Radar Tecnológico

Una guía con opiniones sobre las tecnologías de vanguardia

/thoughtworks

Volume 27

Sobre el Radar

Thoughtworkers son personas a las que les apasiona la tecnología. La construimos, la investigamos, la probamos, abogamos por el código abierto, escribimos sobre ella y constantemente tratamos de mejorarla - para todas las personas. Nuestra misión es defender la excelencia del software y revolucionar la TI. Creamos y compartimos el Radar Tecnológico de Thoughtworks en apoyo de esa misión. El Technology Advisory Board de Thoughtworks, un grupo de líderes tecnológicos de alto nivel de Thoughtworks, crea el Radar. Se reúnen periódicamente para debatir la estrategia tecnológica global de Thoughtworks y las tendencias tecnológicas que tienen un impacto significativo en nuestra industria.

El Radar recoge el resultado de los debates del Technology Advisory Board en un formato que proporciona valor a una amplia gama de partes interesadas, desde las personas desarrolladoras hasta CTOs. El contenido pretende ser un resumen conciso.

Te animamos a explorar estas tecnologías. El Radar es de naturaleza gráfica y agrupa los elementos en técnicas, herramientas, plataformas y lenguajes y frameworks. Cuando los elementos del Radar podían aparecer en varios cuadrantes, elegimos el que nos pareció más apropiado. Además, agrupamos estos elementos en cuatro anillos para reflejar nuestra posición actual al respecto.

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 convencidas de que la industria debería adoptar estos ítems. Nosotras 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.

Equipo de traducción al Español: Alejandra Quintana, Alejandro Sánchez Carrasco, Alex Ulloa, Alexander López, Ana Carolina Díaz, Ana María Zuñiga, Ana Parra Montagne, Andrea Montoya, Andrea Peralta Bravo, Angie Bracho, Araceli Correa Mendoza, Arturo Menendez, Atik Yumbay, Bárbara Deluchi, Camila Vigneaux, Carlos Barroso, Carolina Melgarejo Pezoa, Caroline Tamayo, Catalina Solís, Cecilia Geraldo, Christian Alvarez, Claudia Lertora Ginés, Clemencio Morales Lucas, Cristian Montero, Cristopher Cardenas, Daniel Santibañez, Daniel Negrete, David Benitez Riba, Diana Luna, Diana Pila, Elisa Torres, Elizabeth Sanchez, Elizabeth Parra, Enrique Aracil Mas, Erika Pila, Erika Vacacela, Esteban Grijalva, Esteban Moreno Arias, Eva Cobos Cortina, Eva Villarroya Lorenzo, Felipe Jorquera, Fernanda Pérez, Fernando del Caz, Fernando Tamayo, Ferran Gay Boelle, Francis Alcántara, Francisca Castro Barriga, Gabriel Frias Rivas, Gabriel Loja, Gabriela Marques, Gisela Yumi, Guillermo Garcia Juanes, Gustavo Chiriboga, Gustavo Marin, Hernan Cid Montero, Irene Amo, Irene Torres, Isabel Hong, Ivan Almendros, James Reguart, Javier Escobar, Jenny Huang Chen, Jesús Cardenal, Jhosep Marin, Joan Sanchez Garcia, Joel Armada Herrera, Jorge Agudo, Jorgina Arres Cardona, Jose María Alonso Montero, José Vera, Juan Antonio, Magdalena Grondona, Mateos Rueda, María José Lalama, Johan Ponguillo, Juan Cabrera, Juan Infante Zumer, Juan Romero, Gomez, Katherine Ayala, Laura Mirás, Lucia Parga, Luis Maracara, Manuela Recacochea, Marcelo Cartagena, Maria Montoza, Mayfe Yèpez, Milber Champutiz, Nahuel Delgado, Naileth Rivero, Nati Rivera, Nestor Velazquez, Nicol Rafalowski, Norbs Rodriguez, Patricia Zurita, Pedro Carbonell, Ricardo Armas, Rodrigo Nogués, Rodrigo Tagle, Rubén Trujillo, Sebastian Roman Jarrin, Sergio Espinoza, Silvina Calderon, Wilder Bravo, Xavier Idrovo, Yessenia Erraez y Zully Arellano

Contribuyentes

El Technology Advisory Board (TAB) es un grupo de 17 personas tecnólogas senior de Thoughtworks. El TAB se reúne dos veces al año en persona y virtualmente cada dos semanas. Su función principal es ser un grupo de asesoramiento para el CTO de Thoughtworks, Rebecca Parsons.

El TAB actúa como un organismo amplio que puede examinar los temas que afectan a la tecnología y a las personas tecnólogas en Thoughtworks. Esta edición del Radar Tecnológico de Thoughtworks es en base a la reunión presencial que el TAB realizó en Barcelona en Septiembre 2022.

Rebecca Parsons (CTO)

Martin Fowler (Chief Scientist)

Bharani Subramaniam

Birgitta Böckeler

Brandon Byars

Camilla Falconi Crispim

Erik Dörnenburg

Fausto de la Torre

Hao Xu

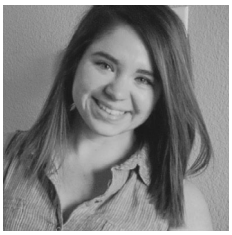
Ian Cartwright

James Lewis

Marisa Hoenig

Mike Mason

Neal Ford



Perla Villarreal

Scott Shaw

Shangqi Liu

Temas

La integración de ML

Machine Learning (ML) fue alguna vez un campo reservado solo para unos pocos afortunados que tenían las herramientas y los recursos para construir cosas geniales. Afortunadamente, vemos una popularización gradual de ML a medida que crece el poder computacional en dispositivos de todos los tamaños, llegan herramientas de código abierto y requerimientos más estrictos y la conciencia sobre la privacidad e información personalizada convergen para crear un ecosistema floreciente. Técnicas como **Machine Learning federado** permiten modelos que brindan privacidad a la información confidencial. El campo de **TinyML** permite que los modelos se ejecuten en dispositivos con recursos limitados, trasladando la inferencia hacia el borde, lo que libera recursos y mejora la privacidad de los datos confidenciales. Los **Feature Stores** brindan beneficios análogos al patrón de diseño para desarrollo de aplicaciones Modelo-Vista-Controlador, permitiendo una separación más clara de las preocupaciones entre la curación de datos, el entrenamiento del modelo y la inferencia. Los modelos disponibles públicamente, como **Stable Diffusion**, destacan tanto las increíbles capacidades de ML como las preocupaciones sobre los datos fuente y la ética. Los componentes de ML también son más fáciles de conectar, lo que hace posible crear experiencias y soluciones de ML con una composición creativa de modelos de negocio personalizados y modelos genéricos de alta capacidad. Aplaudimos las nuevas capacidades en este nuevo espacio y esperamos con ansias futuros avances.

El poder de las plataformas como producto

La palabra “plataforma” sigue siendo una de las más utilizadas en nuestras reuniones de Radar porque es un concepto muy presente en el sector. Se manifiesta de diferentes maneras, incluyendo plataformas centradas en el negocio o el dominio, pero también en infraestructura o experiencia de desarrollo. Fundamentalmente, la causa de muchos de los problemas y decepciones que las organizaciones experimentan con todas las plataformas es no tratarlas adecuadamente como productos. Por ejemplo, muchas plataformas destinadas a ser consumidas por los desarrolladores carecen de la investigación de usuario y el análisis contextual que esperaríamos en otro tipo de productos. Los propietarios de las plataformas deben validar sus supuestos o asunciones sobre las necesidades de los desarrolladores y responder a los patrones de uso reales. Y como todo buen producto, una plataforma necesita un apoyo continuado. Debe evolucionar y adaptarse en respuesta a las necesidades cambiantes del desarrollador. Además, roles como project managers y analistas de negocio suelen tener un alcance diferente al que tienen en aplicaciones tradicionales. La metáfora de la “plataforma como producto” sólo funciona cuando se adopta plenamente como una práctica y no como una frase que está de moda.

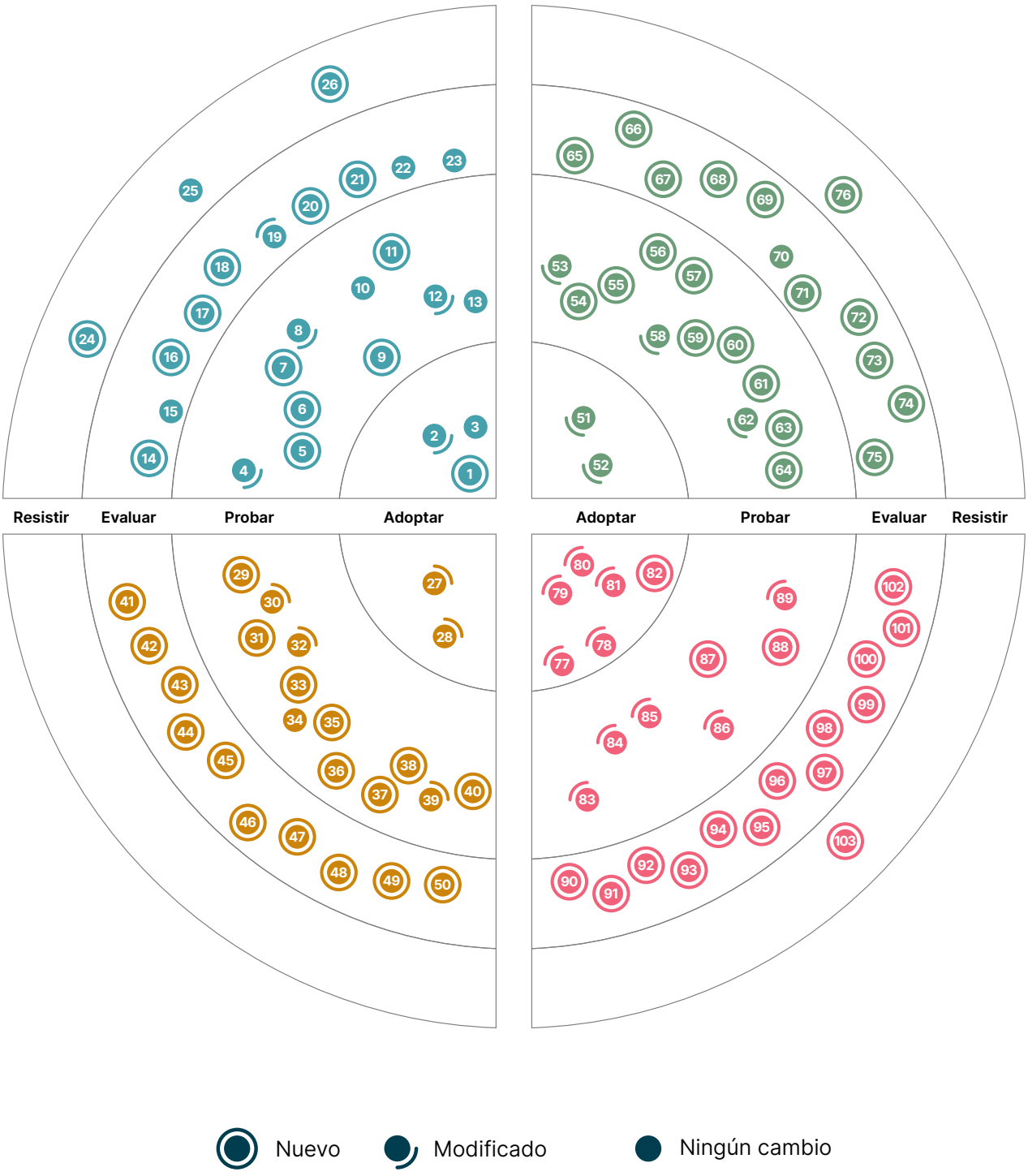
Descentralizando la propiedad de los datos

Como ya todas sabemos muy a nuestro pesar, la centralización de cualquier tipo abre la posibilidad de constricciones, cuellos de botella o exposiciones innecesarias. Debido a esto, estamos constantemente en la búsqueda de nuevas maneras de librarnos de esos puntos en dónde se produce acoplamiento debido a la centralización, algo que ya hemos destacado en varios blips de nuestro Radar. Basándonos en la investigación en tipos de datos replicados sin conflicto (del inglés: conflict-free replicated data types o CRDT), que permiten crear aplicaciones basadas en datos sin una base de datos centralizada, la técnica de software o aplicaciones basadas en enfoque local anima a desarrolladores a pensar en construir datos peer-to-peer en lugar de utilizar una base de datos centralizada. Descentralizar la propiedad de los datos, también permite a los desarrolladores aprovecharse del incremento en capacidades de sus dispositivos, tal y como se expone en el tema sobre la Integración de ML. Por ejemplo, muchas capacidades tales como el reconocimiento facial, pueden ocurrir en la periferia, dónde los datos necesarios para estas capacidades pueden permanecer almacenados en el dispositivo de manera indefinida.

El desarrollo móvil, también modular

Los ingenieros de software han aprendido el valor de estructurar la arquitectura de una aplicación principalmente alrededor de conceptos de dominio y funcionalidad de negocio. Asuntos técnicos –como la separación entre interfaz de usuario y lógica de dominio– son todavía importantes pero juegan un rol secundario. A medida que las aplicaciones móviles maduran, a menudo se hacen más grandes, convirtiéndose a veces en las también llamadas super aplicaciones, las cuales abarcan muchos servicios y pueden ser vistas como plataformas en sí mismas. Aplicaciones que no son tan grandes pero han ido adquiriendo muchas funcionalidades durante los años pueden habitualmente descomponerse también en módulos, y las compañías pueden ver que las aplicaciones móviles también se benefician del mismo enfoque de modularidad. Las aplicaciones modulares se prestan a ser desarrolladas por múltiples equipos autónomos, lo cual conlleva muchos beneficios bien documentados. Además de la complejidad debemos también contar con el requerimiento de desplegar a través de una “app store” y la necesidad de soportar versiones nativas para iOS y Android, así como una versión web, con pequeños cambios en cada una para acomodarse a cada plataforma. Observamos un mejor soporte en los frameworks para soportar las tensiones inherentes al desarrollo móvil, pero en general, a pesar de los beneficios, a muchas organizaciones se les hace difícil aplicar un enfoque modular al desarrollo móvil.

