/thoughtworks

TECHNOLOGY RADAR

Una guía con opiniones sobre las tecnologías de vanguardia

Volumen 24

#TWTechRadar
thoughtworks.com/radar

Contribuyentes

El Radar Tecnológico está preparado por la Junta Asesora de Tecnología de Thoughtworks

La Junta Asesora de Tecnología (TAB) es un grupo de 20 tecnologistas senior de Thoughtworks. El TAB se reúne presencialmente dos veces al año y quincenalmente por teléfono. Su función principal es ser un grupo asesor para Thoughtworks CTO,Rebecca Parsons.

La TAB actúa como un solo individuo que puede analizar temas que influyen en la tecnología y a tecnologistas de Thoughtworks. Con la pandemia mundial de hoy, se volvió a crear este volumen del Technology Radar a través de un evento virtual.

Rebecca Parsons (CTO)

Martin Fowler (Chief Scientist)

Bharani Subramaniam

Birgitta Böckeler

Brandon Byars

Camilla Crispim

Cassie Shum

Erik Dörnenburg

Evan Bottcher

Fausto de la Torre

Hao Xu

Ian Cartwright

James Lewis

Lakshminarasimhan Sudarshan

Mike Mason

Neal Ford

Perla Villarreal

Rachel Laycock

Scott Shaw

Shangqi Liu

Zhamak Dehghani

Sobre el Radar

Nuestros Thoughtworkers son apasionados por la tecnología. La construimos, investigamos, probamos, liberamos su código fuente, escribimos sobre el y constantemente queremos mejorarlo para todas las personas. Nuestra misión es liderar la excelencia tecnológica y revolucionar las TI. En soporte de esta misión, creamos y compartimos el Radar Tecnológico de Thoughtworks. La Junta Asesora de Tecnología de Thoughtworks, un grupo de líderes senior en la tecnología dentro de Thoughtworks, son quienes crean el Radar. Se reúnen regularmente para discutir la estrategia tecnológica global de Thoughtworks y las tendencias tecnológicas que impactan significativamente en nuestra industria.

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

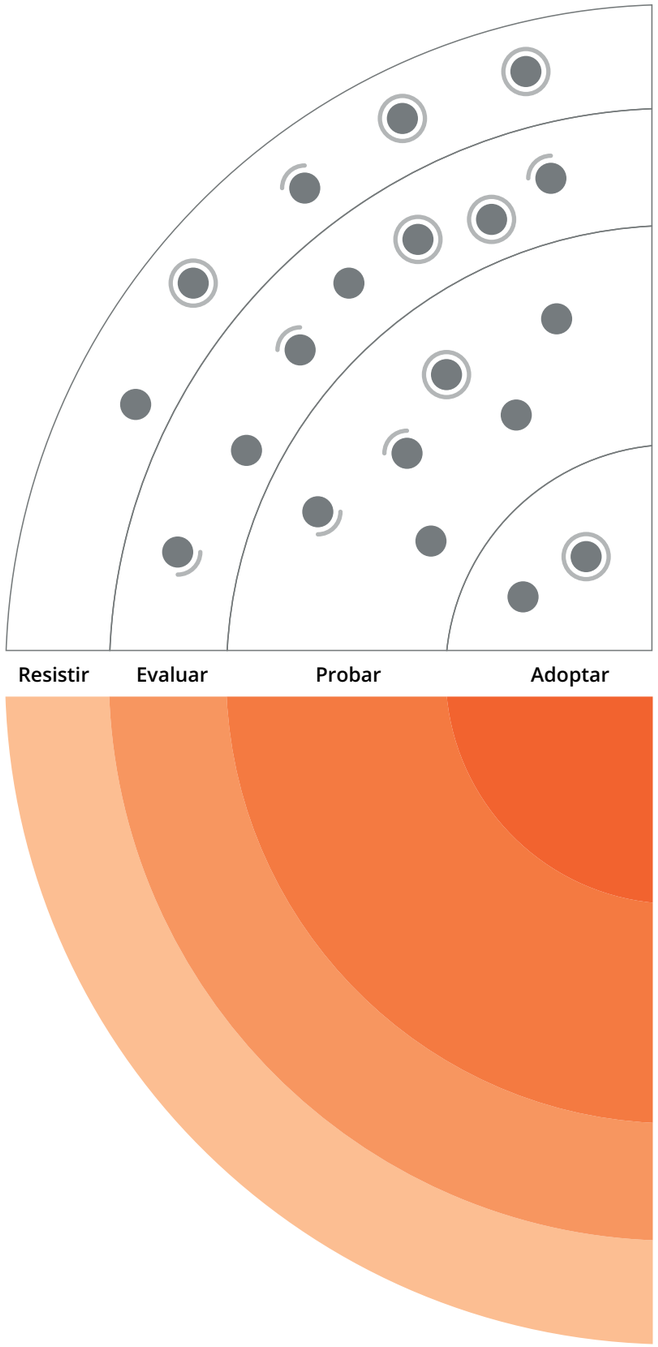
Te alentamos a explorar estas tecnologías. El Radar es gráfico por naturaleza, agrupando items en técnicas, herramientas, plataformas y lenguajes & frameworks. Cuando items del Radar llegan a aparecer en múltiples cuadrantes, elegimos el que parezca más apropiado. Después agrupamos estos items en cuatro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

Para más información del Radar, entra en thoughtworks.com/radar/faq.

Un vistazo al Radar

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

Un blip es la tecnología o técnica que juega un rol en el desarollo de software. Los blips son cosas que están en constante ”movimiento” — es decir, su posición en el Radar está cambiando — generalmente indicando que estamos encontrando una creciente confianza en ellos a medida que avanzan por los anillos.



