente, es una opción convincente para algunos equipos, y además, act te permite correr GitHub Actions localmente. Como siempre, recomendamos un análisis imparcial de las desventajas, sin embargo, algunos de nuestros equipos están contentos con la simplicidad de GitHub Actions.

22. K3s

Probar

K3s es una distribución liviana de Kubernetes construida para Internet de las Cosas (IoT) y edge computing. Se obtienen los beneficios de un Kubernetes totalmente compatible pero con una sobrecarga operativa reducida. Sus mejoras incluyen backends de almacenamiento liviano (sqlite3 por defecto en lugar de etcd), un paquete binario con dependencias mínimas del sistema operativo y huella de memoria reducida, todo esto hace que los K3 sean adecuados para entornos con recursos limitados. Hemos utilizado K3 en máquinas para punto de venta y estamos muy contentos con nuestra decisión.

23. Mambu

Probar

Mambu es una plataforma bancaria Saas en la nube. Empodera a los clientes para construir y cambiar sus productos bancarios y de crédito de manera fácil y flexible. A diferencia de otras plataformas bancarias out-of-box que solo se pueden adaptar con integración de código fuente, Mambu ha sido diseñado para ofertas financieras en constante cambio. Viene con un flujo de trabajo automatizado, al mismo tiempo que proporciona un acercamiento basado en APIs, para personalizar la lógica del negocio, los procesos y las integraciones. Actualmente tenemos algunos proyectos utilizando Mambu. Con su escalabilidad basada en la nube y capacidades altamente personalizables, se está convirtiendo en uno de los sistemas bancarios fundamentales cuando se trata de desarrollo de productos financieros.

24. MirrorMaker 2.0

Probar

MirrorMaker 2.0 (también conocido como MM2), desarrollado utilizando la herramienta Kafka Connect, resuelve varias deficiencias de herramientas de Kafka con enfoques en replicación anteriores. Es capaz de geo-replicar con éxito datos de tópicos y metadatos entre clústeres, incluyendo offsets, grupos de consumidores y líneas de comando de autorización (ACL por sus siglas en inglés). MM2 conserva las particiones originales y detecta nuevos tópicos y particiones. Apreciamos la habilidad de organizar la migración de un clúster a lo largo del tiempo, un enfoque que puede ser útil al migrar de un clúster alojado localmente a uno alojado en la nube. Después de sincronizar los tópicos y grupos de consumidores, en primer lugar, migramos los clientes hacia la nueva ubicación del clúster, luego migramos los productores a la nueva ubicación y finalmente deshabilitamos MM2 y retiramos el antiguo clúster. También hemos visto a MM2 ser utilizado en escenarios de alta disponibilidad y recuperación de desastres.

25. Guardián OPA para Kubernetes

Probar

Guardián OPA para Kubernetes es un webhook de admisión personalizable para Kubernetes que hace cumplir las políticas ejecutadas por el Open Policy Agent (OPA). Estamos utilizando esta extensión de la plataforma Kubernetes para agregar una capa de seguridad a los clústeres,

proporcionando mecanismos de gobierno automatizados que garantizan que las aplicaciones cumplen con las políticas definidas. A nuestros equipos les gusta por su capacidad de personalización; el uso de CustomResourceDefinitions (CRD) nos permite definir ConstraintTemplates y Constraints que hacen que definir reglas y los objetos (por ejemplo, despliegues, trabajos, trabajos cron) y espacios de nombres en evaluación sea una tarea fácil.

26. Pulumi

Probar

Hemos estado viendo un aumento en los equipos que utilizan **Pulumi** en varias organizaciones. Pulumi llena un vacío en el mundo de la codificación de la infraestructura donde **Terraform** se mantiene firme. Mientras que Terraform es un soporte probado y verdadero, su naturaleza declarativa sufre de facilidades de abstracción inadecuadas y una testabilidad limitada. Terraform es adecuado cuando la infraestructura es totalmente estática, pero las definiciones dinámicas de la infraestructura requieren un verdadero lenguaje de programación. Pulumi se distingue por permitir que las configuraciones se escriban en **TypeScript**/JavaScript, **Python** y **Go** — sin necesidad de lenguaje de marcas o plantillas. Pulumi está estrechamente centrado en las arquitecturas nativas de la nube - incluyendo contenedores, funciones sin servidor y servicios de datos — y proporciona un buen soporte para **Kubernetes**. Recientemente, **AWS CDK** ha montado un desafío, pero Pulumi sigue siendo la única herramienta neutral de la nube en esta área.

27. Sealed Secrets

Probar

Kubernetes soporta de manera nativa un objeto clave-valor conocido como secreto. Sin embargo, por defecto, los secretos de Kubernetes no son realmente secretos. Son manejados de forma separada de otros datos clave-valor, de modo que se les pueda aplicar precauciones o controles de acceso de forma separada. Existe soporte para encriptar secretos antes de que sean almacenados en **etcd**, pero los secretos comienzan su existencia como campos de texto plano en ficheros de configuración. **Sealed Secrets** es una combinación de operador y utilidad de línea de comando que usa claves asimétricas para encriptar secretos de modo que solo puedan ser desencriptados por el controlador que hay en el cluster. Este proceso asegura que los secretos no se verán comprometidos mientras se encuentran en los ficheros de configuración que define un despliegue de Kubernetes. Una vez encriptados, estos ficheros pueden ser compartidos o almacenados de forma segura junto a otros artefactos de despliegue.

28. Vercel

Probar

Desde que evaluamos por primera vez **JAMstack**, hemos visto cada vez más aplicaciones web de este estilo. Sin embargo, cuando la infraestructura para construir sitios web con dinámicas tradicionales y servicios back-end es demasiado pesada para JAMstack, nuestros equipos eligen **Vercel**. Vercel es una plataforma en la nube para el alojamiento de sitios estáticos. Más importante aún, provee un flujo de trabajo impecable para el desarrollo, con vista previa y envío a los sitios de JAMstack. La configuración para el despliegue es muy simple. Por estar integrado con GitHub, cada commit o pull request puede desencadenar el despliegue de un nuevo sitio web que contiene una URL para vista previa, lo que acelera enormemente el feedback del desarrollo. Vercel también utiliza CDN para escalar y acelerar los sitios de producción. Vale la pena mencionar que el equipo detrás de Vercel esta también brindando soporte a otro popular framework, **Next.js**.

29. Weights & Biases

Probar

Weights & Biases es una plataforma de machine learning (ML) para construir modelos más rápidamente a través del seguimiento de experimentos, el control de versiones del conjunto de datos, la visualización del rendimiento del modelo y la gestión del modelo. Se puede integrar con el código de ML existente y obtener rápidamente métricas en vivo, logs de terminal y estadísticas del sistema que se transmiten al dashboard para su posterior análisis. Nuestros equipos han utilizado Weights & Biases, y nos gusta su enfoque colaborativo hacia la construcción de modelos.

30. Azure Cognitive Search

Evaluar

Azure Cognitive Search provee la búsqueda como servicio para aplicaciones que requieren búsqueda de texto sobre contenido heterogéneo. Provee APIs basadas en pull o push para cargar e indexar imágenes, texto desestructurado o contenido de documentos estructurados, con limitaciones en tipos de fuente de datos basadas en extracción admitidos. Provee APIs sobre REST y .NET SDK para ejecutar consultas de búsqueda, ya sea utilizando un lenguaje de consulta simple o consultas más poderosa Apache Lucene con consultas de ámbito de campo, búsqueda difusa, búsqueda con comodines para infijos y sufijos, búsqueda de expresiones regulares, entre otras características. Nosotros hemos usado exitosamente Azure Cognitive Search junto con otros servicios de Azure, incluyendo la búsqueda de contenido cargado desde Cosmos DB.

31. Babashka

Evaluar

Incluso hoy en día, considerando todas las herramientas de desarrollo e infraestructura a nuestra disposición, solemos llegar a un punto en donde necesitamos un script para unir varias cosas juntas o para automatizar una tarea recurrente. Actualmente los favoritos para escribir estos scripts son los lenguajes bash y Python, pero estamos felices de comentar que hay una nueva y excitante opción: Clojure. Esto fue posible con Babashka, un Clojure completamente implementado con GraalVM. Babashka utiliza bibliotecas que cubren la mayoría de los casos de uso por los cuales se utilizaría una herramienta de scripting, y también es posible añadir más bibliotecas. El uso de GraalVM aporta tiempos de inicio dentro del mismo rango de las herramientas nativas, y además hace que Babashka sea una de las pocas opciones para un entorno de scripting multiproceso, en esos extraños casos en los que es necesario.

32. ExternalDNS

Evaluar

ExternalDNS sincroniza los ingresos y servicios de Kubernetes con los proveedores de DNS externos, llenando un hueco que anteriormente llenaban kops dns-controller, Zalando's Mate or route53-kubernetes - estos dos últimos han sido descartados en favor de ExternalDNS. La herramienta hace que los recursos internos de Kubernetes sean descubiertos a través de servidores DNS públicos, eliminando un paso a veces manual, para actualizar los registros DNS cuando la dirección IP de un host o servicio de entrada cambia. Es compatible con una enorme lista de proveedores de servicios DNS, y se están añadiendo más a través del apoyo de la comunidad. Como dice el viejo chiste, it's always DNS.

33. Konga

Evaluar

Konga es una UI de código abierto que permite administrar Kong API Gateway, previamente presentada en la sección Probar del Radar. A nuestros equipos les gustó su rápida configuración y la riqueza de características que posee, lo que permite fácilmente probar y experimentar nuevas configuraciones. Además — debido a ser software de código abierto— no nos tenemos que preocupar de los costes de licencia.