El Radar



El Radar

Ténicas

Adoptar

1. Mapeo de procesos hasta producción
2. Carga cognitiva del equipo
3. Modelado de Amenazas

Probar

4. BERT
5. Pruebas de regresión visual de componentes
6. Tokens de diseño
7. Servidor falso SMTP para probar envíos de correos electrónicos
8. Machine Learning federado
9. Plataforma de desarrollo incremental
10. Micro frontends para móvil
11. Observabilidad para los pipelines CI/CD
12. SLSA
13. Software Bill of Materials

Evaluar

14. Eficiencia de carbono como característica arquitectónica
15. CUPID
16. Protección del push de GitHub
17. Aplicación local-first
18. Tienda de métricas
19. Interfaz de Usuario orientada a servidor
20. Indicadores de Nivel Servicio (SLI) y Objetivos de Nivel de Servicio (SLO) como código
21. Datos sintéticos para modelos de prueba
22. TinyML
23. Credenciales verificables

Resistir

24. Trabajadores satélites sin ser “nativamente remotos”
25. SPA por defecto
26. Cloud native superficial

Plataformas

Adoptar

27. Backstage
28. Delta Lake

Probar

29. Servicio de Migración de Base de Datos AWS
30. Colima
31. Databricks Photon
32. DataHub
33. DataOps.live
34. eBPF
35. Feast
36. Monte Carlo
37. Retool
38. Seldon Core
39. Teleport
40. VictoriaMetrics

Evaluar

41. Bun
42. Unity Catalog en Databricks
43. Dragonfly
44. Edge Impulse
45. GCP Vertex AI
46. Gradient
47. IAM Roles Anywhere
48. Keptn
49. OpenMetadata
50. OrioleDB

Resistir

—

El Radar

Herramientas

Adoptar

51. Great Expectations
52. k6

Probar

53. Apache Superset
54. Bloqueo de la Bóveda de Respaldos en AWS (AWS Backup Vault Lock)
55. AWS Control Tower
56. Clumio Protect
57. Cruft
58. Excalidraw
59. Hadolint
60. Kaniko
61. Kusto Query Language
62. Spectral
63. Servicio de autorización declarativa de Styra
64. xbar para monitorizar compilaciones

Evaluar

65. Clasp
66. Databricks Overwatch
67. dbtvault
68. git-together
69. Harness Cloud Cost Management
70. Infracost
71. Karpenter
72. Mizu
73. Soda Core
74. Teller
75. Xcode Cloud

Resistir

76. Servicios en línea para formatear o analizar el código

Lenguajes y Frameworks

Adoptar

77. io-ts
78. Kotest
79. NestJS
80. React Query
81. Swift Package Manager
82. Yjs

Probar

83. Azure Bicep
84. Camunda
85. DSL en Kotlin para Gradle
86. Jetpack Media3
87. Ladle
88. Moshi
89. Svelte

Evaluar

90. Aleph.js
91. Astro
92. BentoML
93. Carbon Aware SDK
94. Cloudscape
95. Connect
96. SDK multi-dispositivo
97. Testing de componentes con Cypress
98. JobRunr
99. Million
100. Soketi
101. Stable Diffusion
102. Bóveda de Datos Sintéticos o Synthetic Data Vault

Resistir

103. Carbon

Técnicas

Adoptar

1. Mapeo de procesos hasta producción
2. Carga cognitiva del equipo
3. Modelado de Amenazas

Probar

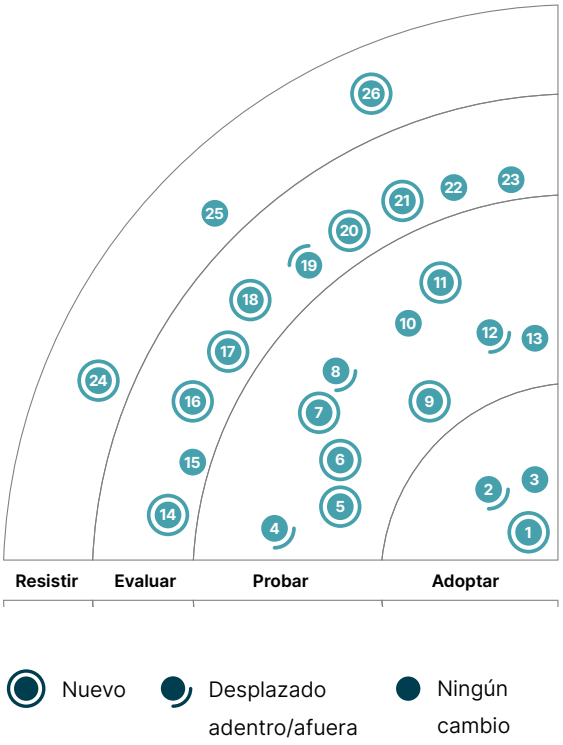
4. BERT
5. Pruebas de regresión visual de componentes
6. Tokens de Diseño
7. Servidor falso SMTP para probar envíos de correos electrónicos
8. Machine Learning federado
9. Plataforma de desarrollo incremental
10. Micro frontends para móvil
11. Observabilidad para los pipelines CI/CD
12. SLSA
13. Software Bill of Materials

Evaluar

14. Eficiencia de carbono como característica arquitectónica
15. CUPID
16. Protección del push de GitHub
17. Aplicación local-first
18. Tienda de Métricas
19. Interfaz de Usuario orientada a servidor
20. Indicadores de Nivel Servicio (SLI) y Objetivos de Nivel de Servicio (SLO) como código
21. Datos sintéticos para modelos de prueba
22. TinyML
23. Credenciales verificables

Resistri

24. Trabajadores satélites sin ser "nativamente remotos"
25. SPA por defecto
26. Cloud native superficial



1. Mapeo de procesos hasta producción

Adoptar

Aunque el mapeo de procesos hasta producción (path-to-production mapping) es una práctica casi universal en Thoughtworks desde la codificación de **Entrega continua**, a menudo nos encontramos con organizaciones que no están familiarizadas con esta práctica. Esta actividad se suele desarrollar en un workshop con un grupo multidisciplinar de personas – que incluye a todo aquel implicado en el diseño, el desarrollo, la publicación y el operación del software – en torno a una pizarra (o su equivalente virtual). En primer lugar, se enumeran en orden todos los pasos del proceso, desde el equipo del desarrollador hasta producción. A continuación, se lleva a cabo otra sesión para recabar más información y puntos débiles. La técnica más común que vemos se basa en el **Mapa del flujo de valor**, aunque existen múltiples variantes del **mapa de procesos** que son igualmente válidas. Esta actividad suele ser reveladora para la mayoría de los participantes, ya que identifica retrasos, riesgos e inconsistencias además de seguir utilizando la representación visual para la mejora continua del proceso de construcción y despliegue. Consideramos esta técnica tan fundamental que nos ha sorprendido descubrir que no la habíamos incluido con anterioridad en el radar.

2. Carga cognitiva del equipo

Adoptar

La interacción entre un equipo supone un concepto clave cuando rediseñamos una organización para mejorar la agilidad y velocidad empresarial. Estas interacciones quedan reflejadas en el software a desarrollar (ver **Ley de Conway**) e indican cuán efectivos pueden ser los equipos entregando valor a sus clientes de manera autónoma. Por ello, aconsejamos invertir tiempo y esfuerzo en establecer el diseño y la interacción de los equipos y, puesto que estas variables pueden evolucionar con el tiempo, es especialmente importante medir y hacer un seguimiento de la carga cognitiva del equipo , la cual indica cómo de fácil o difícil es para los equipos desarrollar, hacer pruebas y mantener sus servicios. Esta **plantilla** nos ha resultado útil para evaluar la carga cognitiva de los equipos, basada en las ideas de los autores del libro ***Team Topologies***.

Seguimos sorprendiéndonos del impacto positivo que tiene aplicar los conceptos de este libro al comunicarnos con clientes y rediseñar organizaciones. Los autores recomiendan un enfoque simple pero efectivo para diseñar organizaciones: consiste en identificar solo cuatro tipos de equipos y tres modos de interacción. Esto ayuda a reducir la ambigüedad dentro de la organización y proporciona un vocabulario común para los equipos, partes interesadas y el equipo de dirección para describir y diseñar el trabajo de un equipo. Para implementar este cambio de diseño organizacional, planteamos una estructura ideal de topología de equipo, aplicamos cualquier limitación técnica o de personal (por ejemplo, no hay suficientes empleados) y así extraemos la estructura definitiva. Esto nos permite poder aconsejar mejor a nuestros clientes y anticipar si estamos efectivamente mejorando la carga cognitiva al comparar la estructura actual con la definitiva.

3. Modelado de Amenazas

Adoptar

Seguimos recomendando que los equipos continúen haciendo uso del **modelado de amenazas** — un conjunto de técnicas que ayudan a identificar y clasificar amenazas potenciales durante el proceso de desarrollo — pero queremos enfatizar que esto no es una actividad a realizar únicamente al comienzo de los proyectos; los equipos deben evitar el **security sandwich**. Esto sucede porque durante todo el ciclo de vida de cualquier software, nuevas amenazas surgirán y las existentes continuarán evolucionando debido a eventos externos y constantes cambios de requisitos y arquitecturas. Esto significa que el modelado de amenazas se tiene que repetir de forma periódica — la frecuencia de repetición dependerá de las circunstancias y necesitará considerar factores como el coste de

realizar el ejercicio y el riesgo potencial para el negocio. En combinación con otras técnicas, como el establecimiento de requisitos de seguridad multifuncionales para manejar riesgos comunes de las tecnologías del proyecto y el uso de escáneres de seguridad automatizados, el modelado de amenazas puede ser un poderoso recurso.

4. BERT

Probar

Since we last talked about **BERT** (Bidirectional Encoder Representations from Transformers) en el Radar, nuestros equipos lo han utilizado exitosamente en unos cuantos proyectos de procesamiento del lenguaje natural (NLP por sus iniciales en inglés). En una de nuestras colaboraciones, observamos mejoras significativas al cambiar el tokenizador por defecto de BERT por un tokenizador word-piece entrenado por el dominio, para preguntas que contenían nombres de marcas o dimensiones. Aunque NLP tiene varios modelos nuevos de transformador, BERT es bien conocido, con una buena documentación y una vibrante comunidad, y continuamos considerándolo efectivo en un contexto de NLP empresarial.

5. Pruebas de regresión visual de componentes

Probar

Las pruebas de regresión visual son una herramienta útil y poderosa que debes tener siempre a mano, aunque tienen un coste significativo dado que se realizan para toda la página. Con el surgimiento de frameworks basados en componentes como **React** y **Vue**, también hemos sido testigos del surgimiento de las pruebas de regresión visual de componentes. Esta técnica logra un buen equilibrio entre valor y coste para asegurarnos de que no se añaden elementos visuales indeseados a la aplicación. En nuestra experiencia, las pruebas de regresión visual de componentes presentan menos falsos positivos y abogan por una buena arquitectura. Utilizar estas pruebas en conjunto con herramientas como **Vite** y la característica de webpack **Hot Module Replacement (HMR)**, puede ser visto como un cambio de paradigma para aplicar Desarrollo Dirigido por Pruebas (TDD) en proyectos de frontend.

6. Tokens de Diseño

Probar

Cuando nos enfrentamos al desafío de usar de forma consistente un **sistema de diseño** en varios factores de forma y plataformas, el equipo de Salesforce ideó el concepto de **Tokens de Diseño**. Los tokens almacenan valores, como colores y fuentes, de forma centralizada. Esto hace posible **separar las opciones de las decisiones**, y mejora significativamente la **colaboración entre equipos**. Los Tokens de Diseño no son nuevos, pero con la introducción de herramientas como **Tailwind CSS** y **Style Dictionary**, vemos que los Tokens de diseño están siendo utilizados con mayor frecuencia.

7. Servidor falso SMTP para probar envíos de correos electrónicos

Probar

Usar cuentas de correo electrónico de pruebas, así como servidores SMTP completos para pruebas (del inglés Simple Mail Transfer Protocol o Protocolo Simple para Transferencia de Correo electrónicos), siguen siendo prácticas habituales de pruebas de software. Sin embargo, el uso de un servidor real conlleva el riesgo de que **los correos electrónicos de prueba se acaben enviando a personas reales** y a menudo complica las pruebas de integración automatizadas. Hemos observado casos de éxito usando un servidor falso

SMTP para probar envíos de correos electrónicos, el cual registra una petición de envío de correo electrónico sin llegar a enviarlo realmente. Existen múltiples herramientas de código abierto en este ámbito, incluyendo **fake-smtp-server**, el cual representa correos electrónicos dentro de una interfaz de usuario web para pruebas visuales, y **mountebank**, que expone los correos electrónicos enviados a través de una API REST para realizar pruebas de integración. Recomendamos explorar esta técnica para reducir el riesgo y mejorar la eficiencia de las pruebas.

8. Machine Learning federado

Probar

Recientemente estamos observando proyectos de clientes que usan Machine Learning federado (ML). Tradicionalmente el entrenamiento de modelos de ML ha requerido que los datos se coloquen en una ubicación centralizada donde ejecutar el correspondiente algoritmo de entrenamiento. Esto es problemático desde el punto de vista de la privacidad, especialmente cuando los datos de entrenamiento contienen información confidencial o de identificación personal. Los usuarios pueden ser reacios a compartir datos o la legislación local de protección de datos nos puede impedir moverlos a una ubicación centralizada. El ML federado es una técnica descentralizada para entrenar un conjunto grande y diverso de datos que permite que éstos permanezcan remotos, por ejemplo, en el dispositivo de un usuario. El ancho de banda de la red y las limitaciones computacionales de los dispositivos aún presentan desafíos técnicos significativos, pero nos gusta la forma en que el ML federado deja a los usuarios en control de su propia información personal.

9. Plataforma de desarrollo incremental

Probar

Hemos estado escribiendo sobre plataformas para desarrolladores y cómo construirlas en prácticamente todas las ediciones del Radar desde 2017. Mientras tanto, el libro ***Team Topologies*** también ha hecho un gran trabajo al describir el ideal de una plataforma que apoya a los desarrolladores con "APIs de autoservicio, herramientas, servicios y conocimiento". Sin embargo, a menudo vemos equipos que se precipitan al perseguir esa visión de plataforma demasiado rápido. En su lugar, es clave la construcción de una plataforma de desarrollo incremental.Team Topologies recomienda esforzarse siempre por conseguir lo que ellos llaman la "Plataforma viable más fina" necesaria en cada etapa, donde la primera versión podría ser incluso sólo un conjunto de documentación en una wiki. El siguiente incremento podría aumentar el nivel de servicio al proporcionar plantillas o permitiendo a los equipos crear pull requests. Incrementos posteriores podrían entonces introducir APIs de autoservicio, pero sólo si aportan valor. En resumen, aunque hayamos advertido contra los **modelos operativos de plataforma orientado a tickets**, pasar de cero al autoservicio es el otro extremo. Hay que ir con calma, **trata tu plataforma como un producto** y constrúyela incrementalmente.