Nuevo

Desplazado adentro/afuera

Ningún cambio

Nuestro Radar tiene una visión hacia el futuro. Para hacer espacio a nuevos items, hemos retirado los items que no han sufrido cambios recientes, lo cual no es un reflejo de su valor sino más bien del espacio limitado disponible en nuestro Radar.

Adoptar

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

Probar

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

Evaluar

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

Resistir

Proceder con precaución.

Temas de esta edición

Equipos de plataforma impulsan la velocidad de salida al mercado

Con más frecuencia, las organizaciones están adoptando el concepto de equipos de plataforma: armar un grupo dedicado que crea y da soporte a las capacidades internas de la plataforma (nativa a la nube, entrega continua, observabilidad moderna, patrones de autenticación/authorización, mallas de servicio, etc), para luego aprovecharlas para acelerar el desarrollo de aplicaciones, reducir la complejidad operacional y mejorar los tiempos de salida al mercado. Esta madurez creciente es bienvenida y presentamos esta técnica en el Radar en 2017. Pero con el incremento de la madurez, también hemos descubierto antipatrones que las organizaciones deben evitar. Por ejemplo, "una plataforma para gobernar a todas" puede no ser óptima, una "plataforma grande desde el inicio" puede tomar años para entregar valor y "constrúyela y ellos vendrán" puede terminar como un esfuerzo desperdiciado. En cambio, usar un enfoque con orientación al producto puede ayudar a esclarecer qué deben proveer cada una de las plataformas internas, dependiendo de sus clientes. Las compañías que arman sus equipos de plataforma utilizando un sistema de tickets como silos de operaciones a la vieja usanza, encuentran las mismas desventajas de una priorización mal alineada: lentitud en la retroalimentación y en las respuestas, contención en la asignación de recursos y otros problemas bien conocidos por el uso de silos. También hemos visto aparecer un gran número de nuevas herramientas y patrones de integración para equipos y tecnologías, lo que permite un particionamiento más efectivo entre ambas.

Comodidades consolidadas frente a lo mejor de su clase

A medida que las prácticas de ingeniería que ofrecen automatización, escala y otros objetivos modernos se vuelven más comunes en los equipos de desarrollo, vemos la correspondiente integración de herramientas de desarrollo en muchas plataformas, particularmente en el espacio de la nube. Por ejemplo, los repositorios de artefactos, el control de código fuente, los *pipelines* de CI/CD, las wikis y demás herramientas similares eran escogidas a mano e individualmente por los equipos de desarrollo y ensambladas a la carta. Ahora, plataformas de entrega como Azure DevOps y ecosistemas como GitHub han integrado muchas de estas categorías de herramientas. Si bien el nivel de madurez varía según las ofertas de cada plataforma, el atractivo de tener "todo en un mismo lugar" con respecto a las herramientas de entrega es innegable. En general, parece que el equilibrio radica en tener un conjunto de herramientas consolidadas que ofrecen una mayor comodidad para el desarrollador y menos rotación, a pesar que estas herramientas rara vez representan lo mejor posible.

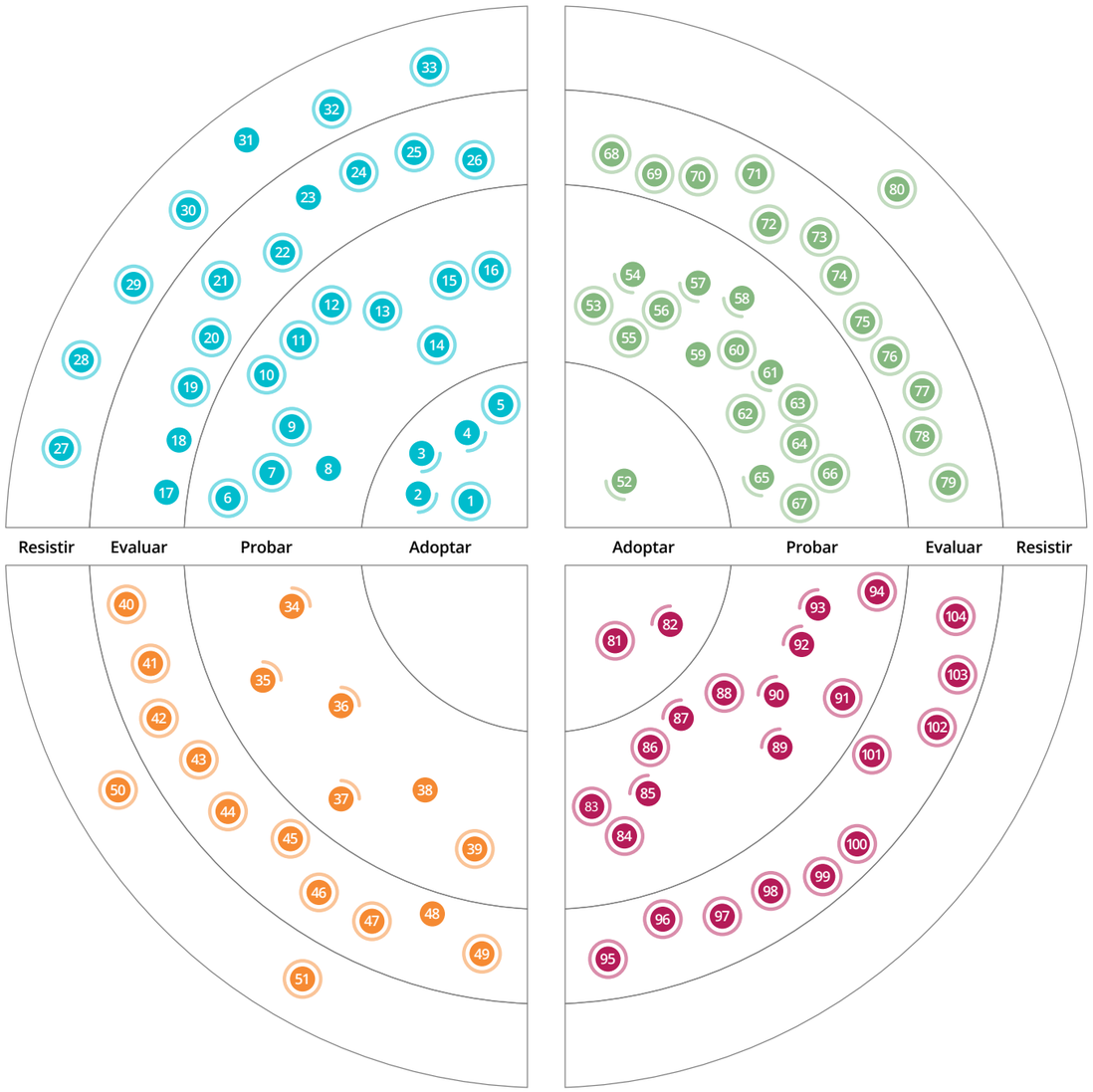
"Muy complejo para ser un blip", lo perenne del radar