34. Milvus 2.0

Evaluar

Milvus 2.0 es una base de datos de vectores, nativa de la nube y de código abierto. Construida para buscar y gestionar vectores embebidos generados mediante modelos de aprendizaje automático y redes neuronales. Soporta diferentes **indices** para aproximar los vecinos más cercanos (ANN) en búsquedas sobre los vectores embebidos de audio, vídeo, imagen o cualquier dato sin estructura. Milvus 2.0 es una base de datos relativamente nueva y recomendamos que se valore su uso si tienen necesidades de búsqueda similares.

35. Thought Machine Vault

Evaluar

Es raro para nosotros presentar en el Radar un software comercial y mucho menos una plataforma bancaria principal. Sin embargo, **Thought Machine Vault** (sin conexión con Thoughtworks) es un ejemplo de un producto en esta clase diseñado para apoyar las buenas prácticas de la ingeniería de software como son el desarrollo impulsado por pruebas, la entrega continua y la infraestructura como código. Los desarrolladores definen los productos bancarios en Vault escribiendo contratos inteligentes en código Python. Esto es claramente diferente del enfoque no-code estándar donde la personalización se realiza mediante interfaces gráficas o archivos de configuración propietarios o ambos. Debido a que los productos se definen en código Python ordinario, los desarrolladores tienen acceso a un rango de herramientas como los frameworks de pruebas y el control de versiones para asegurar que su trabajo es seguro y preciso. Deseamos que más plataformas de servicios financieros sean diseñadas teniendo en cuenta la eficacia del desarrollador.

36. XTDB

Evaluar

XTDB es una base de datos de código abierto con consultas gráficas bitemporales. Admite de forma nativa dos ejes de tiempo para cada registro valid time, para cuando sucede un evento, y transaction time, para cuando un evento es procesado y registrado en la base de datos. La compatibilidad con la bitemporalidad es beneficiosa en numerosos escenarios como: los casos de uso para análisis que ejecutan consultas dependientes del tiempo; en auditorias de cambios históricos en eventos; arquitecturas de datos distribuidos que deben garantizar consultas de puntos en el tiempo de manera consistente a nivel global, tales como **data mesh**; y para preservar la inmutabilidad de datos. XTDB acepta información en forma de documento, y expresada en el formato Extensible Data Notation (EDN), un subconjunto del lenguaje Clojure. XTDB admite consultas en grafos y SQL, además es extensible a través de una capa de REST API y Kafka Connect, entre otros módulos. Estamos entusiasmados de ver un crecimiento en la adopción de XTDB y la adición de características como soporte para transacciones y SQL.

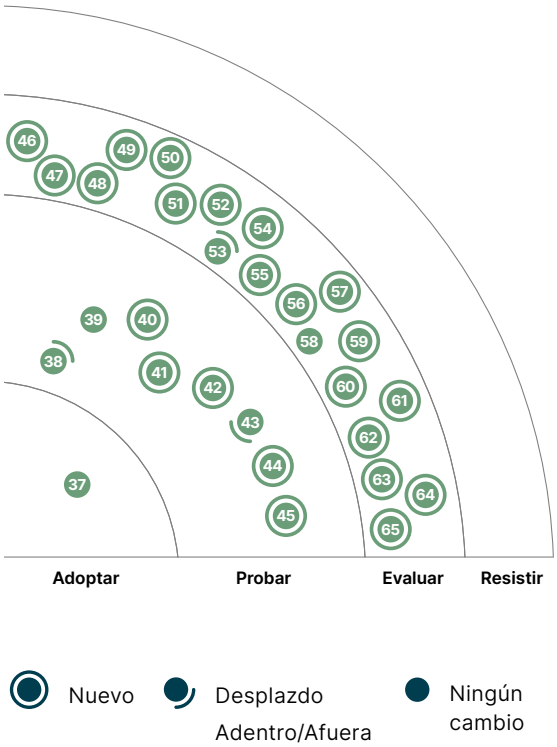
Herramientas

Adoptar
37. fastlane

Probar
38. Airflow
39. Batect
40. Berglas
41. Contrast Security
42. Dive
43. Lens
44. Nx
45. Wav2Vec 2.0

Evaluar
46. cert-manager
47. Cloud Carbon Footprint
48. Code With Me
49. Comby
50. Conftest
51. Cosign
52. Crossplane
53. gopass
54. Micoo
55. mob
56. Comandos modernos de Unix
57. Mozilla Sops
58. Operator Framework
59. Pactflow
60. Prefect
61. Proxyman
62. Regula
63. Sourcegraph
64. Telepresence
65. Vite

Resistir
—



37. fastlane

Adoptar

El lanzamiento de aplicaciones para iOS implica un paso de firma de código. Aunque compatible con las herramientas de Apple, este proceso puede ser engorroso, propenso a errores y lleno de sorpresas. Nos complace informar que **fastlane**, es ya, nuestra herramienta de elección para automatizar el proceso de lanzamiento de aplicaciones móviles, pues proporciona una mejor solución: **match** está integrado en el fluido proceso de fastlane e implementa un **nuevo enfoque** para administrar la firma de código para los equipos. En lugar de almacenar las claves de firma en el keychain del macOS del desarrollador, la estrategia predeterminada; la nueva estrategia gira en torno al almacenamiento de las claves y los certificados en un repositorio de Git. Esto no solo facilita la incorporación de nuevos miembros al equipo y la configuración de nuevas máquinas de desarrollo; en nuestra experiencia, también es el método más fácil de integrar la firma de código en los pipelines de entrega continua.

38. Airflow

Probar

En los años recientes hemos visto el surgimiento de herramientas de manejo de flujos de trabajo genéricas y específicas de dominio. Entre los factores que han impulsado este crecimiento tenemos el incremento en el uso de pipelines de procesamiento de datos y la automatización del proceso de desarrollo del modelo de machine learning (ML). **Airflow** es una de las herramientas tempranas de código abierto de orquestación de tareas que popularizaron la definición de grafos acíclicos dirigidos (DAGs) como código, una mejora con respecto a una configuración de pipeline XML/YAML. Aunque Airflow permanece como una de las herramientas de orquestación más ampliamente utilizadas, les alentamos a evaluar otras herramientas basadas en su situación específica. Por ejemplo, podrían escoger **Prefect**, que soporta tareas dinámicas de procesamiento de datos como preocupación de primera clase con funciones Python genericas como tareas; o **Argo** si prefieres una integración estrecha con Lubernetes; o **Kubeflow** o **MLflow** para flujos de trabajo específicos de ML. Dado el surgimiento de nuevas herramientas, combinado con algunas de las carencias de Airflow (como la falta de soporte nativo para flujos dinámicos y su enfoque centralizado para calendarizar pipelines) no recomendamos Airflow como la herramienta de orquestación por defecto.

Creemos que con el incremento en el uso de streaming en analytics y pipelines de datos, asi como el manejo de datos a través de un **decentralized data mesh**, se reduce la necesidad de herramientas de orquestación para definir y manejar complejos pipelines de procesamiento de datos.

39. Batect

Probar

Batect sigue ganando aceptación entre nuestros desarrolladores y es considerado por muchos como la aproximación por defecto para configurar entornos de prueba y desarrollo local. Esta herramienta de código abierto (que resulta que ha sido desarrollada por un Thoughtworker) facilita crear y compartir entornos de construcción basados en **Docker**. Batect se convierte entonces en el punto de entrada de tu sistema de construcción, reemplazando el omnipresente script de go como la base de una aproximación "**check out and go**" Batect continúa evolucionando en respuesta a la retroalimentación de los desarrolladores y ha añadido recientemente soporte para el BuildKit de Docker y la finalización de la pestaña de shell.

40. Berglas

Probar

Berglas es una herramienta para gestionar secretos en **Google Cloud Platform (GCP)**. En el pasado hemos recomendado **secretos como servicio** como técnica para almacenar y compartir secretos en arquitecturas distribuidas modernas, y GCP ofrece **Secret Manager** para ese fin, y Berglas funciona

bien con Secret Manager. Esto es especialmente útil para aquellos servicios de GCP que aún no tienen integración directa con Secret Manager; la alternativa en estos casos sería escribir código o scripts personalizados. Berglas se presenta como una herramienta de línea de comandos y como una biblioteca, y ambas resultan útiles en casos de uso más allá de los secretos como servicio. El autor de Berglas, que también es el autor original de **HashiCorp Vault**, trabaja ahora en Google; sin embargo, Berglas no es un producto oficial de Google.

41. Contrast Security

Probar

Contrast Security ofrece una plataforma de seguridad con múltiples componentes, incluyendo pruebas de seguridad de aplicaciones estáticas (SAST por sus siglas en inglés), pruebas de seguridad de aplicaciones interactivas (IAST por sus siglas en inglés), escaneo de código abierto y auto-protección de aplicaciones en tiempo de ejecución (RASP por sus siglas en inglés). Existe desde hace algunos años, y lo hemos usado en múltiples proyectos. Una de las cosas que más nos gusta de la plataforma Contrast es su análisis de bibliotecas en tiempo de ejecución; ayuda a identificar bibliotecas que no son usadas, lo que a su vez ayuda a nuestros equipos a priorizar vulnerabilidades y potencialmente eliminar bibliotecas sin uso. Esto es particularmente relevante dado el aumento de importancia que tiene la **securización de la cadena de suministro de software**. También nos gusta particularmente su componente IAST; lo hemos encontrado efectivo en nuestra pipeline de entrega contínua (CD) con una reducción de falsos positivos, y consigue detectar un gran abanico de vulnerabilidades.