10. Micro frontends para móvil

Probar

Desde que se introdujeron en el Radar en 2016, hemos visto la adopción generalizada de **micro frontends** para las interfaces web. Recientemente, sin embargo, hemos visto proyectos que amplían este estilo arquitectónico para incluir también micro frontends para aplicaciones móviles. Cuando una aplicación se vuelve lo suficientemente grande y compleja, es necesario distribuir el desarrollo entre varios equipos. Esto presenta una serie de desafíos en torno a la autonomía del equipo, las estructuras de los repositorios y los frameworks de integración. En el pasado hemos

mencionado **Atlas y BeeHive**, pero estos marcos no lograron ganar tracción y ya no están en desarrollo activo. Otros enfoques más recientes son **Tuist** o el **Swift Package Manager** para integrar el trabajo de varios equipos en una sola aplicación. Pero, según nuestra experiencia, los equipos suelen acabar implementando su propio framework para la integración. Mientras que definitivamente vemos la necesidad de la modularidad en la ampliación de los equipos de desarrollo móvil, el caso de los micro frontends es menos seguro. Esto se debe a que, mientras que los micro frontends implican una correspondencia directa entre los equipos y las páginas o los componentes, esta estructura podría acabar desdibujando las responsabilidades de los contextos del dominio empresarial, aumentando así la **carga cognitiva del equipo**. Nuestro consejo es seguir los fundamentos de un diseño de aplicación correcto y limpio, abrazar la modularidad cuando se amplíe a varios equipos y adoptar una arquitectura de micro frontend sólo cuando los módulos y el dominio de negocio esten fuertemente alineados.

11. Observabilidad para los pipelines CI/CD

Probar

Las prácticas de observabilidad han trasladado la conversación de la monitorización de problemas bien entendidos a ayudar a localizar el origen de problemas desconocidos en sistemas distribuidos. Hemos visto casos de éxito sacando esa perspectiva fuera del entorno tradicional de producción al aplicar la observabilidad para los pipelines CI/CD para ayudar a optimizar cuellos de botella en el testing y los despliegues. Los pipelines complejos ofrecen fricción a los desarrolladores cuando su ejecución es muy lenta o no determinista, ralentizando el feedback y reduciendo la efectividad de los equipos. Además, su rol como infraestructura crítica de despliegue crea puntos de tensión durante períodos de despliegues rápidos, como les sucedió a diferentes organizaciones que respondían a la reciente vulnerabilidad de log4shell. El concepto de trazas se traslada bien a los pipelines: en lugar de capturar la cascada de llamadas a servicios, los spans hijos capturan información de cada fase de la construcción. Las mismas gráficas de cascadas usadas para analizar el flujo de llamadas en una arquitectura distribuida pueden ser efectivas para ayudarnos a identificar cuellos de botella en los pipelines, incluso las complejas con topologías fan-in y fan-out. Esto permite focalizar mucho mejor el esfuerzo en optimización. Aunque la técnica debería funcionar con cualquier herramienta de trazabilidad, **Honeycomb** soporta una herramienta llamada **buildevents** que ayuda a capturar la información de las trazas de los pipelines. Otra alternativa para abordar la captura de información ya expuesta de plataformas CI/CD, adoptada por la herramienta de código abierto **buildviz** (creada y mantenida por un Thoughtworker) permite investigación similares sin cambiar los pasos de configuración en sí.

12. SLSA

Probar

Así como el software sigue creciendo en complejidad, la cantidad de amenazas en las dependencias de software se vuelve cada vez más difícil de proteger. Los niveles de la cadena de suministro para artefactos de software, Supply chain Levels for Software Artifacts o **SLSA** (pronunciado “salsa”), es un conjunto de guías seleccionadas por un consorcio para que las organizaciones se protejan contra los ataques a la cadena de suministro, guías que evolucionaron gracias a la dirección interna que Google ha estado utilizando durante años. Apreciamos que SLSA no se comprometa con una “bala

de plata", es decir, un enfoque de solo herramientas para proteger la cadena de suministro, sino que proporciona una lista de verificación de amenazas y prácticas concretas a lo largo de un modelo de madurez. El **modelo de amenazas** es fácil de seguir con ejemplos de ataques del mundo real y los **requisitos** brindan orientación para ayudar a las organizaciones a priorizar acciones en función de los niveles de una robustez creciente para mejorar su posición de seguridad en la cadena de suministro. Desde que lo mencionamos por primera vez en el Radar, SLSA ha agregado más detalles sobre **certificaciones de software** con ejemplos para rastrear inquietudes como la **fuente de compilación**. Nuestros equipos han encontrado que SLSA logra un buen equilibrio entre asistencia de implementación y la conciencia de alto nivel sobre las amenazas alrededor de la cadena de suministro.

13. Software Bill of Materials

Probar

Con la presión continua para mantener los sistemas seguros y sin que se reduzca el panorama general de las amenazas, una Software Bill of Materials (SBOM) legible por una máquina puede ayudar a los equipos a mantenerse al tanto de los problemas de seguridad en las librerías de las que dependan. Desde que la **Executive Order** original fue publicada, la industria ha ganado claridad y entendimiento de lo que es SBOM y cómo crear uno; el Instituto Nacional de Estádares y Tecnología (INET), por ejemplo, ahora tiene más **consejos específicos** sobre como seguir con la norma. Hemos tenido experiencia en producción utilizando SBOMs en proyectos que van desde pequeñas compañías hasta en grandes multinacionales e incluso en departamentos gubernamentales, y estamos convencidos de que ofrecen un beneficio. Muchas organizaciones y gobiernos deberían considerar exigir SBOMs para el software que utilizan. La tecnica será fortalecida por nuevas herramientas que continuan apareciendo, como **Android Firebase BOM** que automáticamente alinea las dependencias de las librerías de una aplicación con las que aparecen listadas en la BOM.

14. Eficiencia de carbono como característica arquitectónica

Evaluar

La sustentabilidad es un tema que exige la atención de las empresas. En el ámbito del desarrollo de software su importancia ha aumentado, y ahora estamos viendo **diferentes formas** de abordar este tema. Por ejemplo, en cuanto a la huella de carbono del software de construcción, recomendamos evaluar la eficiencia del carbono como una característica arquitectónica. Una arquitectura que tiene en cuenta la eficiencia del carbono es aquella en la que las elecciones de diseño e infraestructura se han hecho para minimizar el consumo de energía, por lo tanto, las emisiones de carbono. Las herramientas de medición y el asesoramiento en este ámbito están madurando, lo que hace factible que los equipos consideren la eficiencia del carbono junto con otros factores como el rendimiento, la escalabilidad, el costo financiero y la seguridad. Como casi todo en la arquitectura de software, esto debería considerarse una compensación; nuestro consejo es pensar en esto como una característica adicional en todo un conjunto de **atributos de calidad relevantes** que son impulsados y priorizados por los objetivos de la organización y no dejados a un pequeño grupo de expertos para reflexionar de manera aislada.

15. CUPID

Evaluar

¿Cómo te planteas la escritura de un buen código? ¿Cómo puedes juzgar si has escrito código de calidad? Como desarrolladores de aplicaciones, siempre estamos buscando reglas, principios y patrones que podamos utilizar para compartir un lenguaje y unos valores a la hora de escribir código simple y fácil de modificar.

Daniel Terhorst-North ha hecho recientemente un nuevo intento para crear una lista de control para un buen código. Sostiene que, en lugar de ceñirse a un conjunto de reglas como **SOLID**, es más aplicable el uso de un conjunto de propiedades a las que aspirar. Ha ideado lo que llama las propiedades **CUPID** para describir lo que debemos hacer para conseguir un código "alegre": el código debe ser componible, seguir la filosofía Unix y ser predecible, idiomático y basado en el dominio.

16. Protección del push de GitHub

Evaluar

La publicación accidental de secretos parece ser un problema perenne, y herramientas como **Talisman** aparecen para ayudar con este asunto. Hasta ahora, los usuarios de GitHub Enterprise Cloud que tuvieran una Advanced Security License (licencia avanzada de seguridad) podían habilitar el escaneo de seguridad de sus cuentas, de forma que cualquier secreto (API keys, access tokens, credenciales, etc.) que fuera accidentalmente incluido en un commit al que se le haya hecho push, activaría una alerta. **Protección del push de GitHub** da un paso más adelantándose a una etapa anterior del workflow de desarrollo, al bloquear y evitar que los cambios pasen a un push si se detecta la presencia de secretos. Pese a que esto requiere ser configurado a nivel de organización y aplica sólo a aquellos que posean las licencias requeridas, siempre se agradecen protecciones extra a la hora de evitar publicar los secretos.

17. Aplicación local-first

Evaluar

En una aplicación centralizada, los datos del servidor son la única fuente de verdad: cualquier modificación de los datos debe pasar por el servidor. Los datos locales están subordinados a la versión del servidor. Esto parece una opción natural e inevitable para permitir la colaboración entre múltiples usuarios del software. La aplicación local-first, o **local-first software**, es un conjunto de principios que permite tanto la colaboración como la propiedad de los datos locales. Prioriza el uso del almacenamiento local y las redes locales sobre los servidores en centros de datos remotos o en la nube. Técnicas como los tipos de datos replicados sin conflictos (CRDTs por sus siglas en inglés) y las redes peer-to-peer (P2P) tienen el potencial de ser una tecnología fundacional para hacer realidad el software local-first.

18. Tienda de Métricas

Evaluar

Tienda de Métricas, a veces denominada como inteligencia empresarial "sin cabeza" (BI), es una capa que desvincula las definiciones de métricas de su uso en informes y visualizaciones. Tradicionalmente, las métricas eran definidas dentro del contexto de herramientas BI, pero este enfoque genera duplicación e incoherencias, ya que diferentes equipos los utilizan en diferentes contextos. Al desacoplar la definición en la tienda de métricas, obtenemos una reutilización clara y consistente en los informes de BI, visualizaciones e incluso analíticas integradas. Esta técnica no es nueva; por ejemplo, Airbnb introdujo Minerva hace un año. Sin embargo, ahora estamos viendo una tracción considerable en el ecosistema de datos y analítica con más herramientas apoyando con las tiendas de métricas fuera de la caja.

19. Interfaz de Usuario orientada a servidor

Evaluar

La interfaz de usuario orientada a servidor sigue siendo un tema candente de discusión en los círculos de aplicaciones móviles debido a que permiten a los desarrolladores aprovechar ciclos de cambio más rápidos sin contradecir las políticas de las tiendas de aplicaciones en torno a la revalidación de la propia aplicación. La interfaz de usuario orientada a servidor separa la representación (rendering) en un contenedor genérico en la aplicación móvil, de la definición de la estructura y los datos para cada vista, que es proporcionada por el servidor. Esto significa que los cambios que en su momento necesitaban pasar por la tienda de aplicaciones, pueden ahora ser logrados a través de cambios simples en las respuestas que envía el servidor. Mientras que algunos equipos muy grandes de desarrollo de aplicaciones móviles han logrado grandes resultados con esta técnica, también se requiere una inversión sustancial para construir y mantener su propio framework. Esta inversión requiere de un caso de negocio que la justifique. Mientras no se disponga de dicho caso de negocio, puede ser mejor proceder con precaución; de hecho, hemos podido experimentar planteamientos horrendos, y exageradamente configurables, que en realidad no ofrecían los beneficios prometidos.

Pero con el respaldo de gigantes como Airbnb y Lyft, puede ser que veamos emerger frameworks útiles que ayuden a domar dicha complejidad. Estemos atentos.

20. Indicadores de Nivel Servicio (SLI) y Objetivos de Nivel de Servicio (SLO) como código

Evaluar

Desde que Google popularizara por primera vez los Indicadores de Nivel de Servicio y los Objetivos de Nivel de Servicio (SLIs y SLOs respectivamente, por sus siglas en inglés) como parte de su práctica de Ingeniería de Confiabilidad de Sitios (SRE), las herramientas de observabilidad como Datadog, Honeycomb y Dynatrace comenzaron a incorporar la monitorización de SLO entre sus utilidades. OpenSLO es un estándar emergente que permite definir SLI y SLO en código, utilizando un lenguaje de especificación declarativo e independiente del proveedor, basado en el formato YAML utilizado por Kubernetes. Si bien el estándar aún es bastante nuevo, estamos viendo un impulso alentador, como la contribución de Sumo Logic con la herramienta slogen para monitorizar y generar dashboards.

Estamos entusiasmados con la promesa de versionar las definiciones de SLI y SLO en código y actualizar las herramientas de observabilidad desde los pipelines de integración y despliegue contínuo del propio servicio que se está desplegando.

21. Datos sintéticos para modelos de prueba

Evaluar

Durante nuestros debates para esta edición del Radar, surgieron varias herramientas y aplicaciones para la generación de datos sintéticos. A medida que las herramientas van madurando, hemos comprobado que el uso de datos sintéticos para modelos de prueba es una técnica potente y ampliamente útil. Aunque no pretenden sustituir a los datos reales a la hora de validar el poder de discriminación de los modelos de aprendizaje automático, los datos sintéticos pueden utilizarse en diversas situaciones. Por ejemplo, pueden usarse para evitar fallos catastróficos de los modelos en respuesta a sucesos que ocurren de forma excepcional o para testear las pipelines de datos sin exponer información personal identificable. Los datos sintéticos también son útiles para explorar casos límite que carecen de datos reales o para identificar el sesgo del modelo. Algunas herramientas útiles para generar datos son **Faker** o **Synth**, que generan datos que se ajustan a las propiedades estadísticas deseadas y herramientas como **Synthetic Data Vault** que pueden generar datos que imitan las propiedades de un conjunto de datos de referencia.

22. TinyML

Evaluar

Seguimos entusiasmados con la técnica **TinyML** y la posibilidad de crear modelos de aprendizaje automático (ML, machine learning) diseñados para ejecutarse en dispositivos móviles y de baja potencia. Hasta hace poco, la ejecución de un modelo de ML se ha considerado computacionalmente costosa y en algunos casos, requería hardware específico. Si bien la creación de los modelos todavía se encuadra generalmente en esta clasificación, ahora pueden crearse dichos modelos de forma que se puedan ejecutar en dispositivos pequeños, de bajo coste y bajo consumo. Si ha estado considerando la posibilidad de utilizar ML pero pensaba que no era realista debido a las limitaciones de capacidad de cómputo o de red, entonces vale la pena evaluar esta técnica.

23. Credenciales verificables

Evaluar

Cuando lo incluimos por primera vez en el Radar hace dos años, las credenciales verificables (CV) eran un estándar intrigante con algunas aplicaciones potenciales prometedoras, pero no eran ampliamente conocidas o comprendidas fuera de la comunidad de entusiastas. Esto era especialmente cierto cuando se trataba de las instituciones que otorgan credenciales, como los gobiernos estatales, que serían responsables de la aplicación de las normas. Dos años y una pandemia después, la demanda de credenciales electrónicas criptográficamente seguras, respetuosas con la privacidad y verificables por máquina ha crecido y, como resultado, los gobiernos están empezando a despertar al potencial de la CV. Ahora estamos empezando a ver cómo las CV aparece en nuestro trabajo para clientes del sector público. El **estándar del W3C** sitúa a los titulares de las credenciales en el centro, lo que es similar a nuestra experiencia cuando utilizamos credenciales físicas: los usuarios pueden poner sus credenciales verificables en sus propias carteras

digitales y mostrarlas a cualquiera en cualquier momento sin el permiso del emisor de las credenciales. Este enfoque descentralizado también permite a los usuarios gestionar mejor y revelar selectivamente su propia información, lo que mejora en gran medida la protección de la privacidad de los datos. Por ejemplo, gracias a la tecnología de prueba de conocimiento cero, puedes construir una credencial verificable para demostrar que eres mayor de edad sin revelar tu cumpleaños. Es importante señalar que, aunque muchas soluciones de **identidad descentralizada** se basan en la tecnología blockchain, ésta no es un requisito previo para todas las implementaciones de la CV.