En la nomenclatura del Radar, el estado final luego de discutir sobre varios temas complejos es "muy complejo para ser un *blip*" o "TCTB" (del inglés "too complex to blip"): son elementos que desafían nuestros parámetros de clasificación porque ofrecen una serie de pros y contras, una gran cantidad de matices en cuanto a la aplicabilidad del consejo o la herramienta, o por otros motivos que nos impiden resumir nuestras opiniones en pocas frases. Con frecuencia, estos temas se tratarán como artículos, *podcasts*, y en otros medios que no son el Radar. Algunas de nuestras mejores conversaciones se centran en estos temas: son importantes pero complejas, e impiden llegar a un único, y resumido, punto de vista. Numerosos temas se repiten reunión tras reunión (además de que los vemos frecuentemente con nuestros clientes) que terminan siendo catalogados como TCTB, como los monorepos, pautas de orquestación para arquitecturas distribuidas y modelos de ramificación, entre otros. Si alguien se pregunta por qué estos temas importantes no aparecen en el Radar, no es por falta de conciencia o deseo de nuestra parte. Como muchos temas en el desarrollo de software, existen demasiados aspectos que equilibrar para poder emitir consejos claros e inequívocos. A veces encontramos elementos más pequeños de los temas más importantes sobre los que si podemos ofrecer consejo, y llegan a estar en el Radar; pero los temas más importantes siguen perpetuamente inestables, con demasiados matices, para el Radar.

Discernir el contexto para el acoplamiento de arquitectura

Una discusión que se repite prácticamente en todas nuestras reuniones (revisa el tema denominado "Muy complejo para ser un *blip*", lo perenne del radar") tiene que ver con el nivel adecuado de acoplamiento en la arquitectura de software entre microservicios, componentes, *API gateways*, centros de integración (*integration hubs*), *front-ends*, etc.; las personas de arquitectura y de desarrollo luchan por encontrar el nivel correcto de acoplamiento prácticamente en todos los lugares donde se pueden conectar dos elementos de software, considerando que muchos consejos comunes fomentan el desacoplamiento extremo, aunque eso dificulta la creación de flujos de trabajo. El acoplamiento en la arquitectura pasa por muchas consideraciones importantes: la forma en que se conectan las cosas, comprender el acoplamiento semántico inherente dentro de cada dominio del problema, cómo se invocan las cosas entre sí o el funcionamiento de la transaccionalidad (a veces en combinación con otras características complicadas como la escalabilidad). El software no puede existir sin algún nivel de acoplamiento por fuera de los sistemas monolíticos singulares; encontrar el balance correcto para determinar los tipos y niveles de acoplamiento se convierte en una habilidad crítica con las arquitecturas modernas. Vemos malas prácticas específicas como la generación de código para bibliotecas cliente, y buenas prácticas como el uso juicioso de los patrones BFF. Sin embargo, brindar consejos generales en este ámbito es inútil y tampoco existen soluciones mágicas. Se debe invertir tiempo y esfuerzo en comprender los factores en juego al tomar estas decisiones caso por caso, en lugar de buscar una solución genérica pero inadecuada.

El Radar



Técnicas

Adoptar
1. Expansión-contracción de APIs
2. Entrega continua para aprendizaje automático (CD4ML)
3. Sistemas de diseño
4. Equipos de producto de ingeniería de plataforma
5. Estrategia de rotación de cuentas de servicios

Probar
6. Sandboxes en la nube
7. Contextual bandits
8. Imágenes Docker sin distribución
9. Ethical Explorer
10. Renovación de legados guiada por hipótesis
11. Enfoque ligero respecto a las RFCs
12. Aprendizaje automático más simple posible
13. Inyección de SPA
14. Carga cognitiva del equipo
15. Xcodeproj administrado por herramientas
16. Tipos compartidos entre la UI y el BFF

Evaluar
17. Plataformas delimitadas de poco código
18. Identidad descentralizada
19. Radiador del desfase de los despliegues
20. Cifrado homomórfico
21. Hotwire
22. Importmaps para micro frontends
23. Modelo Abierto de Aplicaciones (OAM)
24. Analítica web centrada en la privacidad
25. Programación en grupo remota
26. Computación segura entre múltiples partes

Resistir
27. GitOps
28. Equipos de plataforma en capas
29. Requisitos ingenuos de complejidad de contraseñas
30. Revisión por pares equivalente a pull request
31. SAFe™
32. Propiedad separada del código y del pipeline
33. Modelos operativos de plataforma basados en tickets

Plataformas

Adoptar

Probar
34. Kit de desarrollo en la nube de AWS
35. Backstage
36. Delta Lake
37. Materialize
38. Snowflake
39. Fuentes variables