42. Dive

Probar

Dive es una herramienta para analizar imágenes de Docker; ayuda a explorar cada capa de la imagen e identificar qué ha cambiado en cada capa. Dive estima la eficiencia de la imagen y el espacio desperdiciado en una imagen y se puede integrar en el proceso de integración continua (IC) para que la construcción, en base al puntaje de eficiencia o la cantidad de espacio desperdiciado, falle. Lo hemos utilizado en algunos proyectos y ha demostrado ser una herramienta útil, especialmente si estamos creando imágenes con tolerancia muy baja para herramientas adicionales o consumo de espacio.

43. Lens

Probar

Nuestros equipos continúan reportando buenos resultados cuando usan **Lens** para visualizar y administrar sus **Kubernetes** clústeres. Lens, catalogado como un “IDE para Kubernetes”, permite interactuar con el clúster sin tener que memorizar comandos o estructuras de archivos de manifiesto. Kubernetes puede ser complejo y entendemos que una herramienta para visualizar las métricas del clúster y las cargas de trabajo implementadas puede ahorrar tiempo y reducir parte del trabajo que implica el mantenimiento de un clúster de Kubernetes. En lugar de ocultar la complejidad detrás de una interfaz simple de apuntar y hacer click, Lens reúne las herramientas que un administrador ejecutaría desde la línea de comandos. Pero ten cuidado al realizar cambios de forma interactiva en un clúster en ejecución a través de cualquier mecanismo. Por lo general, preferimos que los cambios de infraestructura se **implementen en código** para que sean repetibles, comprobables y menos propensos a errores humanos. No obstante, Lens se destaca como una herramienta integral para navegar interactivamente y comprender el estado de su clúster.

44. Nx

Probar

A lo largo de los años, hemos debatido varias veces si incluir los monorrepos en el Radar. Cada vez terminamos concluyendo que el costo-beneficio introducido por los monorrepos requieren una discusión matizada y que la técnica es "demasiado compleja para hacer un blip". Ahora estamos viendo un mayor interés en los monorrepos en la comunidad de JavaScript, por ejemplo, para crear aplicaciones compuestas por micro frontends, como se discute en este episodio de podcast. Que esto sea una buena idea depende mucho de tu situación, y desde luego no queremos dar una recomendación general. Lo que sí queremos comentar es la herramienta. En nuestros equipos vemos un alejamiento de Lerna y una fuerte preferencia por el uso de Nx para la gestión de monorrepos basados en JavaScript.

45. Wav2Vec 2.0

Probar

Wav2Vec 2.0 es un marco de trabajo con aprendizaje auto-supervisado destinado al reconocimiento de voz. Con este marco de trabajo el modelo es entrenado en dos fases. Primeramente, empieza en modo auto-supervisado usando datos sin clasificar e intenta obtener la mejor representación del discurso. Después, usando un ajuste de precisión supervisado, durante el cual datos ya clasificados enseñan al modelo a predecir palabras y fonemas concretos. Hemos usado Wav2Vec y su método es muy poderoso para construir modelos de reconocimiento automático de voz para idiomas regionales con poca disponibilidad de datos clasificados.

46. cert-manager

Evaluar

cert-manager es una herramienta para gestionar sus certificados X.509 dentro de un clúster Kubernetes. Diseña los certificados y emisores como tipos de recursos de primera clase y proporciona certificados como servicio de forma segura a los desarrolladores y aplicaciones que trabajan en el clúster de Kubernetes. Con soporte incorporado para Let's Encrypt, HashiCorp Vault y Venafi, cert-manager es una herramienta interesante para evaluar la gestión de certificados.

47. Cloud Carbon Footprint

Evaluar

Cada vez más, los stakeholders esperan que se tengan en cuenta las repercusiones medioambientales de sus decisiones, como lo demuestra el aumento de la inversión ambiental, social y de gobierno corporativo (ESG) y el activismo de los empleados en torno al cambio climático. La migración a la nube ofrece el potencial para un uso de la energía más eficiente - los proveedores de la nube tienen un tamaño mucho mayor tal que justifica inversiones en energías renovables e I+D - pero la parte negativa para usuarios de la nube al abstraer el software es que ésta abstracción conlleva la pérdida de visibilidad del impacto energético, ya que los centros de datos están ocultos y son financiados por otra compañía. Cloud Carbon Footprint, es una nueva herramienta de código abierto que aprovecha las APIs de la nube para proporcionar visualizaciones de las estimaciones de emisión de carbono sobre AWS, GCP y Azure. Utiliza heurísticas como Cloud Jewels de Etsy para estimar la utilización de energía, y fuentes de datos públicas para convertir el uso de energía en emisiones en función de la intensidad de carbono de la red de energía subyacente a la región de la nube (GCP ya publica estos datos). Los dashboards de la herramienta actúan como radiadores de información, permitiendo a los responsables modificar la configuración para reducir costos y emisiones al mismo tiempo. El vínculo entre las regiones de la nube y la intensidad de carbono de la red subyacente proporciona un empujón para cambiar en favor de fuentes verdes de energía.

48. Code With Me

Evaluar

La herramienta de codificación colaborativa de JetBrains, Code With Me, ha ido aumentando su popularidad a medida que muchos equipos utilizan varias herramientas de JetBrains en un mundo remote-first. Junto con otras herramientas de colaboración remota, como Visual Studio Live Share de VSCode, Code With Me ofrece a los equipos de desarrollo una experiencia mejorada de emparejamiento y colaboración remota. Vale la pena explorar las capacidades de Code With Me para invitar a compañeros de equipo a los proyectos del IDE y colaborar en tiempo real. Sin embargo, hemos visto algunas limitaciones con respecto a la refactorización y algunos problemas en entornos de alta latencia. Seguiremos observando esta herramienta en este espacio.

49. Comby

Evaluar

Esta edición del Radar presenta dos herramientas para buscar y reemplazar código usando una representación de árbol de sintaxis abstracta (AST). Son herramientas similares a jscodeshift pero contienen analizadores para una amplia gama de lenguajes de programación. Aunque comparten algunas similitudes, también difieren en varios aspectos. Una de estas herramientas, Comby, es única por su sencilla interfaz de línea de comandos, diseñada con el mismo espíritu de herramientas de Unix como **awk** y **sed**. Mientras que los comandos de Unix se basan en expresiones regulares que operan con texto coincidente, Comby emplea una sintaxis de patrones que es específica para las construcciones de los lenguajes de programación y analiza el código antes de realizar la búsqueda. Esto ayuda a los desarrolladores a buscar patrones estructurales en grandes bases de código. Como **sed**, Comby puede reemplazar las ocurrencias de los patrones con nuevas estructuras. Esto es útil para automatizar cambios al por mayor en grandes bases de código o para realizar cambios repetitivos en un conjunto de repositorios de microservicios. Dado que estas herramientas son bastante nuevas, esperamos ver una variedad de usos creativos que aún no se han descubierto..

50. Conftest

Evaluar

Conftest es una herramienta para escribir pruebas contra datos de configuración estructurados. Se basa en el lenguaje Rego de Open Policy Agent para escribir pruebas para configuraciones de Kubernetes definiciones de pipelines de Tekton o incluso planes de Terraform. Las configuraciones son una parte crítica de la infraestructura, y le animamos a evaluar Conftest para verificar las suposiciones y obtener una rápida respuesta.

51. Cosign

Evaluar

Cosign es una herramienta de firma y verificación de contenedores. Parte de Sigstore — un proyecto bajo el paraguas de Cloud Native Computing Foundation (CNCF) que busca simplificar la firma de software y la transparencia — Cosign soporta no solo imágenes de Docker y Open Container Initiative (OCI) sino también otros artefactos que se pueden almacenar en un registro de contenedores. Anteriormente hablamos sobre Docker Notary, que también opera en este espacio; Notary v1, sin embargo, tiene algunas desventajas: no tiene registros nativos y necesita un servidor Notary por separado. Cosign evita este problema y almacena las firmas en el registro junto a una imagen. Actualmente se integra con GitHub actions y Kubernetes usando un Webhook con posteriores integraciones en los pipeline. Hemos usado Cosign en algunos de nuestros proyectos y parece bastante prometedor.

52. Crossplane

Evaluar

Crossplane es otra entrada en la clase de herramientas implementadas por el Patrón Operador en

Kubernetes pero con efectos secundarios que van más allá del clúster de Kubernetes. En nuestro último Radar mencionamos Kube-managed cloud services como una técnica, y Crossplane hace justamente eso. La idea es aprovechar el plano de control de Kubernetes para proporcionar los servicios en la nube de los que depende tu despliegue, incluso si no están desplegados en el propio clúster. Algunos ejemplos son instancias de bases de datos administradas, balanceadores de carga o políticas de control de acceso. La herramienta es destacable por dos razones. La primera, demuestra el potente y flexible entorno de ejecución del plano de control subyacente de Kubernetes. No existe un límite real al rango de recursos personalizados soportados. Segundo, Crossplane provee una alternativa a las opciones habituales de uso de Terraform, CDK o Pulumi. Crossplane viene con un conjunto de proveedores predefinidos para los principales servicios en la nube que cubren los aprovisionamientos más comunes. No intenta ser una herramienta de infraestructura como código (IaC) de propósito general, sino más bien un complemento de las cargas de trabajo que se despliegan en Kubernetes. A menudo asociado con la práctica de GitOps, Crossplane se mantiene por sí solo y permite permanecer dentro del ecosistema de Kubernetes cuando sea necesario administrar los recursos externos de la nube. Sin embargo, Crossplane no ayuda con aprovisionamiento a Kubernetes propiamente; necesitará al menos otra herramienta IaC para iniciar el clúster.