24. Trabajadores satélites sin ser "nativamente remotos"

Resistir

El término "organización de equipos remotos" no sólo describe una estructura; sino que abarca múltiples **patrones y variantes**. Y muchos equipos han estado cambiando de patrones recientemente. Están saliendo del modo "todo el mundo siempre en remoto" al que les obligó una pandemia y pasando a un patrón de trabajadores satélites (a menudo rotativos), en el que parte del equipo está en un mismo lugar y otra parte está en remoto. Vemos que muchos de ellos no consideran correctamente lo que esto significa para sus formas de trabajo. Los trabajadores satélites sin formas de trabajo "nativamente remotas" son un retroceso hacia el privilegio de las prácticas co-localizadas. En una modalidad con trabajadores satélite, es importante seguir **utilizando procesos y enfoques "nativamente remotos" por defecto**. Por ejemplo, si la parte del equipo que se encuentra en el mismo lugar se une a una reunión, todos deben estar en sus computadoras portátiles individuales para participar en la colaboración digital o en el chat de la reunión. Los equipos deben ser conscientes del riesgo de excluir a sus trabajadores satélites y crear silos y sentimientos de exclusión. Si se sabe que siempre habrá al menos un miembro del equipo satélite, las formas de trabajo por defecto deberían asumirse como remotas.

25. SPA por defecto

Resistir

El predominio de los equipos que eligen una aplicación de una sola página (SPA en inglés) cuando necesitan crear un sitio web, continua. Seguimos preocupados que las personas no reconocen correctamente a los SPA como un estilo arquitectónico; en su lugar, saltan inmediatamente a la selección de un framework. Los SPA incurren en una complejidad que simplemente no existe con los sitios web tradicionales basados en servidores: problemas como la optimización de motores de búsqueda, administración de historial del navegador, análisis web y el tiempo de carga de la primera página, todos ellos deben ser tomados en cuenta. Se requiere un análisis adecuado y la consideración de las ventajas y desventajas para determinar si esa complejidad está justificada por el negocio o por la experiencia del usuario. A menudo los equipos se saltan el análisis de compensación, ciegamente aceptan la complejidad de los SPAs por defecto aún cuando las necesidades del negocio no lo justifiquen. Aún vemos algunos desarrolladores que no conocen un enfoque alternativo porque han pasado toda su carrera usando un framework como React. Creemos que muchos sitios web se benefician de la simplicidad de la lógica del lado del servidor y estamos alentados por técnicas como **Hotwire** que ayudan a cerrar esa brecha con la experiencia de usuario.

26. Cloud native superficial

Resistir

El término “cloud native” se usó originalmente para describir arquitecturas con características que aprovechaban al máximo el alojamiento en la nube pública. Ejemplos de las mismas incluyen arquitecturas distribuidas compuestas por muchos pequeños procesos, stateless y colaborativos, y sistemas con altos niveles de automatización en la construcción, prueba y despliegue de aplicaciones. Sin embargo, estamos observando una creciente tendencia hacia los diseños cloud native superficiales que simplemente utilizan una gran cantidad de servicios propietarios de un determinado proveedor de la nube y se detienen ahí, sin revisar la naturaleza fundamentalmente monolítica, frágil o laboriosa de la aplicación en cuestión. Es importante recordar que las funciones serverless por sí solas no hacen que una aplicación sea más resiliente o más fácil de mantener y que la nube nativa es realmente una cuestión de diseño más que un conjunto de opciones de implementación.

Plataformas

Adoptar

27. Backstage
28. Delta Lake

Probar

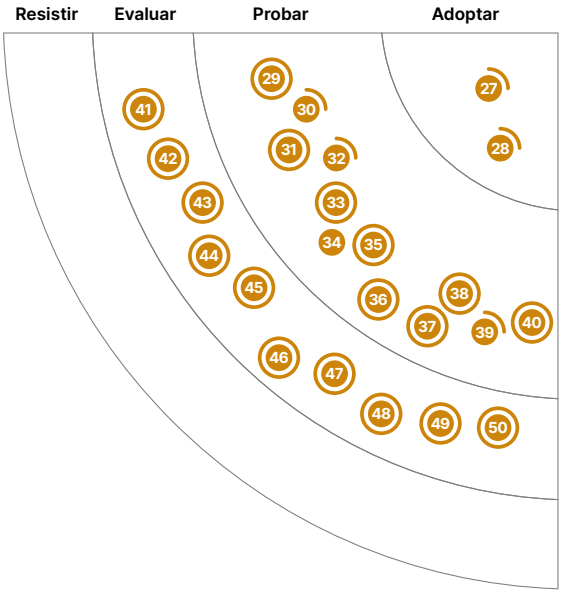
29. Servicio de Migración de Base de Datos AWS
30. Colima
31. Databricks Photon
32. DataHub
33. DataOps.live
34. eBPF
35. Feast
36. Monte Carlo
37. Retool
38. Seldon Core
39. Teleport
40. VictoriaMetrics

Evaluar

41. Bun
42. Unity Catalog en Databricks
43. Dragonfly
44. Edge Impulse
45. GCP Vertex AI
46. Gradient
47. IAM Roles Anywhere
48. Keptn
49. OpenMetadata
50. OrioleDB

Resistir

—



Nuevo

Desplazado adentro/afuera

Ningún cambio

27. Backstage

Adoptar

En un mundo cada vez más digital, mejorar la efectividad de los desarrolladores en las grandes organizaciones suele ser una preocupación central de los líderes sénior. Hemos visto bastante valor en los portales para desarrolladores en general y en **Backstage** en particular, por lo que nos gustaría recomendarlo en "Adopt". Backstage es una plataforma de código abierto para crear portales para desarrolladores creada por Spotify que mejora el descubrimiento de activos de software en toda la organización. Utiliza Markdown **TechDocs** almacenado junto con el código de cada servicio, lo que equilibra muy bien las necesidades de descubrimiento centralizado con la necesidad de propiedad distribuida de los activos. Backstage soporta plantillas de software para acelerar nuevos desarrollos y una arquitectura de plugins que permite aporta extensibilidad y adaptabilidad en el ecosistema de infraestructura de una organización. **Backstage Service Catalog** usa archivos YAML para rastrear la propiedad y los metadatos de todo el software en el ecosistema de una organización; incluso permite rastrear software SaaS de terceros, lo que generalmente requiere trazar en quién recae la propiedad de dichos servicios.

28. Delta Lake

Adoptar

Delta Lake es una **capa de almacenamiento de código abierto**, implementada por Databricks, que intenta incorporar transacciones ACID en el procesado de big data. En nuestros proyectos **data lake** o **data mesh** con Databricks, nuestros equipos prefieren usar almacenamiento Delta Lake a usar directamente tipos de almacenamiento de archivos tales como **AWS S3** o **ADLS**. Hasta hace poco, Delta Lake ha sido un producto cerrado propietario de Databricks, pero ahora es de código abierto y accesible a plataformas no Databricks. Sin embargo, nuestra recomendación de Delta Lake como opción por defecto ahora mismo se extiende solo a proyectos Databricks que usen formato de ficheros **Parquet** Delta Lake facilita los casos de uso de lectura/escritura concurrente de datos donde se requiere transaccionalidad a nivel de archivo. Encontramos de gran ayuda la fluida integración de Delta Lake con las APIS de Apache Spark **batch** y **micro-batch** especialmente en características tales como **viajar en el tiempo** (acceso a datos en un punto en concreto en el tiempo o revertir un commit) así como el soporte a la **evolución de esquemas** durante la escritura, aunque hay algunas limitaciones en dichas características.

29. AWS Database Migration Service

Probar

Servicio de Migración de Base de Datos AWS **AWS Database Migration Service** (DMS) para migrar datos a y desde AWS. En uno de nuestros proyectos de Transformación Digital, en la migración de datos desde Microsoft SQL Server a una instancia del Servicio de AWS de Base de Datos Relacional (RDS) PostgreSQL, logramos un paso a producción al nuevo sistema con casi cero tiempo de indisponibilidad. Este tipo de transformaciones involucran varias partes en movimiento que requieren planeación y coordinación a través de equipos multidisciplinarios, sin embargo para la migración de datos, estamos muy contentos con DMS. Automáticamente administra el despliegue, gestión y monitoreo de todos los recursos requeridos. Con el paso de los años, DMS ha madurado para soportar varias bases de datos **fuentes** y **destino** databases, and we continue to like it.

30. Colima

Probar

Colima se está convirtiendo en una popular alternativa abierta a Docker Desktop. Es capaz de aprovisionar el entorno de ejecución de contenedores Docker en una máquina virtual Lima, configurar la CLI de Docker en macOS, gestionar el redireccionamiento de puertos y el montaje de volúmenes. Colima utiliza containerd como entorno de ejecución, que es también el entorno de ejecución para la mayoría de los servicios gestionados por Kubernetes — mejorando la importante paridad entre entornos de desarrollo y producción. Con Colima puedes usar y probar fácilmente las últimas características de containerd, como la carga perezosa de imágenes de contenedores. Hemos tenido buenos resultados con Colima en nuestros proyectos. En el contexto de Kubernetes, también utilizamos nerdctl, una CLI compatible con Docker para containerd. Dado que Kubernetes considera obsoleto Docker como entorno de ejecución de contenedores y la mayoría de los servicios gestionados (EKS, GKE, etc) están siguiendo su ejemplo, más gente buscará herramientas nativas de containerd, de ahí la importancia de herramientas como nerdctl. En nuestra opinión, Colima se está dando cuenta de su fuerte potencial y se está convirtiendo en una de las alternativas de referencia a Docker Desktop.

31. Databricks Photon

Probar

Empezando con DataBricks 9.1 LTS (Long Term Support, Soporte a largo plazo), un nuevo tiempo de ejecución está disponible llamado Databricks Photon, una alternativa que fue reescrita desde cero en C++. Muchos de nuestros equipos han usado Photon en producción y han destacado las mejoras de rendimiento y los correspondientes ahorros de costes. Las mejoras y los cambios en costes van a depender de múltiples factores como el tamaño del conjunto de datos y los tipos de transacciones. Recomendamos hacer una prueba con una carga de trabajo realista para recopilar datos y poder comparar antes de tomar una decisión sobre el uso de Photon.

32. DataHub

Probar

Desde que mencionamos el descubrimiento de datos por primera vez en el Radar, LinkedIn ha evolucionadoWhereHows hacia DataHub, la plataforma de nueva generación para descubrimiento de datos vía un sistema extensible de metadatos. En lugar de rastrear y extraer metadatos, DataHub adopta un modelo basado en push, en el que los componentes individuales del ecosistema de datos publican metadatos mediante una API o un stream hacia la plataforma central. Este modelo de integración traslada la propiedad de la entidad central a cada uno de los equipos, haciéndolos responsables de sus metadatos. Como resultado, hemos utilizado DataHub con éxito como repositorio de metadatos a nivel de organización y punto de entrada de múltiples productos de datos mantenidos autónomamente. Cuando se toma este enfoque, hay que asegurarse de que sea ligero y evitar el peligroso camino que lleva al control centralizado sobre un recurso compartido.

33. DataOps.live

Probar

DataOps.live es una plataforma de datos que automatiza entornos en Snowflake. Inspirado en las prácticas DevOps DataOps.live permite tratar la plataforma de datos como cualquier otra plataforma web adoptando la integración continua y la entrega continua (CI/CD), las pruebas automatizadas, la

observabilidad y la gestión del código. Puede revertir cambios inmediatamente sin afectar a los datos o recuperarte de fallos completos y reconstruir un nuevo tenant de Snowflake en minutos u horas en lugar de días. Nuestros equipos han tenido buenas experiencias con DataOps.live, ya que nos ha permitido iterar rápidamente al construir productos de datos sobre Snowflake.

34. eBPF

Probar

Desde hace varios años, el kernel de Linux ha incluido la extended Berkeley Packet Filter (**eBPF**), una máquina virtual que proporciona la capacidad de incorporar filtros en sockets específicos. Pero eBPF va mucho más allá del filtro de paquetes y permite disparar la ejecución de scripts personalizados en distintos puntos dentro del kernel con muy poca sobrecarga. Al permitirte ejecutar programas en un sandbox dentro del kernel del sistema operativo, los desarrolladores de aplicaciones pueden ejecutar programas eBPF para incorporar nuevas capacidades al sistema operativo en tiempo de ejecución. Algunos de nuestros proyectos requieren hacer diagnóstico y análisis de problemas a nivel de llamadas de sistema. Un nivel al que nuestros equipos han encontrado que herramientas como **bcc** y **bpftrace** facilitan su trabajo. La observabilidad e infraestructura de redes también se benefician de eBPF — por ejemplo, el proyecto **Cilium** implementa balanceo de carga de tráfico y observabilidad sin **without sidecar overhead** en **Kubernetes**, y **Hubble** proporciona un extra de seguridad y observabilidad de tráfico adicional. El proyecto **Falco** usa eBPF para monitorización de seguridad y el proyecto **Katran** usa eBPF para crear un balanceo de carga L4 más eficiente. La comunidad de eBPF está creciendo rápidamente y vemos más y más sinergia en el campo de la observabilidad.

35. Feast

Probar

Feast es un **Feature Store** de código abierto para machine learning. Tiene varias propiedades útiles, las cuales incluyen la generación de grupos de atributos válidos en un punto en el tiempo — es decir los valores propensos a errores de atributos futuros no se filtran en los modelos durante el entrenamiento — y soporta tanto streaming como orígenes de datos por lotes. Sin embargo, actualmente solo soporta data estructurada fechada, por lo cual puede que no sea apropiado si trabajas con data desestructurada en tus modelos. Nosotros hemos usado exitosamente Feast a significativa escala como almacén offline durante el proceso de entrenamiento del modelo y como almacén online durante la predicción.

36. Monte Carlo

Probar

Monte Carlo es una plataforma para la observabilidad de datos. Usando modelos de machine learning, infiere y aprende sobre los datos, identificando problemas y notificando a los usuarios cuando estos surgen. Permite a nuestros equipos mantener la calidad de los datos a través de los pipelines ETL, data lakes, data warehouses y reportes de business intelligence (BI). Con funciones tales como tableros de monitoreo como código, un catalogo central de datos y linaje a nivel de campos, nuestros equipos ven a Monte Carlo como una herramienta incalculable para data governance en general.