Evaluar
40. Apache Pinot
41. Bit.dev
42. DataHub
43. Feature Store
44. JuiceFS
45. API de Kafka sin Kafka
46. NATS
47. Opstrace
48. Pulumi
49. Redpanda

Resistir
50. Azure Machine Learning
51. Productos hechos en casa para infraestructura como código

Herramientas

Adoptar
52. Sentry

Probar
53. axe-core
54. dbt
55. esbuild
56. Flipper
57. Great Expectations
58. k6
59. MLflow
60. OR-Tools
61. Playwright
62. Prowler
63. Pyright
64. Redash
65. Terratest
66. Tuple
67. Why Did You Render

Evaluar
68. Buildah y Podman
69. GitHub Actions
70. Imagen nativa de Graal
71. HashiCorp Boundary
72. imgcook
73. Longhorn
74. Operator Framework
75. Recommender
76. Remote - WSL
77. Spectral
78. Yelp detect-secrets
79. Zally

Resistir
80. AWS CodePipeline

Lenguajes & Frameworks

Adoptar
81. Combine
82. LeakCanary

Probar
83. Angular Testing Library
84. AWS Data Wrangler
85. Blazor
86. FastAPI
87. io-ts
88. Kotlin Flow
89. LitElement
90. Next.js
91. Módulos bajo demanda
92. Streamlit
93. SWR
94. TrustKit

Evaluar
95. .NET 5
96. bUnit
97. Dagster
98. Flutter para Web
99. Jotai y Zustand
100. Kotlin Multiplatform Mobile
101. LVGL
102. React Hook Form
103. River
104. Federación de Módulos de Webpack

TECHNOLOGY RADAR

Técnicas

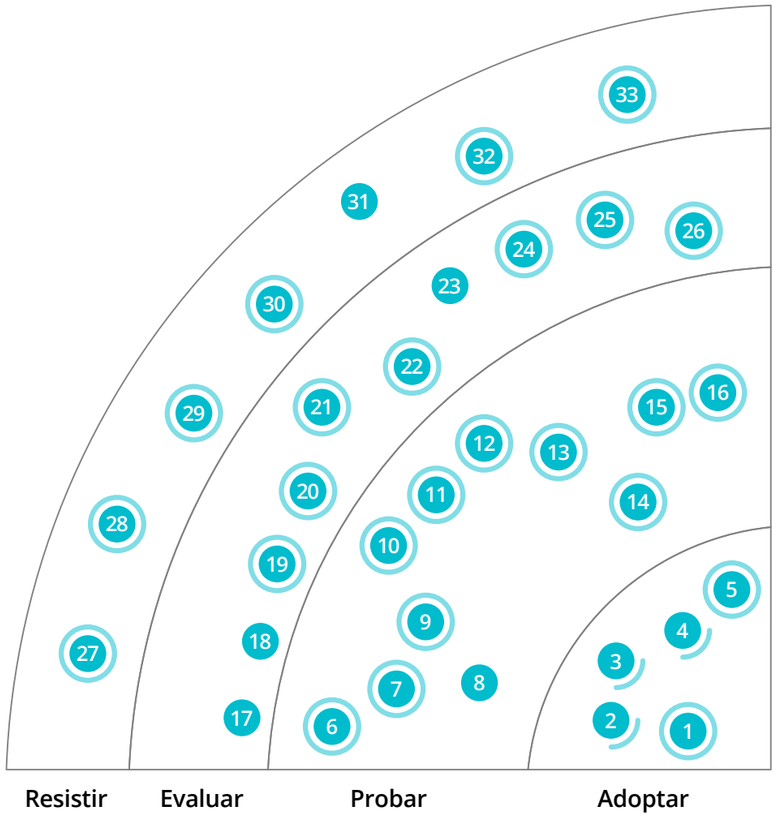
Técnicas

Expansión-contracción de APIs
Adoptar

El patrón de expansión-contracción de APIs , a veces llamado cambios en paralelo (parallel change), será familiar para muchas personas, especialmente en relación a su uso con bases de datos o código; sin embargo, sólo vemos niveles bajos de adopción en las APIs. Específicamente, vemos el uso esquemas complejos de versionamiento y la introducción de cambios disruptivos en escenarios donde la simple expansión y posterior contracción del API sería suficiente. Por ejemplo, primero se añadiría un nuevo elemento a un API, para luego descontinuar un elemento existente, y posteriormente remover los elementos descontinuados, una vez que los consumidores se hayan cambiado al esquema más nuevo. Este enfoque requiere cierta coordinación y visibilidad de los consumidores del API, tal vez a través de técnicas como las pruebas de contratos dirigidos por consumidores.

Entrega continua para aprendizaje automático (CD4ML)
Adoptar

Vemos a la entrega continua para aprendizaje automático (CD4ML) como un buen punto de inicio predeterminado para cualquier solución de aprendizaje automático que se despliegue en producción. Muchas organizaciones dependen cada vez más de soluciones de aprendizaje automático tanto para las ofertas a sus clientes como para operaciones internas, así que tiene sentido comercial aplicar las lecciones y las buenas practicas capturadas por la entrega continua (CD) a las soluciones de aprendizaje automático.



Sistemas de diseño
Adoptar

A medida que el desarrollo de aplicaciones se hace cada vez más dinámico y complejo, es un desafío ofrecer productos accesibles y utilizables con un estilo coherente. Esto es particularmente cierto en las organizaciones más grandes con varios equipos que trabajan en diferentes productos. Los sistemas de diseño definen una colección de patrones de diseño, bibliotecas de componentes y buenas prácticas de diseño e ingeniería que garantizan productos digitales consistentes. Basados en las guías de estilo corporativas del pasado, los sistemas de diseño ofrecen bibliotecas y documentos compartidos que son fáciles de encontrar y usar. Generalmente la guía se escribe como código y se mantiene bajo control de versiones para que sea menos ambigua y más fácil de mantener que los documentos simples. Los sistemas de diseño se han convertido en un enfoque estándar cuando se trabaja entre equipos y disciplinas en el desarrollo de productos porque permiten que los equipos se concentren. Pueden abordar desafíos estratégicos en torno al producto en sí sin reinventar la rueda cada vez que se necesita un nuevo componente visual.