53. gopass

Evaluar

gopass es un administrador de contraseñas para equipos, construido sobre GPG y Git. Ha sido desarrollado a partir de pass y agrega varias características, incluyendo búsqueda interactiva y almacenamiento de múltiples contraseñas en un único árbol. Desde que hemos mencionado gopass por primera vez, nuestros equipos lo han utilizado en varios proyectos, a veces extendiéndose más allá de sus límites. Una característica faltante es la posibilidad de eliminar secretos. La capacidad de encontrarlos ya era un problema existente, pero no poder marcar secretos como que ya no se usarán agrava este problema. La preocupación más importante, sin embargo, fue la escala. Cuando se tiene equipos con más de 50 personas que usan el mismo repositorio durante varios años, encontramos que el repositorio podría crecer varios gigabytes de tamaño. Re-encriptar los secretos al incorporar nuevos miembros podría llevar más de media hora. El problema de fondo parece ser que en nuestros equipos hay cambios todo el tiempo: la gente va y viene, los secretos se rotan, la arquitectura evoluciona, se agregan nuevos secretos, y los viejos ya no son necesarios. gopass parece funcionar bien, incluso para un gran número de usuarios, cuando hay menos cambios.

54. Micoo

Evaluar

Micoo es un nuevo contrincante en el abarrotado espacio de herramientas de regresión visual; es una solución de código abierto y autónomo, que proporciona imágenes de Docker para permitir una fácil y rápida configuración de entorno. También proporciona diferentes clientes para Node.js, Java y Python, así como un complemento de Cypress para que pueda integrarse fácilmente con la mayoría de interfaces de usuario UI comunes, marcos de pruebas de automatización o soluciones. Aunque Micoo no proporciona toda la funcionalidad de aquellas que están basadas en SaaS u otro tipo de soluciones comerciales, nuestros equipos lo han estado utilizando ampliamente y han tenido experiencias positivas. Especialmente han señalado que funciona para aplicaciones móviles y de escritorio, así como para la web.

55. mob

Evaluar

Algunas veces no te das cuenta de que necesitabas una herramienta hasta que se cruza en tu camino: mob es ese tipo de herramientas. Tal y como vivimos en un mundo donde el trabajo remoto en pareja se ha convertido en algo usual para muchos equipos, tener una herramienta que permite una colaboración sin fricciones entre 2 o incluso un mayor número de personas como parte de

una sesión de mob programming es altamente útil. mob oculta toda la parafernalia del control de versiones que hay tras una línea de comandos que hace que la participación en sesiones de mob programming sea más sencilla. También proporciona consejos específicos de cómo participar de forma remota, como por ejemplo, para "robar la pantalla compartida" en Zoom en lugar de terminarla, asegurando que la distribución de los elementos en la sesión no cambia para los participantes. Una herramienta útil y consejo a tener en cuenta, ¿cómo no gustarte?

56. Comandos modernos de Unix

Evaluar

Hay muchas razones para amar a Unix, pero la que ha afectado profundamente a nuestra industria es la filosofía de Unix de construir aplicaciones que "hagan una cosa y la hagan bien". Los comandos de Unix encarnan esta filosofía. Se trata de un conjunto de pequeñas funciones que pueden agruparse para crear soluciones más complejas. En los últimos años, los programadores han contribuido a un creciente conjunto de comandos Unix modernos. Estas versiones modernas intentan ser más pequeñas y rápidas, a menudo escritas en Rust. Incluyen características adicionales como el resaltado de sintaxis y utilizar características de los terminales modernos. Pretenden apoyar a los programadores de forma nativa mediante una buena integración con **`git`** y reconociendo los archivos de código fuente. Por ejemplo, **`bat`** es un sustituto de **`cat`** con paginación y resaltado de sintaxis; **`exa`** es un sustituto de **`ls`** con información ampliada de los archivos y **`ripgrep`** es un sustituto más rápido de **`grep`** que por defecto ignora los archivos gitignore, binarios y ocultos. El repositorio Modern Unix tiene una referencia a algunos de estos comandos. Hemos disfrutado usando estos comandos de Unix. Deberías probarlos para mejorar tu experiencia en la línea de comandos. Sin embargo, advertimos que no se deben utilizar en los scripts como sustitutos de las utilidades de línea de comandos estándar que se incluyen en las distribuciones del sistema operativo por defecto, ya que reducen la portabilidad de los scripts que se ejecutan en otras máquinas.

57. Mozilla Sops

Evaluar

Los secrets en texto plano registrados en el control de código fuente (normalmente Github) son uno de los errores de seguridad más comunes que cometen los desarrolladores. Por esta razón pensamos que es útil presentar Mozilla Sops, una herramienta para encriptar secrets en archivos de texto que nuestros desarrolladores encuentran útil en situaciones en las que es imposible eliminar los secrets de los repositorios de código heredados. Ya hemos mencionado muchas herramientas de este tipo (Blackbox, git-crypt), pero Sops tiene varias características que lo diferencian. Por ejemplo, Sops se integra con almacenes de claves(keystores) gestionados en la nube, como AWS y GCP Key Management Service (KMS) o Azure Key Vault, como fuentes de claves de cifrado. También funciona en varias plataformas y es compatible con las PGP keys . Esto permite un control de acceso detallado a los secrets de archivo por archivo. Sops deja la key de identificación en texto plano para que los secrets puedan seguir siendo localizados y difundidos por git. Siempre apoyamos cualquier cosa que facilite la seguridad de los desarrolladores; sin embargo, recuerda que, para empezar, no tienes que mantener los secrets en el control de código. Consulta Desacoplar la gestión de secrets del código fuente en nuestra edición de noviembre de 2017.

58. Operator Framework

Evaluar

Seguimos viendo la adopción de Kubernetes en escenarios nuevos y novedosos. Por ejemplo, vemos que Kubernetes se está ampliando para administrar recursos que se ejecutan fuera de su clúster o a través de múltiples proveedores de infraestructura, o se usa para administrar aplicaciones con estado más allá del alcance original de Kubernetes. Estas extensiones son posibles

utilizando el patrón Operador de Kubernetes creación de controladores de Kubernetes que tienen el conocimiento específico del dominio del recurso personalizado que gestionan. Por ejemplo, un operador que gestiona una aplicación con estado puede usar las primitivas de Kubernetes para automatizar las tareas específicas de una aplicación más allá de su implementación, como restaurar, respaldar y actualizar su base de datos.

Operator Framework es un conjunto de herramientas de código abierto que simplifica la creación y gestión del ciclo de vida de operadores de Kubernetes. Aunque hay múltiples marcos Para ayudarte a crear operadores de Kubernetes, Operator Framework sigue siendo una buena opción. Admite una gestión completa del ciclo de vida del operador mediante su módulo Operator Lifecycle Manager; admite varios idiomas para crear el propio código del operador utilizando su SDK del operador; y proporciona un catálogo para publicar y compartir el operadores. Si estás planeando crear operadores de Kubernetes, te recomendamos que pruebes Operator Framework para acelerar tu desarrollo de manera confiable.

59. Pactflow

Evaluar

Pensamos que vale la pena evaluar la utilidad de Pactflow para empresas que cuentan con grandes y complejos ecosistemas de API, especialmente si ya utilizan Pact. Pactflow gestiona los flujos de trabajo y el despliegue continuo de las pruebas escritas en Pact, disminuyendo la barrera hacia las pruebas de contrato dirigidas hacia el consumidor. La complejidad inherente que existe en la coordinación entre múltiples productores y varios consumidores dispares puede ser prohibitiva. Hemos visto que algunos equipos invierten un esfuerzo considerable en desarrollos “artesanales” para solucionar este problema y pensamos que vale la pena investigar si Pactflow puede encargarse de esto por ti.

60. Prefect

Evaluar

Prefect es una herramienta para la gestión de flujos de trabajo de datos que facilita añadir opciones como reintentos, mapeo dinámico, caché y notificaciones de error a un pipeline de datos. Puedes definir funciones de Python como task y encadenarlas mediante llamadas a la función para construir el flujo de datos. La API de Python combinada con una colección predefinida de tasks para operaciones comunes hacen de Prefect una opción destacable a valorar para las necesidades de tus pipelines de datos.

61. Proxyman

Evaluar

Puede que no sea una herramienta que necesites todos los días, pero cuando estás perdido, tratando de diagnosticar un problema complicado de red, es muy útil tener a la mano un proxy HTTP lleno de funcionalidades. Proxyman es justo esa herramienta. Muchos de nuestros equipos lo han estado usando desde hace un tiempo, como un reemplazo de Charles que además es específico para macOS. También comentan que les encanta la interfaz estilizada y su manejo de certificados.