37. Retool

Probar

En ediciones anteriores, hemos dado como recomendación evaluar las plataformas delimitadas low-code acomo metodo para aplicar soluciones de bajo código a específicos casos de uso dominios muy limitados. Hemos visto algo de tracción en este campo, específicamente con Retool, una plataforma low-code que nuestros equipos utilizan para construir soluciones para usuarios internos, sobre todo para consultar y visualizar los datos. Les permite producir mas rápidamente soluciones de solo lectura las cuales no sean críticas para el negocio. Las principales ventajas de Retool son sus componentes de UI y su capacidad de ser integrada fácil y rápidamente con fuentes de datos habituales.

38. Seldon Core

Probar

Seldon Core es una plataforma open-source para empaquetar, desplegar, monitorear y manejar modelos de machine learning en clusters de Kubernetes Con soportes instantáneos para varios frameworks de machine learning, puedes fácilmente convertir a contenedores tus modelos usando servidores de inferencia pre-empaquetados, servidores de inferencia personalizados o language wrappers. Con tracking distribuido mediante Jaeger y vía Alibi, Seldon Core aborda varios desafíos de delivery de última-milla con implementaciones de machine learning, y a nuestros equipos de Data les encanta.

39. Teleport

Probar

Teleport es una herramienta de zero trust en el acceso de red a la infraestructura. Las configuraciones tradicionales requieren políticas complejas o servidores de salto para restringir el acceso a recursos críticos. Teleport, sin embargo, simplifica esto con un panel de acceso unificado y controles de autorización con granularidad fina que reemplazan a los servidores de salto, VPNs o credenciales compartidas. Implementado como un simple ejecutable con soporte nativo para múltiples protocolos (incluyendo SSH, RDP, Kubernetes API, MySQL, MongoDB y PostgreSQL), Teleport facilita la configuración y la administración de accesos seguros en entornos Linux, Windows o Kubernetes. Desde que lo mencionamos por primera vez en el Radar, algunos equipos han usado Teleport y nuestra experiencia positiva nos impulsó a destacarlo.

40. VictoriaMetrics

Probar

La observabilidad de hoy depende de recoger y sumar un conjunto exhaustivo de métricas granulares para poder entender, predecir y analizar el comportamiento de un sistema. Sin embargo, cuando se aplica a un sistema en la nube nativo compuesto por muchos procesos y servidores redundantes y que cooperan entre ellos, la cardinalidad (o número de series únicas en tiempo) se vuelve poco manejable porque crece exponencialmente con cada nuevo servicio, contenedor, nodo, clúster, etc. Por eso, para gestionar datos de alta cardinalidad, hemos comprobado que VictoriaMetrics es una buena opción. VictoriaMetrics es especialmente útil para trabajar con arquitecturas de microservicios alojadas en Kubernetes, y el operador de VictoriaMetrics facilita que los equipos puedan implementar su propia monitorización de manera personalizada. También nos agrada su arquitectura de componentes y habilidad para continuar recogiendo métricas incluso cuando el servidor principal

no está disponible. Aunque nuestro equipo está contento con VictoriaMetrics, es un sector que está evolucionando rápidamente, así que recomendamos echar un vistazo a otras bases de datos de serie temporal de alto rendimiento compatibles con **Prometheus** como **Cortex** o **Thanos**.

41. Bun

Evaluar

Bun es un nuevo entorno de tiempo de ejecución de JavaScript, similar a **Node.js** o **Deno**. Sin embargo, a diferencia de Node.js o Deno, Bun se crea utilizando JavaScriptCore de WebKit en lugar del motor V8 de Chrome. Diseñado como reemplazo directo de Node.js, Bun es un único binario (escrito en **Zig**) que actúa como un empaquetador, transpilador y administrador de paquetes para aplicaciones JavaScript y **TypeScript** Bun se encuentra actualmente en versión beta, así que es posible encontrar errores o problemas de compatibilidad con algunas bibliotecas de Node.js. Sin embargo, se ha construido desde cero con varias optimizaciones, incluido un inicio rápido y una mejor representación del lado del servidor, y creemos que vale la pena evaluarlo.

42. Unity Catalog en Databricks

Evaluar

Databricks Unity Catalog es una solución de gobierno de datos para activos tales como archivos, tablas, o modelos de aprendizaje automático en un **lakehouse**. Aunque se encontrarán varias plataformas en el espacio de gobierno de datos, si ya se usan otras soluciones de Databricks, entonces se debería evaluar Unity Catalog. Se quiere resaltar que aunque estas plataformas de gobierno generalmente implementan una solución centralizada para mejorar la consistencia entre los espacios y las cargas de trabajo, la responsabilidad de gobernar debería ser federada hacia equipos individuales permitiéndoles gobiernen sus propios activos.

43. Dragonfly

Evaluar

Dragonfly Es un nuevo almacén de datos en memoria compatible con las APIS de **Redis** y Memcached. Aprovecha la nueva API específica de Linux **io_uring** para E/S e implementa **novedosos algoritmos y estructuras de datos** sobre una arquitectura multi hilo de tipo shared-nothing. Gracias a estas inteligentes elecciones en la implementación, Dragonfly logra resultados impresionantes en cuanto a rendimiento. Aunque Redis sigue siendo nuestra opción predeterminada para las soluciones de almacenamiento de datos en memoria, creemos que Dragonfly es una opción interesante para evaluar.

44. Edge Impulse

Evaluar

En versiones anteriores del Radar, hemos escrito sobre **TinyML** — la práctica de ejecutar modelos entrenados en dispositivos pequeños con sensores a bordo para tomar decisiones o extraer funcionalidades sin necesidad de que la información tenga que ir y volver de la nube. **Edge Impulse**

ha simplificado al máximo el proceso de recopilación de datos de sensórica y posterior entrenamiento y despliegue de un modelo. Edge Impulse es una plataforma hospedada end-to-end para el desarrollo de modelos optimizados para ser ejecutados en dispositivos edge pequeños como microcontroladores. La plataforma guía al desarrollador a través de todo el pipeline, incluyendo las tareas de recolección y etiquetado de los datos de entrenamiento. Lo han puesto fácil para empezar utilizando tu teléfono móvil tanto para la recolección de datos como para la ejecución del clasificador mientras que el entrenamiento y ajuste del modelo se realizan en el entorno alojado en la nube, que es más potente. Los algoritmos de reconocimiento resultantes también pueden optimizarse, compilarse y cargarse en una amplia gama de arquitecturas de microcontroladores. Aunque Edge Impulse es una propuesta comercial, la plataforma es gratis para desarrolladores y hacen que todo el proceso sea divertido y atractivo incluso para aquellas personas que son nuevas en el área del machine learning. La baja barrera de entrada para crear una aplicación que funcione significa que veremos más dispositivos edge con capacidad integrada de toma de decisiones.

45. GCP Vertex AI

Evaluar

GCP Vertex AI es una plataforma de inteligencia artificial unificada que permite a los equipos crear, implementar y escalar modelos de aprendizaje automático (ML por sus siglas en inglés). Vertex AI incluye modelos preentrenados, que pueden ser utilizados directamente, afinados o combinados con AutoML, e incluso con infraestructura como feature stores y pipelines para modelos de aprendizaje automático. Nos gustan las capacidades integradas de Vertex AI, que contribuyen a percibir esta plataforma como una plataforma de IA coherente.

46. Gradient

Evaluar

Gradient es una plataforma para crear, hacer el deploy y correr aplicaciones de machine-learning muy similar a Google's Colab. Los notebooks pueden ser creados desde templates, ayudándote a comenzar el trabajo de PyTorch o TensorFlow o a través de aplicaciones como Stable Diffusion. En nuestra experiencia, Gradient es ideal para modelos con un uso intenso de GPU, y nos agrada que el ambiente basado en web sea persistente.

47. IAM Roles Anywhere

Evaluar

IAM Roles Anywhere es un nuevo servicio de AWS que te permite obtener credenciales de seguridad temporales en IAM para cargas de trabajo como servidores, contenedores y aplicaciones que se ejecutan fuera de AWS. Nos resulta particularmente útil en configuraciones de nube híbrida donde las cargas de trabajo se dividen entre recursos de AWS y recursos ajenos a AWS. En lugar de crear credenciales de larga duración, con IAM Roles Anywhere, ahora puedes crear credenciales de corta duración para acceder a los recursos de AWS mediante certificados X.509. Creemos que este enfoque agiliza el patrón de acceso a lo largo de la nube híbrida y te recomendamos que lo pruebes.

48. Keptn

Evaluar

Keptn es un panel de control para operaciones y entrega basado en CloudEvents para la instrumentación. Como una de las técnicas que mencionamos en observabilidad para pipelines de CI/CD, Keptn visualiza la orquestación en forma de trazas. La definición declarativa del pipeline tiene como objetivo independizar los objetivos de los SREs de su implementación, apoyándose en otra instrumentación de observabilidad, integración y despliegue para poder responder a ciertos eventos. Estamos particularmente entusiasmados con la idea de agregar verificaciones de Objetivos de Nivel de Servicio (SLOs) en forma de fitness functions de arquitectura a los pipelines de CI/CD: Keptn te permite definir Indicadores de Nivel Servicio (SLIs) en forma de pares de clave-valor, donde el valor representa la consulta a tu infraestructura de observabilidad. Luego se evaluará el resultado contra los Objetivos de Nivel de Servicio (SLOs) definidos como quality gates (pruebas de calidad). Keptn adopta el mismo enfoque para las operaciones automatizadas, lo que por ejemplo permite definir de forma declarativa una especificación para escalar un ReplicaSet en respuesta a una degradación del tiempo de respuesta promedio. Creado por Dynatrace, Keptn también se integra con Prometheus y Datadog.

49. OpenMetadata

Evaluar

Indudablemente, la capacidad de decubrir los datos se ha convertido en un punto central muy importante para las empresas ya que es un facilitador para que los datos sean compartidos y utilizados eficientemente por diferentes personas. Hemos incluido plataformas como DataHub y Collibra en ediciones anteriores del Radar. Sin embargo, nuestros equipos están evaluando constantemente opciones en este espacio y recientemente han mostrado interés en OpenMetadata, una plataforma dedicada a la gestión de metadatos utilizando estándares abiertos. A nuestros equipos les gusta esta plataforma de código abierto porque mejora la experiencia de desarrollo gracias a su arquitectura simple, fácil despliegue con un enfoque en la automatización y un fuerte enfoque en la capacidad de descubrir los datos.

50. OrioleDB

Evaluar

OrioleDB es un nuevo motor de almacenamiento para PostgreSQL. Nuestros equipos utilizan mucho PostgreSQL, pero su motor de almacenamiento fue diseñado originalmente para discos duros. Aunque hay múltiples opciones para ajustarlo al hardware moderno, puede ser difícil y engorroso conseguir resultados óptimos. OrioleDB aborda estos retos implementando un motor de almacenamiento nativo de la nube con soporte específico para unidades de estado sólido (SSD) y memoria de acceso aleatorio no volátil (NVRAM). Para probar el nuevo motor, instale primero los parches de mejora de los actuales métodos de acceso a tablas y luego instale OrioleDB como una extensión de PostgreSQL. Creemos que OrioleDB tiene un gran potencial para resolver una serie de problemas pendientes desde hace tiempo en PostgreSQL, y le animamos a que lo evalúe cuidadosamente.

Herramientas

Adoptar

51. Great Expectations
52. k6

Probar

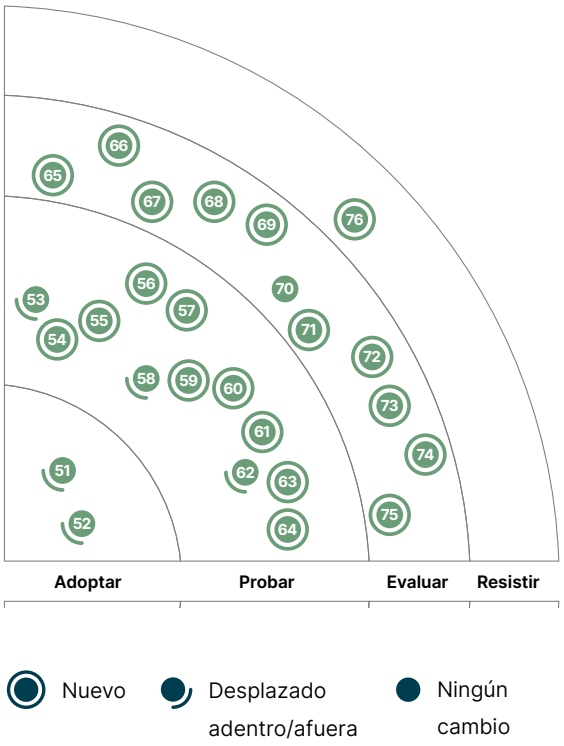
53. Apache Superset
54. Bloqueo de la Bóveda de Respaldos en AWS (AWS Backup Vault Lock)
55. AWS Control Tower
56. Clumio Protect
57. Cruft
58. Excalidraw
59. Hadolint
60. Kaniko
61. Kusto Query Language
62. Spectral
63. Servicio de autorización declarativa de Styra
64. xbar para monitorizar compilaciones

Evaluar

65. Clasp
66. Databricks Overwatch
67. dbtvault
68. git-together
69. Harness Cloud Cost Management
70. Infracost
71. Karpenter
72. Mizu
73. Soda Core
74. Teller
75. Xcode Cloud

Resistir

76. Servicios en línea para formatear o analizar el código



51. Great Expectations

Adoptar

Great Expectations se ha convertido en una prácticas por default para nuestros equipos en el ámbito de la calidad de datos, es por esto que recomendamos su adopción, no sólo por la falta de mejores alternativas, sino también porque nuestros equipos han obtenido grandes resultados en varios proyectos de clientes. Great Expectations es un framework que permite elaborar controles integrados para identificar anomalías o problemas de calidad en los pipelines de datos. Al igual que las pruebas unitarias que se ejecutan en un pipeline de construcción, Great Expectations realiza validaciones durante la ejecución de un proceso de datos. Nos gusta su simplicidad y facilidad de uso — las reglas almacenadas en JSON pueden ser modificadas por nuestros expertos en el dominio de datos sin necesidad de tener conocimientos en ingeniería de datos.

52. k6

Adoptar

Desde que lo mencionamos por primera vez en el Radar, k6 se ha convertido en una herramienta de referencia para las pruebas de rendimiento. Seguimos siendo fanáticos de lo fácil que es escribir código JavaScript para pruebas, pero k6 también tiene un creador de pruebas low-code para que jugar con la herramienta sea aún más fácil. La documentación describe lo fácil que es agregar pruebas de rendimiento a una pipeline a través de múltiples herramientas de CI/CD. A nuestros equipos les resulta fácil integrar herramientas de visualización como Grafana y New Relic, que les ayudan a poner a punto tanto la infraestructura como las aplicaciones. La facilidad de uso para los desarrolladores y su ecosistema hacen de k6 una opción convincente para investigar el comportamiento de un sistema bajo una carga elevada.