Adoptar
1. Expansión-contracción de APIs
2. Entrega continua para aprendizaje automático (CD4ML)
3. Sistemas de diseño
4. Equipos de producto de ingeniería de plataforma
5. Estrategia de rotación de cuentas de servicios

Probar
6. Sandboxes en la nube
7. Contextual bandits
8. Imágenes Docker sin distribución
9. Ethical Explorer
10. Renovación de legados guiada por hipótesis
11. Enfoque ligero respecto a las RFCs
12. Aprendizaje automático más simple posible
13. Inyección de SPA
14. Carga cognitiva del equipo
15. Xcodeproj administrado por herramientas
16. Tipos compartidos entre la UI y el BFF

Evaluar
17. Plataformas delimitadas de poco código
18. Identidad descentralizada
19. Radiador del desfase de los despliegues
20. Cifrado homomórfico
21. Hotwire
22. Importmaps para micro frontends
23. Modelo Abierto de Aplicaciones (OAM)
24. Analítica web centrada en la privacidad
25. Programación en grupo remota
26. Computación segura entre múltiples partes

Resistir
27. GitOps
28. Equipos de plataforma en capas
29. Requisitos ingenuos de complejidad de contraseñas
30. Revisión por pares equivalente a pull request
31. SAFe™
32. Propiedad separada del código y del pipeline
33. Modelos operativos de plataforma basados en tickets

Técnicas

Con un nombre que hace honor a las máquinas tragamonedas de los casinos (en inglés, "bandits" o "one-armed bandits"), el algoritmo explora diferentes opciones para aprender más sobre los resultados esperados y los equilibra explotando aquellas que se desempeñan bien.

(Contextual bandits)

Equipos de producto de ingeniería de plataforma

Adoptar

Como se mencionó en uno de los temas de esta edición, la industria está ganando cada vez más experiencia con los equipos de producto de ingeniería de plataforma que crean y dan soporte a plataformas internas. Estas son utilizadas por los equipos de toda la organización y aceleran el desarrollo de aplicaciones, reducen la complejidad operativa y mejoran el tiempo de comercialización. Con una adopción cada vez mayor, también tenemos más claros los patrones buenos y malos de este enfoque. Al crear una plataforma, es fundamental tener clientes y productos claramente definidos que se beneficiarán de ella en lugar de construir en el vacío. Advertimos en contra de los equipos de plataforma en capas, que conservan los silos de tecnología existentes pero se aplican la etiqueta de "equipo de plataforma", y también contra los modelos operativos de plataforma basados en tickets. Todavía somos partidarios de utilizar los conceptos de Team Topologies mientras pensamos en cómo organizar mejor a los equipos de plataforma. Consideramos que los equipos de producto de ingeniería de plataforma son un enfoque estándar y un habilitador significativo para lograr TI de alto rendimiento.

Estrategia de rotación de cuentas de servicios

Adoptar

Recomendamos encarecidamente a las organizaciones que, cuando realmente se necesite usar cuentas de servicios en la nube, roten las credenciales. La rotación es una de las tres R de la seguridad. Es muy fácil para las organizaciones perder de vista a estas cuentas hasta que ocurre un incidente. Esto da lugar a la existencia de cuentas con permisos innecesariamente amplios que se mantienen en uso por largos periodos de tiempo, además de la falta de planificación sobre como reemplazarlas o rotarlas. El aplicar regularmente una estrategia de rotación de cuentas de servicios en la nube también da la oportunidad de practicar el principio del menor privilegio.

Sandboxes en la nube

Probar

Dado que la nube se está convirtiendo cada vez más en un producto básico y que la posibilidad de crear sandboxes en la nube es mucho más sencillo y está disponible a gran escala, nuestros equipos prefieren entornos de desarrollo en la nube (en lugar de locales) para reducir la complejidad del mantenimiento. Estamos viendo que las herramientas para hacer una simulación local de los servicios nativos de la nube limitan la confianza en los ciclos de construcción y prueba de los desarrolladores; por lo tanto, estamos buscando centrarnos en estandarizar los ambientes sandbox en la nube, en lugar de ejecutar los componentes nativos de la nube en una máquina del desarrollador. Esto impulsará las buenas prácticas de infraestructura como código como medida obligatoria y los buenos procesos de incorporación para proveer de ambientes sandboxes a los nuevos desarrolladores. Existen riesgos asociados a esta transición, ya que supone que los desarrolladores tendrán una dependencia absoluta de la disponibilidad del entorno en la nube, y puede ralentizar el ciclo de retroalimentación. Recomendamos encarecidamente adoptar algunas prácticas de gobernanza lean en relación con la estandarización de estos entornos de sandboxes, especialmente en lo que respecta a la seguridad, la administración de identidades (IAM por sus siglas en inglés) y los despliegues regionales.

Contextual bandits

Probar

Contextual bandits es un tipo de aprendizaje por refuerzo muy adecuado para problemas que requieren un equilibrio entre exploración y explotación ("Exploration-Exploitation Trade-off"). Con un nombre que hace honor a las máquinas tragamonedas de los casinos (en inglés, "bandits" o "one-armed bandits"), el algoritmo explora diferentes opciones para aprender más sobre los resultados esperados y los equilibra explotando aquellas que se desempeñan bien. Hemos usado esta técnica exitosamente en escenarios donde se ha tenido muy poca información para entrenar y desplegar otros modelos de aprendizaje automático. El hecho de que es posible agregar contexto a este equilibrio entre exploración y explotación lo hace apropiado para una amplia variedad de casos de uso, como pruebas A/B, recomendaciones y optimizaciones de diseño, etc.