62. Regula

Evaluar

Uno de los principios clave de la infraestructura como código (IaC) es la prueba automatizada. Si tenemos una pirámide de pruebas sólidas con una buena cobertura a nivel de código en la parte inferior, podemos producir una mejor y más segura infraestructura. Desafortunadamente, las herramientas utilizadas en este espacio han sido escasas. Conftest es utilizado frecuentemente para probar código Terraforn JSON and HCL, pero es una herramienta de uso general. Regula es una alternativa atractiva. Similar a Conftest, Regula comprueba el cumplimiento del código de

infraestructura mediante la aplicación de reglas escritas en lenguaje Open Policy Agent's Rego, pero también provee un conjunto de reglas específicas para validar configuraciones de infraestructura. Debido a que ambas herramientas están basadas en el lenguaje Rego, las reglas de Regula pueden ser ejecutadas en Conftest. Sin embargo, Regula viene con su propia herramienta de línea de comando para ejecutar pruebas como parte de un pipeline sin dependencia de Conftest u OPA. Nuestros desarrolladores han encontrado que Regula ahorra tiempo y produce un código de prueba mucho más legible, mantenible y sucinto, ambas herramientas solamente validan el código de infraestructura. Una suite completa también debería probar la infraestructura para asegurar que el código ha sido interpretado con precisión.

63. Sourcegraph

Evaluar

Otra herramienta de búsqueda de código basada en árbol de sintaxis abstracta que recibió nuestra atención es **Sourcegraph**. A diferencia de **Comby**, que es de código abierto, Sourcegraph es una herramienta comercial (con un limite de 10 usuarios en su nivel gratuito). Sourcegraph es particularmente adecuada para buscar, navegar o para hacer referencias cruzadas en bases de código grandes. La versión alojada en la nube puede ser accedida mediante el sitio web de Sourcegraph y está diseñada para buscar repositorios de código abierto públicos. Mientras que Comby es una herramienta ligera de línea de comando para automatizar tareas repetitivas, el énfasis de Sourcegraph está en las herramientas de desarrollo interactivas para entender y navegar grandes bases de código. A diferencia de la interfaz tipo sed de Comby, la capacidad reescritura de código automátizada de Sourcegraph se maneja desde una UI, lo que permite a quien lo use el revisar cambios antes de que sean hechos. Ya que Sourcegraph es un servicio alojado, también tiene la habilidad de continuamente monitorear bases de código y enviar alertas cuando una coincidencia ocurre.Whereas Comby is a lightweight command-line tool for automating repetitive tasks, Sourcegraph's emphasis is on interactive developer tools for understanding and navigating large code bases. Unlike Comby's **sed**-like interface, Sourcegraph's automated code rewriting capability is driven from a UI, allowing users to review changes before they're made. Because Sourcegraph is a hosted service, it also has the ability to continuously monitor code bases and send alerts when a match occurs.

64. Telepresence

Evaluar

Telepresence es una herramienta que ayuda a acortar el ciclo de feedback de los cambios que requieren un despliegue para realizar una prueba adecuada. Los desarrolladores pueden utilizarla para conectar un proceso que se ejecuta localmente en sus máquinas con un clúster remoto de Kubernetes. Así, se da acceso desde el proceso local a los servicios y herramientas del clúster remoto, y el servicio local puede reemplazar temporalmente alguno de los servicios del clúster. En situaciones donde la configuración de la integración del servicio se ha vuelto difícil de manejar, Telepresence puede mejorar la productividad del desarrollador y permitir una prueba local más efectiva. Sin embargo, si se acostumbra a usar una herramienta inteligente como esta, puede encontrarse con problemas más graves. Por ejemplo, si se utiliza Telepresence porque se ha vuelto imposible configurar todas las dependencias necesarias para el desarrollo en local, se debería investigar la complejidad de la configuración y arquitectura. Si Telepresence se convierte en la única manera de hacer pruebas de integración de servicios, sería interesante consultar **pruebas dirigidas por contratos con el consumidor** u otras formas automáticas para hacer pruebas de integración.

65. Vite

Evaluar

El feedback rápido es crucial para una buena experiencia de desarrollador. Nada interrumpe más el flujo de desarrollo que tener que esperar uno o dos minutos antes de recibir comentarios sobre los últimos cambios de código. Desafortunadamente, con las aplicaciones que crecen en tamaño y complejidad, las populares herramientas de compilación para las pipelines de front-end, a menudo ya no son lo suficientemente rápidas. Anteriormente, presentamos **esbuild**, que ofrece una mejora significativa del rendimiento, ya que se implementa en un lenguaje compilado a nativo en lugar de JavaScript. **Vite**, construido sobre esbuild, ofrece **mejoras significativas** sobre otras herramientas. Consta de dos partes principales: un servidor de desarrollo que proporciona mejoras de funciones completas sobre los módulos ES nativos, como el reemplazo de módulo en caliente (HMR) extremadamente rápido, y un comando de compilación que empaqueta su código con Rollup. Vite se basa en módulos ES y, a diferencia de la mayoría de las herramientas más antiguas, no proporciona shimming ni polyfills, lo que significa que no es compatible con navegadores más antiguos que no admiten módulos ES. En aquellos casos en que teníamos que soportar navegadores más antiguos, algunos de nuestros equipos utilizaron Vite durante el desarrollo y otras herramientas para las compilaciones de producción.

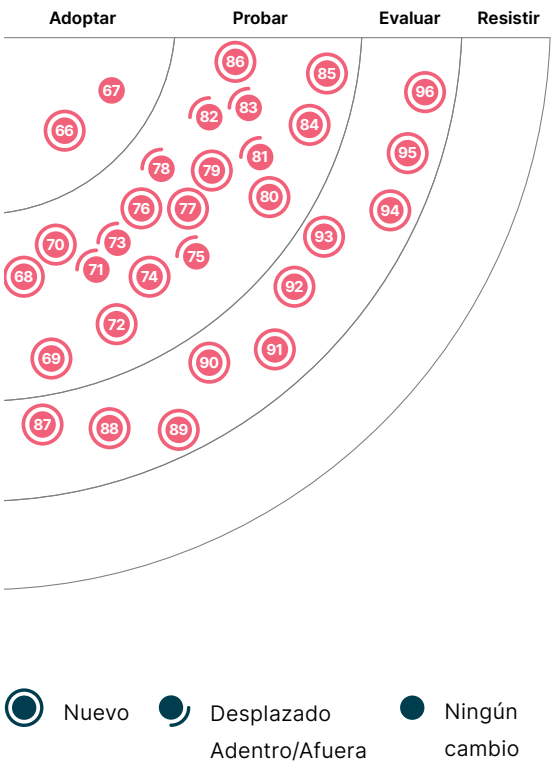
Lenguajes y Frameworks

Adoptar
66. Jetpack Compose
67. React Hooks

Probar
68. Arium
69. Chakra UI
70. DoWhy
71. Gatsby.js
72. Jetpack Hilt
73. Kotlin Multiplataforma Móvil
74. lifelines
75. Mock Service Worker
76. NgRx
77. pydantic
78. Quarkus
79. React Native Reanimated 2.0
80. React Query
81. Tailwind CSS
82. TensorFlow Lite
83. Three.js
84. ViewInspector
85. Vowpal Wabbit
86. Zap

Evaluar
87. Headless UI
88. InsightFace
89. Kats
90. ksqlDB
91. Polars
92. PyTorch Geometric
93. Qiankun
94. React Three Fiber
95. Tauri
96. Transloco

Resistir
—



66. Jetpack Compose

Adoptar

En una maniobra que imita a la presentación por parte de Apple de SwiftUI, Google ha presentado Jetpack Compose como un enfoque nuevo y poco diferente al desarrollo de interfaces para aplicaciones Android modernas. Compose ofrece herramientas más poderosas y una intuitiva API de Kotlin. En la mayoría de los casos es necesario menos código, y se ha vuelto más fácil crear interfaces de usuario en tiempo de ejecución en lugar de definir una UI estática que pueda rellenarse con datos. Con Compose Multiplatform y Kotlin Multiplatform los desarrolladores cuentan con un set de herramientas unificado para construir aplicaciones de escritorio, web y nativas Android. También se incluye Wear OS 3.0+, y con el soporte para iOS ya presente en Kotlin Multiplatform Mobile, es probable que IOS sea soportado por Compose en el futuro.

67. React Hooks

Adoptar

React Hooks ha introducido un nuevo enfoque en la gestión de la lógica de los estados, dado que los componentes de React siempre han estado más cerca de ser funciones que clases, los Hooks han adoptado esto y han llevado el estado a las funciones, en vez de usar clases para llevar funciones al estado con métodos. Otro elemento básico de la gestión del estado en las aplicaciones React es Redux, y ya hemos notado que ha sido objeto de escrutinio, lo que sugiere que a veces la complejidad de Redux no vale la pena y, en tales casos, es preferible un enfoque simple usando Hooks. Hacer esto completamente solo puede volverse complicado rápidamente; por lo tanto, recomendamos considerar una combinación de React Context y los hooks useContext y useReducer, explicadas en este blog post.

68. Arium

Probar

Arium es un framework de pruebas automatizadas para aplicaciones en 3D escrito en Unity. Las pruebas funcionales son una parte importante de la pirámide de pruebas saludables. Arium, el cual es construido como un wrapper en el framework de pruebas de Unity, , permite escribir pruebas funcionales para apps en 3D en múltiples plataformas. Lo hemos utilizado exitosamente en algunos de nuestros proyectos.

69. Chakra UI

Probar

Chakra UI es una biblioteca de componentes de interfaz de usuario para React.js que está diseñada para la accesibilidad. Nos gusta, especialmente por sus características de accesibilidad, incluyendo el modo oscuro y la compatibilidad con las directrices de la Iniciativa de Accesibilidad Web – Aplicaciones nriquecidas de Internet Accesibles (WAI-ARIA). Es más, es fácil de probar y personalizar lo que contribuye a una buena experiencia de desarrollo, acelerando el proceso de desarrollo de soluciones UI en ambientes de producción.

70. DoWhy

Probar

DoWhy es una biblioteca de Python para realizar inferencia causal y análisis de punta a punta.