53. Apache Superset

Probar

Apache Superset es una gran herramienta de inteligencia de negocio (BI) para la exploración y visualización de datos para trabajar con grandes configuraciones de lagos de datos y almacenes de datos. Admite varias data sources — incluyendo AWS Redshift, BigQuery, Azure MS SQL, Snowflake y ClickHouse. Además, no es necesario ser un ingeniero de datos para usarlo; está destinado a beneficiar a todos los ingenieros que exploran datos en su trabajo diario. Para casos de uso exigentes, nos resultó fácil escalar Superset al implementarlo en un clúster de Kubernetes Desde la última vez que hablamos de ello en el Radar, Superset se ha graduado como un producto de Apache y hemos visto un gran éxito en varios proyectos.

54. Bloqueo de la Bóveda de Respaldos en AWS (AWS Backup Vault Lock)

Probar

Al implementar una recuperación de desastres robusta, segura y confiable, es necesario asegurarse que los respaldos (backups) no puedan ser eliminados o modificados antes que caduquen, ya sea de manera malintencionada o accidental. Anteriormente, con AWS Backup, estas políticas y garantías tenían que ser implementadas a mano. Recientemente, AWS añadió la funcionalidad de Vault Lock para garantizar que los respaldos (backups) sean inmutables e inviolables. AWS Backup Vault Lock aplica políticas de retención y eliminación, y evita que aún aquellos con privilegios de administrador puedan modificar o eliminar archivos de respaldos (backups). Esto ha demostrado tener un valor adicional y cubre una brecha existente.

55. AWS Control Tower

Probar

La gestión de cuentas de varios equipos es un desafío en AWS, especialmente en la configuración y la gobernanza; **AWS Control Tower** intenta abordar este desafío. Nuestro equipo ha reportado buenos resultados al utilizarla para gestionar las cuentas y el control de acceso de varios equipos de la organización a través de un lugar único y centralizado.

56. Clumio Protect

Probar

Tuvimos éxito con **Clumio Protect** para realizar respaldos de datos de AWS, en particular S3. Clumio Protect, una solución SaaS comercial, también puede realizar copias de seguridad de una variedad de otros servicios de AWS y almacenar los datos offline donde no se puede acceder a ellos a través de Internet. Nuestros equipos responsables de manejar la protección y la recuperación de datos a gran escala descubrieron que Clumio Protect es fácil de configurar y mantener y supera con creces al servicio nativo de respaldos de AWS cuando los repositorios S3 son particularmente grandes.

57. Cruft

Probar

Llevamos hablando de **plantillas de servicio a medida** desde que identificamos por primera vez el concepto de los **microservicios** Si una organización se propone crear una colección de pequeños servicios que se pueden desarrollar, construir, implementar y operar de forma independiente pero consistente, tiene sentido brindar a los equipos un punto de partida sólido que se alinee con el estándar. Sin embargo, uno de los problemas persistentes con ese enfoque es que, a medida que la plantilla evoluciona con el tiempo en respuesta a los cambiantes requisitos técnicos y comerciales, los proyectos basados en versiones anteriores de la plantilla quedan obsoletos. Aplicar retroactivamente las mejoras de la plantilla a un proyecto ya establecido se convierte en un reto. **Cruft** intenta abordar este problema proporcionando herramientas para identificar y parchear las diferencias entre un proyecto local y el estado actual de un repositorio de plantillas maestras. Combina el motor de plantillas **Cookiecutter** con los hashes de git para identificar y aplicar cambios a las plantillas. Piense en ello como un administrador de paquetes para un modelo de proyecto tipo. Mantener las plantillas actualizadas es un problema notoriamente difícil y persistente en el tiempo, por lo que para nosotros la solución que ofrece Cruft parece casi demasiado buena para ser verdad. Según los primeros comentarios de nuestro equipo, Cruft realmente funciona y facilita la vida tanto de las personas que desarrollan como de las que mantienen los servicios. Estamos impacientes por ver cómo funciona a largo plazo, pero por ahora vale la pena echarle un vistazo a esta herramienta potencialmente útil.

58. Excalidraw

Probar

Seguimos recibiendo de nuestros equipos informes entusiastas sobre **Excalidraw** , pero nuestra advertencia anterior sobre la seguridad sigue latente. Excalidraw es una herramienta en línea para dibujar, simple pero potente. A veces los equipos solo necesitan una imagen rápida en lugar de un diagrama formal; para equipos remotos, Excalidraw provee una forma rápida de crear y compartir

diagramas. A nuestro equipos también les agrada el aspecto “lo-fi” (“baja-fidelidad”) de los diagramas que se pueden hacer. Una reminiscencia de los diagramas que hubieran hecho en una pizarra si estuvieran trabajando en el mismo lugar físico. Respecto a la seguridad, en el momento de escribir este blip, cualquier persona que tenga el enlace puede ver los diagramas; cabe recalcar, sin embargo, que la versión de pago de Excalidraw provee más autenticación y opciones para ejecutarse en un servidor local si es necesario.

59. Hadolint

Probar

Nos gusta correr la voz acerca de las herramientas de linting que realmente ayudan a encontrar problemas en lugar de solo disputas de estilo abreviado en el equipo. **Hadolint** es una de esas herramientas: ayuda a encontrar problemas comunes en Dockerfiles. Creemos que es rápido, preciso y con buena documentación. Explica ambos, cómo solucionar un problema y por qué es un problema en primer lugar, empujando así a los autores de Dockerfile hacia las buenas prácticas. De paso, Hadolint está construido sobre **ShellCheck**, que recomendamos por derecho propio para verificar sus shell scripts.

60. Kaniko

Probar

La mayoría de las herramientas y plataformas actuales para pipelines CI/CD están construidas sobre contenedores como entorno de ejecución. Varios de nuestros equipos están usando **Kaniko** para construir imágenes de contenedores desde dentro de esas pipelines basadas en contenedores. Esto forma parte de una tendencia de dejar atrás **Docker** como el estándar de facto para los entornos de ejecución de contenedores. Con Kaniko puedes construir tus imágenes sin usar un demonio Docker. Esto ayuda a evitar problemas de seguridad del modo “privilegiado” de Docker, que sería necesario para cualquier actividad “Docker-en-Docker”. Además, no tienes que asumir a priori que tu pipeline tiene acceso al demonio de Docker, lo cual ya no puede darse por sentado y a menudo requiere configuración extra.

61. Kusto Query Language

Probar

A medida que trabajar con datos se vuelve cada vez más común, seguimos viendo herramientas que intentan potenciar el lenguaje SQL; **Kusto Query Language** (KQL) es una de ellas. KQL fue creado por Azure, y viene a aportar modularidad, encapsulamiento, componibilidad, reusabilidad, extensibilidad y dinamismo a las consultas relacionales. Nuestros equipos aprecian bastante su interactividad: puedes unir una consulta al operador de renderizado y ver un gráfico de manera instantánea. Puedes también agrupar estos gráficos en tableros y, a partir de los registros, obtener información para los ejecutivos en minutos. Aunque el lenguaje KQL actualmente está limitado al **Explorador de datos de Azure**, anticipamos que la tendencia de potenciar SQL para lograr una mejor operabilidad de datos no se detendrá.

62. Spectral

Probar

Spectral es un linter de JSON/YAML con énfasis en las especificaciones OpenAPI y AsyncAPI. Incluye un amplio conjunto de reglas para dichas especificaciones, que pueden evitar dolores de cabeza a los desarrolladores al diseñar e implementar APIs o colaboración orientada a eventos. Estas reglas comprueban que las especificaciones de los parámetros de la API sean correctas o la existencia de una declaración de licencia en la especificación, entre otras cosas. El CLI facilita incorporar Spectral tanto en el desarrollo local como en los pipelines de CI/CD y la API JavaScript soporta casos de uso más avanzados. El sitio de GitHub enlaza a conjuntos de reglas del mundo real disponibles de manera pública por compañías como Adidas, ayudando a que los equipos puedan adoptar sus propias reglas de linting.

63. Servicio de autorización declarativa de Styra

Probar

Servicio de autorización declarativa de Styra (DAS) es una herramienta de governanza y automatización para gestionar Open Policy Agent (OPA) a escala. Creado por los fundadores de OPA, esta herramienta nos permite desplegar políticas a través de "sistemas" incluyendo clusters de Kubernetes, repositorios de código de infraestructura, namespaces y más. No obstante, lo más importante es que nos permite hacer un análisis en tiempo real de las decisiones hechas por un agente OPA, junto con funcionalidades para reproducir cambios en las políticas con fines de depuración e investigación de escenarios de tipo "what-if". También cuenta con un registro de auditoría que ayuda a los equipos de seguridad con informes históricos.

64. xbar para monitorizar compilaciones

Probar

En equipos remotos, existe una dolorosa carencia de monitores de compilación dedicados Desafortunadamente, las nuevas herramientas de integración continua (CI) carecen de soporte para el viejo formato de CCTray Como consecuencia, las compilaciones fallidas no siempre se identifican tan rápido como nos gustaría. Para resolver el problema, muchos de nuestros equipos han comenzado a usar xbar para monitorizar compilaciones. Con xbar, se puede ejecutar un script para conocer el estado de la compilación, mostrándolo en la barra de menú. Además se puede programar para realizar un seguimiento de otras métricas del equipo como credenciales pendientes de expirar o cuánto se retrasa la publicación de una versión en producción respecto a la de pruebas de aceptación. Por supuesto, xbar tiene un propósito más general, pero resuelve un problema inmediato y emergente causado por el trabajo en remoto. Rumps, entre otras herramientas, puede resolver el mismo problema.

65. Clasp

Evaluar

Desafortunadamente, una gran parte del mundo sigue utilizando hojas de cálculo y lo continuará haciendo. Estas son ideales ya que permiten a cualquier persona construir esas pequeñas herramientas adaptadas a sus necesidades más particulares. Sin embargo, cuando se quiere mejorarlas aplicando cierto nivel de lógica entonces necesitan código “real”, y el que las hojas de cálculo sean por naturaleza de bajo código se transforma en una limitante. Si eres parte de una empresa que, como Thouhgtworks, que utiliza la G-Suite de Google, **Clasp** te permite implementar al menos algunas prácticas de **Entrega Continua (CD)** en los scripts de estas aplicaciones. También puedes escribir el código fuera de tu proyecto, te permite crear opciones para realizar pruebas, control de código fuente y construir pipelines, e incluso permite el uso de **TypeScript**. Clasp tiene algún tiempo entre nosotros, y aunque no debes esperar que sea un ambiente de programación con todas las comodidades que usualmente tenemos, ciertamente puede ayudar de forma muy grata a mejorar la experiencia de uso de la secuencia de comandos de estas aplicaciones.

66. Databricks Overwatch

Evaluar

Databricks Overwatch es un proyecto de Databricks Labs que habilita a los equipos a analizar varias métricas operacionales de cargas de trabajo de Databricks en relación al costo, gobierno y desempeño, soportados por la posibilidad de ejecutar experimentos del tipo qué pasa si. Esencialmente es un conjunto de pipelines de datos que pueblan tablas en Databricks, que luego puede ser analizado utilizando herramientas como notebooks. Overwatch es una herramienta muy fuerte, sin embargo, está aún en sus primeras fases por lo que puede tomar cierto esfuerzo para configurarla — nuestro uso de esta herramienta requirió de arquitectos de solución de Databricks para ayudarnos a configurar y poblar una tabla de referencia de precios para cálculos de costos — esperamos, sin embargo, que su adopción se vuelva más fácil en el tiempo. El nivel de análisis que Overwatch hace posible, es más profundo que el permitido por las herramientas de análisis de costo por proveedores de nube. Por ejemplo, pudimos analizar el costo de fallas de jobs — reconociendo que fallar tempranamente ahorra dinero , comparado con los jobs que fallan sólo cerca de los pasos finales — así como descomponer el costo en varios tipos de agrupamientos.

67. dbtvault

Evaluar

Data Vault 2.0 es una metodología de modelado de datos y un patrón de diseño cuya intención es mejorar la flexibilidad de los almacenes de datos (data warehouses), comparado con otros enfoques de modelado populares. Data Vault 2.0 se puede aplicar a cualquier almacén de datos como **Snowflake** o **Databricks**. Al implementar almacenes de Data Vault, hemos descubierto que el paquete **dbtvault** es una herramienta muy útil para **dbt** dbtvault proporciona un conjunto de plantillas **jinja** que generan y ejecutan los scripts ETL necesarios para poblar un almacén de Data Vault. Aunque dbtvault tiene algunas mejoras pendientes — carece de soporte para forzar unicidad implícita o realizar cargas incrementales — en general, cubre un nicho de mercado y requiere una configuración mínima para comenzar.

68. git-together

Evaluar

Siempre estamos buscando maneras de eliminar fricciones cuando programamos en pares y es por eso que estamos muy emocionadas con **git-together**: una herramienta programada en Rust que simplifica la manera de atribuir un commit de git durante la programación en pares. Utilizando `git` como alias de `git-together`, permitimos que esta herramienta pueda añadir extensiones a `git config` que capturan la información de las personas que hacen commit utilizando las iniciales de éstas como alias. Para cambiar de par, o pasar a programar sola o a hacer mob programming, requiere que se utilice el comando `git with` seguido de las iniciales del par, por ejemplo `git with bb cc`, permitiendo que después se pueda continuar con el flujo de trabajo habitual. Cada vez que se haga un commit, git-together hará rotación de la pareja designada como autora que git tiene guardada, y añadirá automáticamente a las otras autoras al final del mensaje que se hará commit. La configuración se puede añadir al repositorio, permitiendo de esta manera que git-together ya funcione después de haber clonado el repositorio.

69. Harness Cloud Cost Management

Evaluar

Harness Cloud Cost Management es una herramienta comercial que funciona con los tres proveedores de nube principales y sus clústeres de **Kubernetes** para ayudar a visualizar y gestionar costes de nube. Este producto calcula un indicador de eficiencia de costes en base a los recursos inactivos y aquellos que no tienen ninguna carga de trabajo asignada, y usa tendencias históricas para ayudar a optimizar la asignación de recursos. Los paneles de control resaltan picos de coste y permiten al usuario registrar anomalías inesperadas, que posteriormente alimentan el algoritmo de aprendizaje que mejora la detección de las mismas. El gestor de costes de nube puede recomendar ajustes a los límites de memoria y uso del CPU, con la opción de optimizar para coste o para rendimiento. Las “perspectivas” permiten agrupar costes basados en filtros definidos por la organización (que podrían ser unidades de negocio, equipos o productos) y automatizar el envío de informes para aumentar la visibilidad del gasto de la nube. Creemos que este gestor de costes de la nube ofrece una gama de funcionalidades interesantes para que las organizaciones maduren sus prácticas de FinOps.