Imágenes Docker sin distribución

Probar

Cuando se construyen imágenes Docker para las aplicaciones, usualmente nos preocupan dos cosas: la seguridad y su tamaño. Tradicionalmente hemos usado herramientas de escaneo de seguridad en contenedores para detectar y corregir vulnerabilidades y riesgos comunes, y distribuciones pequeñas como Alpine Linux para resolver el tema del tamaño de la imagen y del rendimiento de la distribución. Pero con el aumento de las amenazas de seguridad, eliminar todos los posibles vectores de ataque es más importante que nunca. Es por esto que las imágenes Docker sin distribución se están convirtiendo en la opción predeterminada para contenedores para despliegue. Este tipo de imágenes

reducen el tamaño y las dependencias suprimiendo las distribuciones de sistemas operativos completos. Esta técnica reduce el ruido en los escaneos de seguridad y la superficie de ataque a la aplicación: hay menos vulnerabilidades que corregir y, como extra, estas imágenes son más eficientes. Google ha publicado un conjunto de imágenes de contenedor sin distribución para diferentes lenguajes. Se pueden crear imágenes para aplicación sin distribución usando la herramienta de construcción Bazel de Google o simplemente usando Dockerfiles multistage. Hay que tener en cuenta que los contenedores sin distribución no tienen un intérprete de comandos (shell) para depurar. Sin embargo es fácil encontrar en línea versiones de contenedores sin distribución habilitados para la depuración que incluyen a BusyBox como intérprete de comandos. Google ha sido pionero en esta técnica y, en nuestra experiencia, sigue limitada en gran medida a las imágenes generadas por ellos. Nos sentiríamos mejor si existiera más de un proveedor para elegir. Además, se debe tener precaución al correr Trivy o escáneres de vulnerabilidades similares, ya que solamente sus versiones más recientes son compatibles con este tipo de contenedores.

Ethical Explorer

Probar

El grupo que está detrás de Ethical OS (Omidyar Network, una empresa autodenominada de cambio social creada por el fundador de eBay, Pierre Omidyar) ha liberado una nueva iteración llamada Ethical Explorer. Este nuevo paquete se basa en las lecciones aprendidas con el uso de Ethical OS y agrega más preguntas para que los equipos de producto las tengan en cuenta. El kit, que puede descargarse de forma gratuita y doblarse en tarjetas para iniciar una conversación, tiene preguntas abiertas para varias "zonas de riesgo" técnicas, incluida la vigilancia ("¿puede alguien utilizar nuestro producto o servicio para rastrear o identificar a otros usuarios?"), la desinformación, la exclusión, el sesgo algorítmico, la adicción, el control de datos, los malos actores y el poder desmesurado. La guía que se incluye tiene actividades y talleres, ideas para iniciar conversaciones y consejos para lograr la aceptación de la organización. Si bien tenemos un largo camino por recorrer como industria para representar mejor los factores externos éticos de nuestra sociedad digital, hemos tenido algunas conversaciones fructíferas sobre productos utilizando Ethical Explorer y nos alienta la creciente conciencia de la importancia de las decisiones sobre los productos para abordar los problemas sociales.

Renovación de legados guiada por hipótesis

Probar

A menudo nos piden que renovemos, actualicemos o reparemos sistemas legados que originalmente no hemos construido. En ocasiones, debemos atender problemas técnicos como mejorar el desempeño y la fiabilidad del sistema. Un enfoque común para abordar estos asuntos es crear "historias técnicas" usando el mismo formato que una historia de usuario, pero con un resultado técnico en vez de uno de negocio. Sin embargo, estas tareas técnicas a menudo son difíciles de estimar, llevan más tiempo del que se anticipa, o no terminan produciendo el resultado deseado. Un método alternativo y más exitoso es aplicar la renovación de legados guiada por hipótesis. En lugar de trabajar con un banco (backlog) estándar de historias de usuario, el equipo se apropia de un resultado técnico cuantificable y establece colectivamente un conjunto de hipótesis sobre el problema. Luego llevan a cabo experimentos en iteraciones de tiempo establecidas para verificar o refutar cada hipótesis en orden de prioridad. El flujo de trabajo resultante está optimizado para reducir la incertidumbre en lugar de seguir un plan hacia un resultado predecible.

Enfoque ligero respecto a las RFCs

Probar

A medida que las organizaciones se mueven hacia arquitecturas evolutivas, es importante llevar un registro de las decisiones de diseño, arquitectura, técnicas y formas de trabajar de los equipos. El proceso de recolectar y consolidar la retroalimentación que conduce a esas decisiones comienza con las Solicitudes de Comentarios (Request for Comments, RFCs). Las RFCs son una técnica para recoger contexto, ideas de diseño y de arquitectura y colaborar con los equipos para finalmente llegar a tomar decisiones junto con su contexto y consecuencias. Nosotros recomendamos que las organizaciones tomen un enfoque ligero respecto a las RFCs, usando una plantilla estandarizada entre los equipos además de control de versiones para recoger las RFCs.
Es importante agregar estas decisiones en un registro de auditoría, para beneficiar a los miembros futuros de los equipos y así documentar la evolución técnica y de negocio de una organización. Las organizaciones maduras han usado los RFCs en equipos autónomos para impulsar mejoras en la comunicación y la colaboración, especialmente cuando se trata de tomar decisiones importantes entre equipos.

Técnicas

Cuando necesitamos modernizar sistemas heredados con aplicaciones de página única (en inglés single-page applications o SPA), en lugar de envolver el sistema heredado, incorporamos el comienzo de la nueva SPA en el documento HTML que contiene el anterior y dejamos que se expanda lentamente en funcionalidad.