Aunque los modelos de machine learning pueden hacer predicciones basados en datos fácticos aprovechando la correlación entre variables que estuviesen presente en ese momento, son insuficientes en escenarios en los que tenemos que hacer preguntas del tipo y si y por qué: ¿Y si cambiase una variable? ¿Cuál sería el impacto en el resultado? La inferencia causal es un enfoque para responder a estas preguntas. La inferencia causal estima el efecto causal, es decir, la magnitud en la que un resultado cambiaría si cambiáramos una de sus variables. Este enfoque se utiliza cuando no podemos obtener una respuesta mediante observaciones y recopilando datos mediante pruebas A/B - debido al costo de los experimentos o limitaciones. La biblioteca DoWhy estima el efecto causal basado en un proceso que usa los hechos y datos recopilados con anterioridad, así como las suposiciones que se pueden hacer al conocer el dominio. Utiliza un proceso de cuatro pasos para modelar el gráfico de relaciones causales basado en supuestos, identificando la causa de un resultado, estimando el efecto causal y finalmente desafiando esos supuestos refutando el resultado. Hemos utilizado esta biblioteca con éxito en producción y es una de las bibliotecas más utilizadas en casos de uso de estimación causal.

71. Gatsby.js

Probar

A pesar de que muchos frameworks prometen la misma facilidad en el desarrollo y escalabilidad típica en los generadores de sitio estático (SSR), continuamos teniendo buenas experiencias con Gatsby.js. En particular, lo hemos usado para construir y desplegar sitios web que escalan a gran cantidad de usuarios sin tener que preocuparnos por planificar la capacidad o infraestructura de despliegue. Nuestros desarrolladores también se han impresionado por el enfoque en la accesibilidad y soporte a navegadores antiguos, y a que pueden reutilizar su experiencia en React.js En definitiva, sentimos que Gatsby ha madurado bien y es una opción sólida en este espacio.

72. Jetpack Hilt

Probar

Jetpack Hilt recientemente llegó a su versión 1.0, y podemos reportar que hemos tenido buenas experiencias. Jetpack Hilt ofrece extensiones para integrar Hilt con varias otras librerías de AndroidX tales como: WorkManager y Navigation. Expande aún más el alcance de Hilt para proveer a los desarrolladores con una manera estándar de incorporar inyecciones de dependencias con Dagger en las aplicaciones de Android. Hemos presentado a Koin como a una librería nativa en Kotlin para inyecciones de dependencias en el Radar anteriormente, y no aconsejamos reemplazar a Koin en grandes bases de codigo ya existentes. Sin embargo, al iniciar un nuevo proyecto, Hilt al parecer, es la herramienta a utilizar.

73. Kotlin Multiplataforma Móvil

Probar

Para muchas organizaciones, el desarrollo móvil multiplataforma se está convirtiendo en una opción fuerte, especialmente porque la experiencia de construir aplicaciones móviles multiplataforma se vuelve más agradable y eficiente. Kotlin Multiplatform Mobile (KMM) es un SDK proporcionado por JetBrains que aprovecha las capacidades multiplataforma de Kotlin e incluye herramientas y características diseñadas para agilizar la experiencia del desarrollador. Con KMM se escribe el código una sola vez para la lógica de negocio y el núcleo de la aplicación en Kotlin y luego se comparte con las aplicaciones de Android e iOS. Sólo escribes el código específico de la plataforma cuando es necesario, por ejemplo, para aprovechar los elementos nativos de la interfaz de usuario; y el código específico se mantiene en vistas diferentes para cada plataforma. Estamos trasladando KMM a Trial, ya que está evolucionando rápidamente y estamos viendo que algunas organizaciones

lo utilizan por defecto.

74. lifelines

Probar

lifelines es una librería para análisis de supervivencia en Python. Originalmente desarrollada para eventos de nacimiento y muerte, ha evolucionado en una librería completa de análisis de supervivencia para predecir cualquier duración de tiempo. Más allá de los casos de uso en medicina (como responder a, ¿Cuanto tiempo vive esta población?), la hemos utilizado en retail y fabricación para contestar a preguntas como ¿Cuánto tiempo están los usuarios suscritos a un servicio? o ¿Cuando deberíamos realizar el próximo mantenimiento preventivo?

75. Mock Service Worker

Probar

Las aplicaciones web, especialmente las de uso interno en las empresas, suelen estar escritas en dos partes. La interfaz de usuario y parte de la lógica de negocio se ejecutan en el navegador, mientras que la lógica de negocio, la autorización y la persistencia se ejecutan en un servidor. Estas dos partes se comunican normalmente a través de JSON sobre HTTP. Los endpoints no deben confundirse con una API real; son simplemente un detalle de implementación de una aplicación que se divide en dos entornos de ejecución. Al mismo tiempo, proporcionan una unión válida para probar las piezas individualmente. Al probar la parte de JavaScript, el lado del servidor puede ser simulado a nivel de red por una herramienta como Mountebank. Mock Service Worker ofrece un enfoque alternativo para interceptar las peticiones en el navegador. Esto también simplifica las pruebas manuales. Al igual que Mountebank, Mock Service Worker se ejecuta fuera del navegador como un proceso Node.js para probar las interacciones de red. Además de interacciones REST, simula las APIs GraphQL - una ventaja porque GraphQL puede ser complejo de simular manualmente a nivel de red.

76. NgRx

Probar

La gestión de los estados en Aplicaciones React ha sido un tema recurrente en el Radar, y recientemente hemos clarificado nuestra posición sobre Redux, un framework popular en este espacio. NgRx es, en esencia, Redux para Angular. Este es un framework para construir aplicaciones reactivas con Angular, aportando maneras de gestionar estados y aislar efectos secundarios. Nuestros equipos reportan que entender NgRx fue sencillo, sobre todo porque está construido con RxJS, ellos destacan un beneficio similar a la que conocemos de Redux: agregar gestión de estado reactivo viene con una complejidad adicional que solo tiene sentido en aplicaciones grandes. La experiencia del desarrollador se ve reforzada por esquemas, una librería de scaffolding, así como también por un conjunto de herramientas que permiten el seguimiento visual del estado y la depuración time-travel.

77. pydantic

Probar

Originalmente, se agregaron las anotaciones de tipo a Python para apoyar el análisis estático. Sin embargo, teniendo en cuenta la amplitud con la que se utilizan las anotaciones de tipo, y las anotaciones en general, usadas en otros lenguajes de programación, era sólo cuestión de tiempo que los desarrolladores comenzaran a utilizar las anotaciones de tipo de Python para otros fines.

pydantic entra en esta categoría. Permite utilizar anotaciones de tipo para la validación de datos y la gestión de configuraciones en tiempo de ejecución. Cuando los datos llegan como, por ejemplo, un documento JSON y necesitan ser analizados en una estructura compleja de Python, pydantic asegura que los datos entrantes coinciden con los tipos esperados o informa de un error si no es así. Aunque puedes utilizar pydantic directamente, muchos desarrolladores lo han utilizado como parte de FastAPI, uno de los frameworks web más populares de Python. De hecho, el uso de pydantic en FastAPI se considera tan indispensable que un cambio propuesto recientemente en Python, destinado a reducir el coste de la carga de código anotado en memoria, fue reconsiderado porque habría roto el uso de anotaciones de tipo en tiempo de ejecución.

78. Quarkus

Probar

Evaluamos Quarkus thace dos años, y ahora nuestros equipos tienen más experiencia con él. Quarkus es una pila Java nativa de Kubernetes diseñada para OpenJDK HotSpot y GraalVM. En los últimos dos años, Quarkus ha conectado las mejores bibliotecas del mundo Java y simplificado la configuración del código, lo que ha proporcionado a nuestros equipos una buena experiencia de desarrollo. Quarkus tiene un tiempo de arranque muy rápido (decenas de milisegundos) y tiene una baja huella de memoria RSS; esto se debe a su enfoque de construcción container-first: utiliza técnicas de compilación anticipada para hacer la inyección de dependencias en tiempo de compilación y, por lo tanto, evita los costes de ejecución de la reflexión. Nuestro equipo también ha tenido que hacer concesiones: Quarkus tarda casi 10 minutos en construirse en nuestro pipeline; algunas características que dependen de anotaciones y reflexión (como el ORM y el serializador) también están limitadas. Parte de estas concesiones son el resultado de usar GraalVM. Así que si tu aplicación no se ejecuta como FaaS, el uso de Quarkus con HotSpot es también una buena opción.

79. React Native Reanimated 2.0

Probar

Si queremos animaciones en aplicaciones React Native, React Native Reanimated 2.0 es el camino. Previamente teníamos Reanimated 1.x, pero tenía problemas relacionados con la complejidad del lenguaje declarativo de Reanimated y también tenía algunos costos de rendimiento adicionales relacionados con la inicialización y comunicación entre el hilo de JavaScript de React Native y el hilo de UI. Reanimated 2.0 es un intento de reimaginar cómo ejecutar animaciones en el hilo de UI; esto nos permitirá programar animaciones en JavaScript y ejecutarlas en el hilo de UI utilizando un nuevo API llamado animation worklets. Para esto se genera un contexto secundario de JavaScript en el hilo de UI que puede entonces ejecutar las funciones de JavaScript. Nosotros estamos utilizando esto en nuestros proyectos de React Native que necesitan animaciones y nos gusta mucho.