70. Infracost

Evaluar

Continuamos viendo organizaciones que se mueven a la nube sin comprender correctamente cómo van a realizar el seguimiento de los gastos en los que van a incurrir. Anteriormente mencionamos **coste de ejecución como fitness function de la arquitectura**, e **Infracost** es una herramienta que tiene como objetivo mostrar las diferencias de coste de la nube en los pull request de Terraform. Es un software de código abierto y está disponible para macOS, Linux, Windows y Docker. Soporta de forma nativa los precios de AWS, GCP y Microsoft Azure. También proporciona una API pública en la que se pueden consultar los datos actuales de costes. Seguimos entusiasmados con su potencial, especialmente cuando se trata de obtener una mejor visibilidad de costes en el IDE.

71. Karpenter

Evaluar

Una de las características fundamentales de Kubernetes es su habilidad para lanzar automáticamente nuevos pods cuando se necesita capacidad adicional y apagarlos cuando las cargas disminuyen. Este autoescalado horizontal es una característica muy útil, pero solo puede funcionar si los nodos necesarios para alojar los pods ya existen. Mientras que Cluster Autoscaler puede realizar una expansión rudimentaria de clusters desencadenada por fallos de pods, este tiene una flexibilidad limitada. Sin embargo, es un autoescalador de código abierto Kubernetes Operator con más inteligencia incorporada: Analiza las cargas de trabajo actuales y las restricciones de programación de los pods para seleccionar automáticamente un tipo de instancia adecuada para luego iniciarla o detenerla según sea necesario. Karpenter es un operador en el alma de herramientas como Crossplane que puede aprovisionar recursos en la nube fuera del clúster. Karpenter es un atractivo compañero para el servicio de autoescalado automático en la nube, que los proveedores de servicios en la nube proporcionan de forma nativa en sus Kubernetes gestionados. Por ejemplo, AWS ahora soporta Karpenter como una alternativa de primera clase en su servicio EKS Cluster Autoscaler.

72. Mizu

Evaluar

Mizu es una API de visualización de tráfico para Kubernetes. A diferencia de otras herramientas, Mizu no requiere de instrumentación o modificación de código. Funciona como un DaemonSet que inyecta un contenedor a nivel de nodo de tu clúster de Kubernetes y ejecuta operaciones del tipo tcpdump. Encontramos esta herramienta muy útil como depurador, ya que puede observar todas las comunicaciones de API a través de múltiples protocolos (REST, gRPC, Kafka, AMQP y Redis) en tiempo real.

73. Soda Core

Evaluar

Soda Core es una herramienta open-source de observabilidad y calidad de datos. Hemos hablado sobre Great Expectations previamente en el Radar, y Soda Core es una alternativa con una diferencia clave — las validaciones de datos se definen en un DSL llamado SodaCL (previamente conocido como Soda SQL) en contraposición a las funciones de Python. Una vez las validaciones han sido escritas, pueden ejecutarse como parte de un data pipeline o ser planificadas para ejecutarse programáticamente. Dada la tendencia actual de la orientación al dato, mantener la calidad de los mismos es una tarea crítica, por lo que les animamos a evaluar Soda Core.

74. Teller

Evaluar

Teller es un administrador universal de secretos de código abierto para desarrolladores que garantiza que se establezcan las variables de entorno correctas al iniciar una aplicación. No obstante, no es un vault en sí: es una herramienta CLI que se integra con diversas fuentes, que van desde proveedores de secretos en la nube hasta soluciones de terceros como HashiCorp Vault, pasando por archivos de entorno locales. Teller tiene funcionalidades adicionales para escanear tu código en busca de

secretos presentes en tus vaults, para limpiar de secretos tus logs, para detectar desviaciones entre proveedores de secretos y para sincronizarlos entre ellos. Dada la sensibilidad de acceder a secretos, no podemos enfatizar lo suficiente la necesidad de securizar la cadena de suministro para dependencias de código abierto, así como valoramos lo fácil que es usar la CLI en entornos de desarrollo locales, pipelines de CI/CD y automatización de despliegues.

75. Xcode Cloud

Evaluar

Xcode Cloud es una herramienta de CI/CD integrada en Xcode y utilizada para desarrollar, probar y desplegar aplicaciones de Apple. Proporciona una experiencia integrada con herramientas familiares para los desarrolladores de Apple como Xcode, App Store Connect y TestFlight. Según la experiencia de nuestros equipos, hace un buen trabajo a la hora de simplificar la configuración del pipeline y el aprovisionamiento de perfiles y certificados. Esta herramienta es bastante reciente y la mayoría de nuestros equipos de desarrollo móvil siguen utilizando la más madura **Bitrise**. Aun así, creemos que merece la pena evaluarla y seguir su evolución.

76. Servicios en línea para formatear o analizar el código

Resistir

Anteriormente nos hemos referido a los **datos de producción en entornos de prueba** y ahora queremos señalar otra práctica común que debe ser abordada con cuidado o incluso detenida por completo: Servicios en línea para formatear o analizar el código. Hay muchos sitios útiles para formatear o analizar formatos como JSON y YAML, así como sitios que evalúan tutoriales de código o producen métricas de código en línea. Se debe tener extremo cuidado al utilizarlos. Pegar un bloque de JavaScript, JSON o similar en un sitio web desconocido puede fácilmente crear incidentes de seguridad o privacidad, y podría exportar, sin que uno lo sepa, datos personales a una jurisdicción diferente. Estos sitios nunca deben ser utilizados con datos en producción y deben ser abordados con precaución en cualquier otra circunstancia.

Lenguajes y Frameworks

Adoptar

77. io-ts
78. Kotest
79. NestJS
80. React Query
81. Swift Package Manager
82. Yjs

Probar

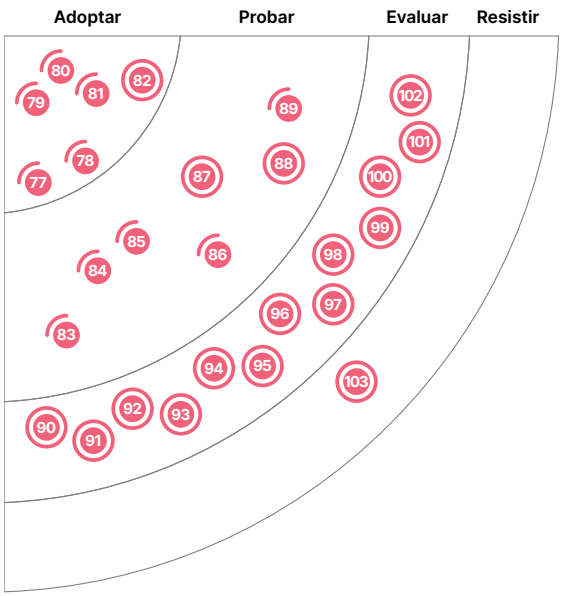
83. Azure Bicep
84. Camunda
85. DSL en Kotlin para Gradle
86. Jetpack Media3
87. Ladle
88. Moshi
89. Svelte

Evaluar

90. Aleph.js
91. Astro
92. BentoML
93. Carbon Aware SDK
94. Cloudscape
95. Connect
96. SDK multi-dispositivo
97. Testing de componentes con Cypress
98. JobRunr
99. Million
100. Soketi
101. Stable Diffusion
102. Bóveda de Datos Sintéticos o Synthetic Data Vault

Resistir

103. Carbon



Nuevo

Desplazado adentro/afuera

Ningún cambio

77. io-ts

Adoptar

Nuestros equipos que desarrollan en TypeScript están encontrando io-ts inestimable, en especial al interactuar con APIs que en última instancia crean objetos de tipos específicos. Cuando se trabaja con TypeScript, obtener datos dentro de los límites del sistema de tipos (es decir, de las APIs anteriormente mencionadas) puede conducir a errores en tiempo de ejecución que pueden ser difíciles de encontrar y depurar. io-ts cierra la brecha entre la verificación de tipos en tiempo de compilación y el consumo de datos externos en tiempo de ejecución al proveer funciones de codificación y decodificación. Dadas las experiencias de nuestros equipos y la elegancia de su enfoque, creemos que vale la pena adoptar io-ts.

78. Kotest

Adoptar

Kotest (previamente KotlinTest) es una herramienta de testing independiente para el ecosistema Kotlin la cual es ampliamente usada entre nuestros equipos en múltiples implementaciones de Kotlin - nativo, JVM o JavaScript. Sus ventajas clave residen en que ofrece una amplia variedad de estilos de testing a fin de estructurar test suites, así como un exhaustivo conjunto de matchers, que permiten llevar a cabo tests expresivos en un elegante DSL interno. Además de su soporte a testing basado en propiedades testing, a nuestros equipos les encanta el sólido plugin de IntelliJ y su comunidad de soporte. Muchos de nuestros desarrolladores lo consideran su primera opción y recomiendan migrar a Kotest a aquellos que siguen usando JUnit en Kotlin.

79. NestJS

Adoptar

En el pasado, hemos avisado acerca de la sobrecarga de Node, y todavia somos cautelosos sobre las razones para escogerlo. Sin embargo, en escenarios adecuados donde Node.js sea requerido para construir aplicaciones back-end, nuestros equipos reportan que NestJS Es una buena opción para permitir que los desarrolladores creen aplicaciones empresariales testeables, escalables, poco acopladas y fáciles de mantener. NestJS es un framework TypeScript que inicialmente hace que el desarrollo de aplicaciones Node.js sea mas seguro y menos proclive a errores. NestJS es dogmático y viene con los principios SOLID y una arquitectura inspirada en Angular por defecto.

80. React Query

Adoptar

React Query es frecuentemente conocida como la biblioteca de consulta de datos que faltaba para React. Un requisito habitual en muchas aplicaciones React es la consulta, sincronización, almacenamiento en caché y actualización del estado del servidor y a pesar de que dichos requisitos están claros, lograr la implementación correcta es notablemente difícil. React Query ofrece una sencilla solución utilizando hooks (ganchos), que trabajan mano a mano con librerías de consulta asíncrona como axios, Fetch y GraphQL dado que todas están construidas en base a promesas. Como desarrollador de aplicaciones, simplemente basta con pasar una función que resuelve los datos y dejar todo lo demás al framework (marco de trabajo). Nos gusta que si bien ofrece múltiples opciones de configuración, éstas no son necesarias para hacerla funcionar. Las herramientas

de desarrollador, si bien aún no están disponibles para **React Native**, también ayudan a los desarrolladores nuevos a entender cómo funciona el framework. Para React Native pueden utilizarse **plugins para las herramientas de desarrollo** de terceros como **Flipper**. En nuestra experiencia, la versión 3 de React Query ha traído la estabilidad necesaria para ser usada en producción por nuestros clientes.

81. Swift Package Manager

Adoptar

Cuando se introdujo en 2014 Swift no incluía administrador de paquetes. Más tarde **Swift Package Manager** fue creado como un proyecto de código abierto oficial de Apple y desde entonces ha continuado desarrollándose y madurando. Nuestros equipos confían cada vez más en SwiftPM debido a que la mayoría de los paquetes pueden ser incluídos a través de este, y los procesos tanto para creadores como consumidores de paquetes se han agilizado. En el anterior Radar recomendamos probarlo, pero ahora creemos que tiene sentido seleccionarlo por defecto para comenzar nuevos proyectos. Para proyectos existentes que utilicen herramientas como CocoaPods or **Carthage**, puede merecer la pena hacer un experimento rápido para estimar el nivel de esfuerzo requerido para migrar y comprobar si todas las dependencias están disponibles.

82. Yjs

Adoptar

Se ha demostrado que los algoritmos de tipo de datos replicados libres de conflictos (CRDT, conflict-free replicated data type) son capaces de distribuir y fusionar automáticamente los cambios entre pares sin conflictos. Pero en la práctica, incluso para datos lo suficientemente pequeños, estos algoritmos generalmente requieren una cantidad significativa de memoria para rastrear todos los cambios realizados por diferentes pares, lo que los hace poco prácticos. **Yjs** es una implementación CRDT cuidadosamente optimizada que mantiene el consumo de memoria a un nivel razonable para grandes conjuntos de datos y millones de modificaciones. También proporciona bindings para editores de texto populares, lo que reduce en gran medida el coste de crear herramientas colaborativas.

83. Azure Bicep

Probar

Para quienes prefieren un lenguaje más natural que JSON para el código de la infraestructura, **Azure Bicep** es un lenguaje específico del dominio (DSL) que utiliza una sintaxis declarativa y admite plantillas parametrizadas reutilizables para definiciones de recursos modulares. Una **Visual Studio Code extensión** proporciona seguridad de tipo instantánea, intellisense y comprobación de sintaxis, mientras que el compilador permite la transpilación bidireccional hacia y desde plantillas de Azure Resource Manager (ARM). El DSL orientado a los recursos de Bicep y la integración nativa con el ecosistema de Azure lo convierten en una opción convincente para el desarrollo de la infraestructura de Azure.

84. Camunda

Probar

Desde la última vez que lo mencionamos **Camunda**, hemos visto que muchos de nuestros equipos y clientes usan la plataforma, lo que la convierte en uno de nuestros motores de flujo de trabajo preferidos en los casos en que un motor de flujo de trabajo es un buen ajuste para el dominio. Camunda ofrece motores de flujo de trabajo y de decisión que pueden ser integradas como una librería a tu código Java. Esto facilita los flujos de trabajo de pruebas, versionamiento y la refactorización, aliviando algunas de las desventajas de otros motores de flujo de trabajo de bajo código. Incluso hemos visto a Camunda ser utilizada en entornos con requisitos de alto rendimiento. A los equipos también les gusta lo fácil que es integrarse con **Spring Boot** y su agradable interfaz de usuario.

85. DSL en Kotlin para Gradle

Probar

Anteriormente ya emitimos un blip sobre el DSL (lenguaje específico de dominio) en Kotlin para el plugin de Gradle para Android, o lenguaje específico de dominio en Kotlin para Gradle (Gradle Kotlin DSL, en inglés), que añadía soporte para scripts en **Kotlin** como alternativa a **Groovy** para proyectos Android que utilizan scripts de construcción de **Gradle** El objetivo de reemplazar Groovy con Kotlin es proporcionar un mejor soporte para el refactoring, una edición más simple en IDEs y finalmente producir código que sea más fácil de leer y mantener. Para equipos que ya estén usando Kotlin, también significa trabajar en la construcción en un lenguaje ya conocido. Sugerimos ahora probar el DSL en Kotlin como un lenguaje alternativo a Groovy para proyectos de Gradle en general, especialmente si tenemos scripts de construcción de Gradle que sean grandes y complejos. Muchos IDEs incluyen ya soporte para la migración de proyectos existentes. Sugerimos comprobar previamente en la **documentación** los detalles más actualizados, incluyendo los prerrequisitos. Uno de nuestros equipos que tenía un script de construcción de al menos 7 años de antigüedad y 450 líneas de código, consiguió migrarlo exitosamente en unos pocos días.