(Inyección de SPA)

Técnicas

La interacción dentro del equipo es una de las variables que define la facilidad con la que los equipos pueden entregar valor a sus clientes. Las autoras de las topologías de equipos desarrollaron una evaluación para medir estas interacciones las cuales nosotros llamamos carga cognitiva del equipo

(Carga cognitiva del equipo)

Aprendizaje automático más simple posible

Probar

Todos los proveedores grandes de la nube ofrecen una gama deslumbrante de soluciones de aprendizaje automático. Estas poderosas herramientas pueden proveer mucho valor, aunque tienen costos asociados. Existe el costo puro por la ejecución de estos servicios, cobrado por el proveedor de la nube. Además, hay una especie de impuesto a su operación. Estas herramientas complejas necesitan ser entendidas y operadas, y con cada nueva herramienta que se agrega a la arquitectura, esta carga impositiva se incrementa. En nuestra experiencia, muchas veces los equipos eligen herramientas complejas porque menosprecian el poder de herramientas más sencillas como la regresión lineal. Muchos de los problemas de aprendizaje automático no necesitan de una GPU ni de redes neuronales. Por esta razón recomendamos utilizar el aprendizaje automático más simple posible , aprovechando herramientas y modelos sencillos, algunos cientos de líneas de código Python, en la plataforma de cómputo que esté más a la mano. Se debe dejar las herramientas complejas solo para cuando se pueda demostrar su necesidad.

Inyección de SPA

Probar

El patrón de estrangulamiento es a menudo la primera estrategia que viene a la mente para modernizar sistemas legados, donde el nuevo código envuelve al antiguo y absorbe lentamente la capacidad de manejar toda la funcionalidad necesaria. Este tipo de enfoque “de afuera hacia adentro” funciona bien para algunos sistemas legados, sin embargo, ahora que hemos tenido suficiente experiencia con aplicaciones de página única (SPA), y dado que estas aplicaciones se están volviendo sistemas legados en sí, notamos que el enfoque opuesto, “de adentro hacia afuera”, se está usando para reemplazarlas. En vez de envolver al sistema legado, se agrega las bases de la nueva SPA en el documento HTML que contiene a la anterior y dejamos que se expanda lentamente en funcionalidad. Los marcos de trabajo de SPA ni siquiera necesitan ser los mismos, siempre que los usuarios puedan tolerar el impacto causado al rendimiento por el aumento del tamaño de la página (por ejemplo, al incrustar una nueva aplicación React dentro de una antigua en AngularJS). La inyección de SPA permite eliminar iterativamente la aplicación antigua hasta que la nueva tome su lugar por completo. Mientras que la metáfora del árbol estrangulador puede verse como un tipo de parásito que usa la superficie externa, estable, del árbol huésped para sostenerse hasta echar raíces, mientras el huésped muere; este enfoque es más como inyectar un agente externo en el huésped, apoyándose en la funcionalidad de la SPA original hasta poder reemplazarla por completo.

Carga cognitiva del equipo

Probar

La arquitectura de un sistema replica la estructura de la organización y sus patrones de comunicación. No es nada nuevo que debamos ser intencionales sobre cómo interactúan los equipos (véase, por ejemplo, la Maniobra Invertida de Conway). La interacción de un equipo es una de las variables que definen cuán rápido y cuán fácil les resulta entregar valor a sus clientes. Estamos contentos de haber encontrado una forma de medir esas interacciones: usamos la evaluación que proponen las autoras de Team Topologies, que proporciona una forma de entender cuán fácil o difícil le resulta a los equipos construir, probar y mantener sus servicios. Midiendo la carga cognitiva del equipo , podríamos aconsejar mejor a nuestros clientes sobre cómo cambiar la estructura de sus equipos y evolucionar sus interacciones.

Xcodeproj administrado por herramientas

Probar

Muchas de nuestras personas que desarrollan para iOS con Xcode a menudo sufren por culpa de los cambios en el archivo Xcodeproj cada vez que cambia el proyecto. El formato de este archivo no es legible para las personas, por lo que intentar manejar conflictos es bastante complicado y puede conducir a una pérdida de productividad con el riesgo de estropear el proyecto entero (si algo malo pasa con el archivo, Xcode no funcionará bien y las desarrolladoras quedarán bloqueadas). En lugar de intentar fusionar los cambios y arreglar el archivo manualmente o versionarlo, recomendamos un enfoque llamado Xcodeproj administrado por herramientas : la configuración del proyecto Xcode se define en YAML (XcodeGen, Struct), Ruby (Xcake) o Swift (Tuist) y estas herramientas generan el archivo Xcodeproj a partir del archivo de configuración y la estructura del proyecto. Como resultado, los conflictos al fusionar cambios en el archivo Xcodeproj serán una cosa del pasado y, cuando ocurran en el archivo de configuración, son mucho más fáciles de resolver.

Tipos compartidos entre la UI y el BFF

Probar

Con TypeScript convirtiéndose en un lenguaje común para el desarrollo de

front-end, y Node.js siendo una de las tecnologías preferidas para el desarrollo de BFFs, percibimos un incremento en el uso de tipos compartidos entre la UI y el BFF. En esta técnica, un único conjunto de definiciones de tipos de datos es utilizado para definir tanto los objetos de datos devueltos por las consultas del front-end, como los datos entregados desde el servidor de back-end para satisfacer esas consultas. Normalmente, seríamos cautelosos con esta práctica debido al acoplamiento innecesario que crea a través de los límites de los procesos; pero, muchos equipos están encontrando que los beneficios de este enfoque superan cualquier riesgo de alto acoplamiento. Dado que el patrón BFF funciona mejor cuando el código de la interfaz de usuario y el del BFF son propiedad del mismo equipo, y considerando que ambos componentes conviven en el mismo repositorio con frecuencia, el par UI-BFF puede verse como un único sistema cohesionado. Cuando el BFF ofrece consultas de datos fuertemente tipados, los resultados se pueden adaptar a las necesidades específicas del front-end, en lugar de utilizar una única entidad de propósito general que deba satisfacer las necesidades de muchos consumidores y contener más campos de los que realmente son necesarios. Esto reduce el riesgo de exponer accidentalmente datos que el usuario no debería ver, previene la interpretación incorrecta del objeto de datos devuelto y hace que la consulta sea más expresiva. Esta práctica es particularmente útil cuando se implementa con io-ts para hacer cumplir la seguridad de los tipos en tiempo de ejecución.