80. React Query

Probar

React Query se describe a menudo como la biblioteca de recuperación de datos que faltaba para React. Obtener, almacenar en caché, sincronizar y actualizar el estado de un servidor son requerimientos comunes en muchas de las aplicaciones de React y, aunque los requisitos están claros, lograr la implementación correcta es considerablemente más difícil. React Query proporciona una solución sencilla utilizando hooks (ganchos). Como desarrollador de aplicaciones, simplemente pase una función que resuelva sus datos y deje todo lo demás al framework (marco de trabajo). Nos gusta que funcione por sí solo, pero que siga ofreciendo opciones de configuración cuando se

necesiten. Las herramientas para el desarrollador, desafortunadamente no están disponibles todavía para React Native, sí que ayudan a entender cómo funciona el framework, lo que beneficia a los desarrolladores que son nuevos en ello. En nuestra experiencia, la versión 3 del framework trae la estabilidad necesaria para que sea utilizado en producción con nuestros clientes.

81. Tailwind CSS

Probar

Nuestros desarrolladores continúan su productividad con **Tailwind CSS** y están impresionados con su capacidad para escalar tanto con grandes equipos como con código base. Tailwind CSS ofrece un enfoque alternativo a las herramientas y marcos CSS que reducen la complejidad mediante clases CSS de utilidad de nivel inferior. Las clases CSS de Tailwind pueden personalizarse fácilmente para adaptarse a la identidad visual de cualquier cliente. También hemos comprobado que combina muy bien con **Headless UI**. Tailwind CSS evita que tengas que escribir cualquier clase o CSS por tu cuenta, lo que conduce a una base de código más fácil de mantener a largo plazo. Parece que Tailwind CSS ofrece el equilibrio adecuado entre reutilización y personalización para crear componentes visuales.

82. TensorFlow Lite

Probar

Desde que hablamos por primera vez de **TensorFlow Lite** en el Radar de 2018, lo hemos usado en varios productos y estamos contentos de confirmar que funciona cómo prometía. El caso de uso más común es para integrar modelos pre-entrenados en aplicaciones móviles, pero TensorFlow Lite también soporta el proceso de aprendizaje en el propio dispositivo, lo que permite más áreas de aplicación. En el sitio web hay numerosos ejemplos que muestran áreas comunes de aplicación, como clasificación de imágenes y detección de objetos, pero también da indicios de nuevas formas de interacción, utilizando, por ejemplo, la estimación de poses y el reconocimiento de gestos.

83. Three.js

Probar

En 2017 mencionamos **Three.js** en el Radar de Assess. Desde entonces, esta biblioteca de renderizado 3D para la web ha evolucionado y mejorado rápidamente. Las API estándar de WebGL han mejorado y Three.js ha agregado soporte para WebXR, convirtiéndolo en una herramienta viable para crear experiencias inmersivas. Al mismo tiempo, ha mejorado la compatibilidad del navegador con la representación 3D y las API de dispositivos WebXR, lo que convierte a la web en una plataforma cada vez más atractiva para el contenido 3D. Aunque existen otras bibliotecas de renderizado 3D, nuestros equipos han llegado a preferir Three.js, especialmente cuando se combina con **React Three Fiber** para abstraer algunos de los detalles de bajo nivel. Descubrimos que los desarrolladores aún deben ser conscientes de los problemas de rendimiento y, a veces, necesitarán reestructurar los datos para optimizar la velocidad de renderizado.

84. ViewInspector

Probar

Cuando se crea una interfaz de usuario con **SwiftUI**, la idea es construir un modelo de vistas que puedan ser fácilmente relacionadas con los elementos de la interfaz de usuario. En estos casos, la mayor parte de las comprobaciones se pueden realizar en el propio modelo, usando frameworks estándar de pruebas unitarias que hacen estos tests fáciles de escribir y muy rápidos de ejecutar.

Para probar la relación entre el modelo y las vistas, los desarrolladores normalmente se decantan por XCUITest, un framework para construir tests automatizados que ejecuta toda la aplicación y controla la interfaz de usuario de forma remota. Funciona, los tests son razonablemente estables, pero requieren mucho tiempo de ejecución.

Para una alternativa más rápida de pruebas unitarias con SwiftUI, prueba ViewInspector, un framework de código abierto que usa la API pública de reflection de SwiftUI para acceder a las definiciones de las vistas creadas por SwiftUI. Con ViewInspector, un test simplemente crea una instancia de una vista de SwiftUI, localiza los elementos de la interfaz de usuario que se quieren probar y realiza las comprobaciones necesarias contra los mismos. Las interacciones básicas como una pulsación táctil también se pueden comprobar. Como muchos otros frameworks de test para interfaces de usuario, proporciona una API para localizar elementos de la interfaz, ya sea mediante el uso de una ruta específica a través de la jerarquía visual o usando un conjunto de funciones de búsqueda. Este tipo de pruebas son normalmente más sencillas que las realizadas con XCUITests y se ejecutan mucho más rápido. Cabe tener en cuenta, sin embargo, que debido a la facilidad que ViewInspector ofrece para escribir este tipo de tests, podrías caer en la tentación de sobrepasarte probando la interfaz de usuario. Comprobar cada una de las relaciones una a una implica el doble de código a mantener. Aunque ViewInspector hace más sencillo probar el código de SwiftUI, recuerda mantener la mayor parte de la lógica en el modelo.

85. Vowpal Wabbit

Probar

Vowpal Wabbit es una biblioteca de machine-learning de propósito general. Aunque fue creada originalmente en Yahoo! Research hace más de una década, queremos mencionarla para resaltar que sigue siendo el lugar donde se añaden primero muchas de las técnicas más nuevas de machine-learning. Si te interesa machine-learning, quizá quieras echar un vistazo a las innovaciones de Vowpal Wabbit. Ten en cuenta también que Microsoft ha mostrado un mayor interés en Vowpal Wabbit en los últimos años, contratando características principales e integrándolo en sus ofertas de Azure, por ejemplo en su machine-learning designer y en Personalizer.

86. Zap

Probar

Zap es una librería de registro estructurado de alto rendimiento para GoLang que es más rápida que las implementaciones de registro estándar y que otras bibliotecas de registrado. Zap tiene un registrador "bonito", que proporciona una interfaz estructurada y de estilo **`printf`**, así como una implementación (aún) más rápida con solo la interfaz estructurada. Nuestros equipos lo han utilizado ampliamente a la escala necesaria y están felices de recomendarlo como su solución preferida.

87. Headless UI

Evaluar

Headless UI es una biblioteca de componentes sin estilo para React.js o Vue.js de la misma gente que creó Tailwind CSS. A nuestros desarrolladores les gusta no tener que personalizar o trabajar con los estilos por defecto que traen otras bibliotecas de componentes. La gran funcionalidad de los componentes y su total accesibilidad, combinada con el estilo sin fricciones, permite a los desarrolladores enfocarse de forma más productiva en el problema del negocio y en la experiencia del usuario. Como es de esperar, Headless UI también se combina muy bien con las clases CSS de

Tailwind.

88. InsightFace

Evaluar

InsightFace es un conjunto de herramientas de código abierto para análisis profundo de rostros en 2D y 3D, principalmente basado en PyTorch y MXNet. InsightFace usa algunos de los más recientes y precisos métodos para detección, reconocimiento y ajuste de rostros. Nos interesa especialmente porque tiene una de las mejores implementaciones para ArcFace, un modelo de machine learning de vanguardia que detecta las similitudes entre dos imágenes. InsightFace con ArcFace obtuvieron una puntuación del 99.83% de precisión en el conjunto de datos Labeled Faces in the Wild (LFW) Estamos experimentando con ella en el contexto de la de-duplicación facial y hemos visto resultados prometedores.

89. Kats

Evaluar

Kats es un framework ligero para realizar análisis de series temporales, recientemente lanzado por Facebook Research. Las series temporales son una importante área en la ciencia de datos; abarca los ámbitos de problema de previsión, detección (incluyendo detección de estacionalidades, valores atípicos y puntos de cambio), extracción de características y análisis multivariable. Típicamente solemos tener diferentes librerías para diferentes técnicas en un análisis de series temporales. Kats, en cambio, pretende ser una solución única para los análisis de series temporales y proporciona un conjunto de algoritmos y modelos para todos los dominios de problemas de análisis de series temporales. Previamente mencionamos Prophet, también de Facebook Research, el cual es uno de los modelos que implementa Kats para la previsión. Esperamos probar Kats en problemas que involucran análisis de series temporales.

90. ksqlDB

Evaluar

Si estás usando Apache Kafka y construyendo aplicaciones de procesamiento de stream, ksqlDB es un gran framework para escribir aplicaciones simples utilizando sentencias similares a SQL. ksqlDB no es una base de datos tradicional SQL. Sin embargo, permite utilizar sentencias ligeras similares a sentencias SQL para construir nuevos streams de Kafka o tablas en adición a los topicos Kafka existentes. Las sentencias pueden obtener datos, similar a leer de una base de datos tradicional, o enviar resultados a la aplicación cuando ocurren cambios incrementales. Puedes escoger ejectuarlo como un servidor independiente nativamente como parte de tu instalación Apache Kafka existente, o como un servicio totalmente administrado en Confluent Cloud. Sugerimos utilizar ksqlDB en casos de uso de procesamiento de datos simples. Para casos de uso más complejos, cuando una aplicación requiere código de programación más allá de sentencias SQL algebraicas, nosotros seguimos utilizando frameworks de procesamiento de datos como Apache Spark o Apache Flink además de Kafka. Recomendamos experimentar con ksqlDB en escenarios donde la simplicidad de la aplicación lo permite.