86. Jetpack Media3

Probar

Android ha tenido múltiples APIs para multimedia: Jetpack Media, también conocido como MediaCompat, Jetpack Media2 y ExoPlayer. Desafortunadamente esas librerías fueron desarrolladas de forma independiente, con diferentes objetivos, pero con funcionalidades solapadas. Los desarrolladores Android no sólo han tenido que elegir qué librería usar sino que también han tenido que programar adaptadores o conectores cuando se requerían funcionalidades de varias APIs. **Jetpack Media3** es una API que toma áreas de funcionalidad común de esas APIs existentes — incluyendo UI, reproducción y gestión de sesiones multimedia — y se combinan en una API agregada y refinada. La interfaz del reproductor de ExoPlayer también ha sido actualizada, mejorada y simplificada para funcionar como la interfaz de reproducción común de Media3. Tras una fase inicial de acceso limitado, ahora Media3 está en beta. Aunque su primera versión está aún por llegar, ya hemos tenido experiencias positivas usándola en aplicaciones.

87. Ladle

Probar

A medida que **Storybook** fue creciendo en popularidad, se fue convirtiendo cada vez más en un gigante. Si lo que realmente te interesa es aislar y probar tus componentes UI en React, entonces **Ladle** es la alternativa. Ladle es compatible con la mayoría de las API de Storybook (los archivos MDX aún no son compatibles) y puede ser usado como un reemplazo directo. Es liviano y tiene una mejor integración con **Vite**. También provee API simples y limpias que se pueden integrar fácilmente con otros frameworks de prueba.

88. Moshi

Probar

Hemos escuchado que nuestros equipos basados en **Kotlin**-están buscando alternativas a los frameworks de Java como GSON cuando manejan JSON. Apesar de que existe desde hace un tiempo, **Moshi** se ha convertido recientemente en uno de los frameworks preferidos para muchos de estos equipos. Es fácil migrar desde GSON y Moshi brinda soporte nativo para parámetros predeterminados y de tipo que no aceptan valores nulos. Moshi hace que trabajar con JSON sea más rápido y sencillo. Si actualmente estás usando un framework de Java con Kotlin para manejar JSON, te recomendamos probar Moshi.

89. Svelte

Probar

Entre los frameworks de componentes web, **Svelte** se destaca al mover la reactividad fuera del navegador, hacia dentro del compilador. En lugar de optimizar las actualizaciones DOM usando un DOM virtual y trucos de optimización del navegador, Svelte compila tu código dentro de un código vainilla JavaScript framework-less que, quirúrgicamente, actualiza el DOM directamente. Esto, adicional a los beneficios de rendimiento en tiempo de funcionamiento, también permite a Svelte optimizar la cantidad de código que el navegador tiene que descargar sin tener que sacrificar herramientas para los desarrolladores; de hecho, está comprobado que es eficiente y compatible con ahorro de batería para aplicaciones web móviles debido a que menos código tiene que ejecutarse en el propio navegador. Dejando de lado los beneficios de eficiencia, nuestros equipos han apreciado la amistosa curva de aprendizaje y los beneficios de mantenimiento que vienen de **escribir menos código**. Svelte por sí mismo es solamente el framework de componente, pero **SvelteKit** añade herramientas para construir aplicaciones web completas.

90. Aleph.js

Evaluar

Ciertamente no hay una escasez de frameworks para construir aplicaciones web en JavaScript/**TypeScript**. Hemos presentado varios de ellos en el Radar, pero lo que hace destacar a **Aleph.js** sobre el resto en este grupo tan nutrido, es que está diseñado para ser ejecutado en **Deno**, el nuevo entorno de ejecución del lado del servidor creado por el desarrollador original de **Node**. Esto sitúa a Aleph.js sobre una base moderna que aborda varios de los defectos y problemas con Node. Aleph.js es aún nuevo — se acerca el lanzamiento de su versión 1.0 en este momento — pero ya ofrece una sólida experiencia de desarrollo, incluyendo el reemplazo de módulos en caliente. Estando Deno actualmente muy por delante de su **versión 1.0**, Aleph.js es una opción moderna para aquellos proyectos que puedan asumir el riesgo.

91. Astro

Evaluar

Es difícil de creer pero en 2022, la comunidad de desarrolladores sigue produciendo nuevos e interesantes frameworks para la creación de aplicaciones web. **Astro** es un nuevo framework, de código abierto, para aplicaciones multipágina que renderiza HTML en el servidor y minimiza la cantidad de JavaScript transferido. Astro parece especialmente adecuado para sitios web orientados a albergar contenidos que se nutren de múltiples fuentes. Nos gusta el hecho de que, aunque Astro fomenta el envío sólo de HTML, también permite -cuando es necesario- seleccionar componentes activos escritos en el framework front-end JavaScript de tu elección. Esto es posible gracias a su **arquitectura de islas**. Estas islas son regiones interactivas dentro de una misma página donde el JavaScript necesario para su funcionamiento sólo se descarga cuando se necesita. Astro es relativamente nuevo pero parece estar dando soporte a un ecosistema creciente de desarrolladores y código. Hay que seguir supervisando su desarrollo.

92. BentoML

Evaluar

BentoML es un framework basado en python para servir modelos de aprendizaje automático en entornos de producción escalables. Los modelos que provee son agnósticos al entorno; todos los artefactos, código y dependencias son encapsuladas en un formato auto-contenido llamado Bento. Es como tener tu modelo "como un servicio". Piensa en BentoML como **Docker** para modelos de aprendizaje automático: genera imágenes virtuales con APIs pre-programadas listas para ser implementadas e incluye funcionalidades que facilitan la prueba de estas imágenes. BentoML puede ayudar a acelerar el esfuerzo de desarrollo inicial al facilitar el inicio de los proyectos, razón por la cual lo incluimos en Assess.

93. Carbon Aware SDK

Evaluar

Cuando se trata de reducir la huella de carbono de una aplicación — las emisiones de dióxido de carbono causadas indirectamente por el funcionamiento del software — la atención suele dirigirse a hacer que el software sea más eficiente. La idea es clara: un software más eficiente necesita menos electricidad y menos servidores, reduciendo las emisiones de la generación de electricidad y la fabricación de los servidores. Una estrategia adicional es hacer que la aplicación sea carbon aware. Esto se debe a que la misma carga de trabajo no siempre tiene la misma huella de carbono. Por ejemplo, cuando la ejecución se realiza en un centro de datos con un clima más frío, se requiere menos energía para el aire acondicionado; o, cuando la ejecución se realiza en un momento en que hay más energía renovable disponible (más luz solar, vientos más fuertes), se necesita menos electricidad procedente de fuentes basadas en el carbono. Con el **Carbon Aware SDK**, los ingenieros de software pueden consultar determinadas fuentes de datos para encontrar opciones menos intensivas en carbono para una determinada carga de trabajo y a continuación, trasladarla a otra ubicación o ejecutarla en otro momento. Esto tiene sentido para grandes cargas de trabajo que no son sensibles al tiempo ni a la latencia, como el entrenamiento de un modelo de machine-learning. Aunque el SDK y las fuentes de datos disponibles aún no son muy completos, creemos que es hora de empezar a estudiar cómo podemos hacer que nuestros sistemas sean conscientes del carbono.

94. Cloudscape

Evaluar

Cloudscape es un sistema de diseño de código abierto que no sólo tiene un completo set de componentes, también tiene 35 patrones de interacción y representación de contenido. Además utiliza tokens de diseño para aplicar temas y provee de wrappers de elementos para todos los componentes, lo que simplifica notablemente el testeo unitario. Esto lo hace destacar sobre otros sistemas de diseño que existen.

95. Connect

Evaluar

Connect es una familia de librerías para crear APIs HTTP compatibles con navegadores y gRPC. De manera similar a gRPC, escribes un esquema de Protocol Buffer, implementas la lógica de la aplicación, y Connect genera código para manejar la serialización, enrutamiento, compresión y negociación del tipo de contenido. Sin embargo, Connect intenta mejorar gRPC de varias maneras. Esto incluye soporte nativo para gRPC-Web sin un proxy de traducción; interoperabilidad con enrutadores o middleware de terceros, porque connect-go está construido sobre net/http (a diferencia de grpc-go); y genera clientes con tipado seguro y con la ergonomía del código hecho a mano. En su mayoría preferimos REST, y no somos muy fanáticos del enfoque RPC para crear APIs. Dicho lo cual, Connect parece abordar algunas de nuestras preocupaciones con los RPC, y te alentamos a que lo evalúes.

96. SDK multi-dispositivo

Evaluar

A medida que los dispositivos inteligentes siguen introduciéndose en nuestras vidas, empezamos a ver que surgen nuevos casos de uso en múltiples dispositivos. El ejemplo clásico es un texto que empezamos a leer en un teléfono pero preferimos terminar en una tablet. Otros ejemplos son el trazado de una ruta ciclista en una computadora y luego transferir los datos a una computadora para bicicletas para una navegación más fácil o utilizar un teléfono móvil como cámara web. Tales casos de uso requieren algunos tipos de características muy específicas, como descubrir dispositivos cercanos, comunicaciones seguras o sesiones multi-dispositivo. Apple empezó introduciendo tales características hace tiempo en sus propios SDKS, y ahora Google ha lanzado la primera vista previa de su SDK multi-dispositivo. Aunque esta vista previa tiene varias limitaciones (por ejemplo, sólo los teléfonos y tablets están soportados y sólo dos dispositivos a la vez) la tecnología es emocionante y puede ser utilizada conforme se despliega con el tiempo.

97. Testing de componentes con Cypress

Evaluar

Cypress Component Testing proporciona un banco de pruebas para componentes testeables que permite construir y probar componentes de interfaz de usuario rápidamente. Puedes escribir pruebas de regresión de componentes de interfaz de usuario con el mismo API con el que escribes pruebas end-to-end (E2E) sobre la interfaz de usuario. A pesar de que aún está en beta, las pruebas de componentes será la función más importante de Cypress 10.

98. JobRunr

Evaluar

JobRunr es una biblioteca en Java para procesamiento de tareas en segundo plano, y una alternativa al planificador Quartz. Nuestros equipos han disfrutado usando el cuadro de mando integrado de JobRunr, el cual no sólo es fácil de usar, sino que también que permite monitorear y planificar tareas en segundo plano. JobRunr es de código abierto y gratis para uso comercial; sin embargo, para funcionalidades como migración de jobs y recuperación de tareas, es necesario obtener una licencia pagada.

99. Million

Evaluar

Million es una nueva librería de DOM virtual para JavaScript. Similar a Svelte, aprovecha el compilador, Vite, para crear pequeños paquetes de JavaScript con un rendimiento de renderizado excepcional. La librería Million se presenta como un único paquete NPM con varios módulos, incluidos router, jsx-runtime y un módulo de Compatibilidad con React para crear aplicaciones de página única. Aunque React popularizó el DOM virtual hace una década, es fascinante ver nuevas innovaciones en este ámbito.

100. Soketi

Evaluar

Soketi es un servidor de WebSockets de código abierto. Si tu aplicación es compatible con el protocolo Pusher Soketi se puede conectar directamente ya que implementa por completo la versión 7 del protocolo pusher. Consideramos que el soporte beta para Cloudflare Workers es especialmente interesante ya que abre la puerta a utilizar WebSockets en la frontera de red.

101. Stable Diffusion

Evaluar

El programa DALL·E de OpenAI llamó la atención por su capacidad de crear imágenes a partir de indicaciones de texto. Ahora, Stable Diffusion ofrece la misma capacidad pero, sobre todo, es de código abierto. Cualquiera que tenga acceso a una tarjeta gráfica potente puede experimentar con el modelo, y cualquiera que disponga de suficientes recursos informáticos puede recrear el modelo por sí mismo. Los resultados son asombrosos pero también plantean cuestiones importantes. Por ejemplo, el modelo se ha entrenado con pares de imagen-texto obtenidos a través de un amplio rastreo de Internet y, por tanto, reflejará sesgos sociales, lo que significa que podría producir contenidos ilegales, molestos o, como mínimo, indeseables. Stable Diffusion incluye ahora un clasificador de seguridad basado en la IA; sin embargo, dada su naturaleza de código abierto, la gente puede desactivar el clasificador. Por último, los artistas han señalado que, con las indicaciones adecuadas, el modelo es capaz de imitar su estilo artístico. Esto plantea cuestiones sobre las implicaciones éticas y legales de una IA capaz de imitar a un artista.

102. Bóveda de Datos Sintéticos o Synthetic Data Vault

Evaluar

Bóveda de Datos Sintéticos o Synthetic Data Vault (SDV) es un ecosistema de librerías para generación de datos sintéticos que puede aprender la distribución de un conjunto de datos para generar datos sintéticos con el mismo formato y las mismas propiedades estadísticas que la fuente. En el pasado, hablamos sobre las desventajas de usar **datos de producción en entornos de prueba**. Sin embargo, los matices de la distribución de datos en producción es muy difícil de reproducir manualmente, resultando en defectos y sorpresas. Creemos que SDV y otras herramientas similares pueden abordar esa brecha creando datos similares a los de producción para datos en **tabla simple**, **tablas múltiples complejas** y **series multivariables cronológicas.** Aunque SDV no es nuevo, nos gusta bastante y decidimos destacarlo.

103. Carbon

Resistir

Estamos observando cierto interés por el lenguaje de programación conocido como **Carbon**. Esto no nos sorprende debido a que cuenta con el respaldo de Google y se presenta como sucesor natural de C++. En nuestra opinión C++ no puede ser reemplazado lo suficientemente rápido, tal y como han demostrado los ingenieros de software durante las últimas décadas: escribir código seguro y libre de errores en C++ es extremadamente difícil y costoso en tiempo. Si bien Carbon es un concepto interesante enfocado a la migración desde C++, a falta de un compilador operativo, es evidente que aún queda un largo camino por recorrer antes de que sea utilizable, además de que existen otros lenguajes de programación modernos que son una buena opción para migrar desde C++. Todavía es muy pronto para determinar si Carbon se convertirá en el sucesor natural de C++, pero desde la perspectiva que tenemos hoy, les recomendamos a los equipos considerar **Rust** y **Go** en lugar de posponer la migración mientras esperan la llegada de Carbon.

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

Síguenos en tu red social favorita o suscríbete.

Suscríbete ahora

Thoughtworks es una consultora global de tecnología que integra estrategia, diseño e ingeniería para impulsar la innovación digital. Somos 12,000+ personas en 50 oficinas en 18 países. En los últimos 25+ años, hemos logrado un impacto extraordinario junto con nuestros clientes, ayundándoles a resolver problemas complejos de negocio a través de la tecnología como elemento diferenciador.

/thoughtworks