91. Polars

Evaluar

Polars es una biblioteca de marcos de datos en memoria implementada en Rust. A diferencia de

otros marcos de datos (como Pandas), Polars es multihilo y seguro para operaciones paralelas. Los datos en memoria están organizados en el formato **Apache Arrow** para realizar operaciones analíticas eficientes y permitir la interoperabilidad con otras herramientas. Si estás familiarizado con Pandas, puedes empezar rápidamente con los enlaces de Python para Polars. Creemos que Polars, con la implementación de Rust y los enlaces de Python, es un marco de datos en memoria de alto rendimiento para evaluar sus necesidades analíticas.

92. PyTorch Geometric

Evaluar

PyTorch Geometric Es una biblioteca de extensión de aprendizaje profundo geométrico para **PyTorch**. El deep learning geométrico tiene como objetivo construir redes neuronales que puedan aprender de datos no euclidianos como los gráficos. Los enfoques de Graph network-based machine-learning han sido de creciente interés en el modelo de redes sociales y en los campos biomédicos, específicamente en el descubrimiento de fármacos. PyTorch Geometric proporciona una biblioteca fácil de usar para diseñar complicados problemas de gráficos como la representación de estructuras de proteínas. Tiene soporte para GPU y CPU e incluye una buena colección de algoritmos de graph-based machine-learning y en investigaciones recientes.

93. Qiankun

Evaluar

Los **micro frontends** han ido haciéndose cada vez más populares desde que se introdujeron por primera vez. Sin embargo, es sencillo caer en la **anarquía de micro frontends** si los equipos fallan en mantener la consistencia en toda la aplicación, desde las técnicas de estilo a la gestión de estados. **Qiankun**, que significa cielo y tierra en Chino, es una biblioteca de JavaScript construida para proveer una solución out-of-the-box para este problema. Qiankun está basado en **single-spa**, de manera que permite que diferentes marcos de trabajo coexistan en una única aplicación. También provee aislación de estilos y un entorno de pruebas de Javascript para asegurar que el estilo o estado de las microaplicaciones no interfieren con los de otras. Qiankun ha recibido cierta atención en la comunidad; nuestros equipos también lo están probando, con la esperanza de que pueda soportar una depuración más amigable.

94. React Three Fiber

Evaluar

Con el creciente interés — y viabilidad — de las aplicaciones 3D y de realidad extendida (XR) en los navegadores web, nuestros equipos han estado experimentando con **React Three Fiber** para desarrollar experiencias 3D en la web. React Three Fiber es una biblioteca que toma el modelo de componentes y estados de React.js y lo traduce a objetos 3D renderizados con la biblioteca **Three.js.** Este enfoque abre la programación web en 3D a un grupo más amplio de desarrolladores que ya están familiarizados con React.js y el abundante ecosistema de herramientas y bibliotecas que lo rodean. Sin embargo, al desarrollar aplicaciones con React Three Fiber, nuestros equipos a menudo tienen que manipular la escena 3D imperativamente. Esto no combina bien con el paradigma de componentes que proporciona React. No hay forma de escapar a la necesidad de entender los mecanismos básicos de renderizado 3D. El comité aún no ha decidido si React Three Fiber ofrece suficiente abstracción para justificar su aprendizaje o si es mejor trabajar con Three.js directamente.

95. Tauri

Evaluar

Tauri es una alternativa de Electron para construir aplicaciones de escritorio utilizando una combinación de herramientas de Rust y HTML, CSS y JavaScript renderizados en el System's WebView. A diferencia de Electron, que incluye Chromium, las aplicaciones construidas con Tauri aprovechan el WebView subyacente, es decir, WebKit en macOS, WebView2 en Windows y WebKitGTK en Linux. Este enfoque tiene interesantes ventajas y desventajas: por un lado, obtiene binarios de aplicación pequeños y rápidos; por otro, es necesario verificar las peculiaridades de compatibilidad entre los WebViews de diferentes sistemas.

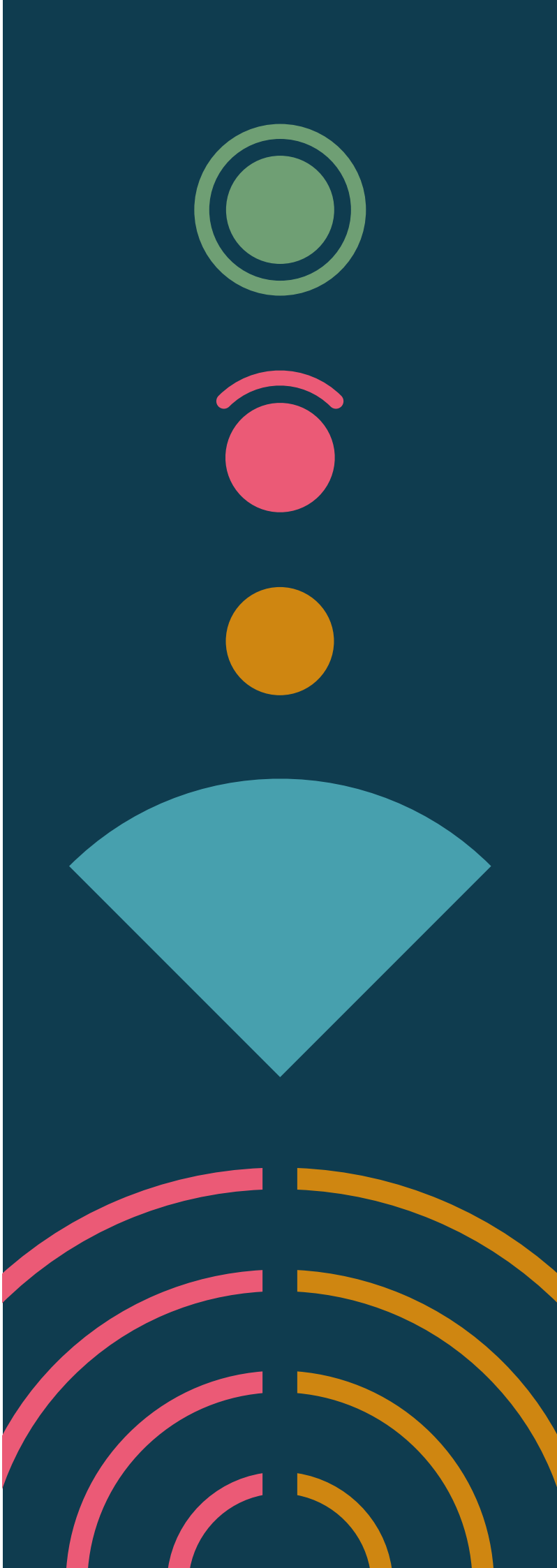
96. Transloco

Evaluar

Transloco es una librería para Angular para construir aplicaciones multilingües. Puede ser usada en plantillas y ofrece una función para cubrir casos de uso más complejos. Debido a que las traducciones se cargan a demanda en tiempo de ejecución, Transloco facilita la implementación del cambio de idioma en el navegador web. También cubre localización de números, fechas, y más usando pipes de plantilla.

Equipo de traducción al Español

Alejandro Batanero, Álvaro Paz, Ana Jimenez Valbuena, Ana Lallena Arquillo, Andrés de los Reyes, Antonio Gálvez Montes, Araceli Correa, Ayoze Hernández, Camila Vigneaux, Carlos Barroso Baltasar, Daniel Santibañez, David Corrales, Diana Barreno Diana Luna, Fernando Naranjo, Francisco Plaza, Gabriel Frias Rivas, Gabriel Loja, Gaby Gurfinkel, Geovanny Campoverde, Giovanny Sayas, Irma Blanco Fernández, Iván Gómez Rodríguez,, Jesús Cardenal, Jhosep Marin, Joan Galvan, Joan Sánchez, Jorge Agudo Praena, José Herdoíza, José Luis Lebrón, José Peñaherrera, José Rodríguez Toledo. José Zurita. Juan Cabrera, Juan Pazmiño, Julieta Pilar Corvi Peral, Katherine Ayala, Kelly Landázuri, Laura Bassani, Luciano Iannicelli, Pablo Arciniega, Paola Cajilema, Paola Jiménez, Paula Marin, Pedro Carbonell, Pedro Grijalva, Rodrigo Campos Hernández, Rodrigo Nogués, Sebastian Muñoz, Sebastian Roman, Sergio Jimenez, Silvina Calderon, Tex Albuja, Victor Cordova, Winston Castillo, Xavier Idrovo, Yafo Pereiro, Dave Montano, Luis Aucañir Huaiquinao, Luis Burgos, Luis Bustamante, Mafer Escudero, Manu Escudero, Manuel David Vicent Gimenez, Marcelo “Mamgui” Morales Padilla, Marcelo Cartagena, María Córdova, Mayfe Yépez, Mia Krasteva, Michelle Salguero, Mijail Rondon, Milber Champutiz, Mónica Giraldo, Neysa Alarcón, Nicol Rafalowski, María José Lalama, Elizabeth Parra, Magdalena Grondona y Daniel Negrete.



¿Quieres estár al tanto de todas las noticias e insights relacionadas al Radar?

Síguenos en tu red social favorita o
conviértete en un suscriptor/a.

suscríbete ahora

Thoughtworks es una consultora global de tecnología que integra estrategia, diseño e ingeniería para impulsar la innovación digital. Somos 10,000+ personas en 48 oficinas en 17 países. En los últimos 25+ años, hemos logrado un impacto extraordinario junto con nuestros clientes, ayudándoles a resolver problemas complejos de negocio a través de la tecnología como elemento diferenciador.

/thoughtworks