Plataformas delimitadas de poco código

Evaluar

Una de las decisiones más complejas a las que se enfrentan las compañías en este momento es la adopción de plataformas de poco o ningún código, es decir, plataformas que resuelven problemas muy específicos en dominios bastante limitados. Muchos proveedores están presionando agresivamente hacia este espacio. Los problemas que vemos con estas plataformas se relacionan típicamente con la imposibilidad de aplicar buenas prácticas de ingeniería como el versionamiento. Además, realizar pruebas es generalmente muy difícil. Sin embargo, hemos notados la existencia de algunos nuevos e interesantes participantes en el mercado, como Amazon Honeycode que facilita la creación de aplicaciones simples de gestión de tareas o eventos y Parabola para flujos de trabajo en la nube similares a los de IFTTT, por lo que estamos incluyendo a las plataformas delimitadas de poco código en esta edición del Radar. No obstante, seguimos siendo profundamente escépticos acerca de su mayor aplicabilidad, ya que estas herramientas tienen el poder de escapar de sus límites y enredarlo todo, como lo hace la maleza. Es por eso que seguimos aconsejando tener mucho cuidado en su adopción.

Identidad descentralizada

Evaluar

En 2016, Christopher Allen, uno de los contribuyentes principales a SSL/TLS, nos inspiró con una introducción de 10 principios que apuntaban a una nueva forma de identidad digital así como una forma de llegar a ella: el camino hacia la auto-identidad soberana. La auto-identidad soberana, también conocida como la identidad descentralizada , es una "identidad portable y para toda la vida de una persona, organización o cosa que no depende de una autoridad centralizada y que nunca se puede confiscar", de acuerdo al estándar Trust over IP. La adopción e implementación de identidades descentralizadas está ganando impulso y convirtiéndose en algo asequible. Vemos que se está adoptando en aplicaciones médicas, infraestructura sanitaria pública e identidad empresarial legal que respetan la privacidad. Si quieres comenzar pronto con la identidad descentralizada, puedes evaluar proyectos de código abierto como Sovrin Network, Hyperledger Aries e Indy, así como los estándares de identificadores descentralizados y de credenciales verificables. Estamos muy pendientes de lo que sucede en este campo mientras ayudamos a nuestros clientes con su posicionamiento estratégico en la nueva era de confianza digital.

Radiador del desfase de los despliegues

Evaluar

Un radiador del desfase de los despliegues hace visible las discrepancias entre las versiones del software desplegado en múltiples ambientes. Las organizaciones que utilizan despliegues automáticos suelen tener flujos de aprobación manual para los ambientes cercanos a producción, y con frecuencia, el código en estos ambientes está varias versiones por detrás del código en desarrollo actual. Esta técnica muestra esta diferencia entre las versiones de cada componente en cada ambiente mediante un tablero simple. Esto ayuda a destacar el costo de oportunidad del código terminado que aún no está en producción y puede señalar posibles riesgos, como parches de seguridad aún no liberados.

Cifrado homomórfico

Evaluar

El cifrado homomórfico (homomorphic encryption, HE) total se refiere a una clase de métodos de cifrado que permiten que los cálculos (como la búsqueda y

Técnicas

Las organizaciones que utilizan despliegues automatizados algunas veces requieren aprobaciones manuales para ambientes parecidos a producción, provocando que el código de este ambiente esté retrasado con respecto al código actual. Un Visor de desviación de despliegues logra hacer visible este retraso a través de un tablero de comandos simple.

(Visor de desviación de despliegues)

Técnicas

La computación segura de múltiples partes soluciona el problema de computación colaborativa que protege la privacidad entre partes que no confían entre las mismas, sin involucrar a una tercera parte.

(Computación segura de múltiples partes)

la aritmética) se realicen directamente sobre datos cifrados. El resultado de los cálculos permanece encriptado, y se puede desencriptar y revelar posteriormente. Aunque el problema de la HE se propuso por primera vez en 1978, no se construyó una solución sino hasta 2009. Con los avances en el poder computacional y la disponibilidad de bibliotecas de código abierto fáciles de usar, como SEAL, Lattigo, HElib y el cifrado homomórfico parcial en Python, HE se está volviendo factible en aplicaciones del mundo real. Los escenarios motivadores incluyen casos de uso con preservación de la privacidad, donde la computación se puede subcontratar a un grupo que no es de confianza, por ejemplo, para ejecutar cálculos sobre datos cifrados en la nube, o para permitir que un tercero agregue resultados intermedios cifrados homomórficamente de aprendizaje automático federado. Además, la mayoría de los esquemas de HE se consideran seguros frente a las computadoras cuánticas y se están realizando esfuerzos para estandarizar la HE. A pesar de sus limitaciones actuales, rendimiento y viabilidad de los tipos de cálculos, HE es digno de atención.

Hotwire

Evaluar

Hotwire (HTML over the wire) es una técnica para construir aplicaciones web. Las páginas se construyen a partir de componentes, pero a diferencia de las SPA modernas, el HTML de los componentes se genera en el lado del servidor y luego se envía por el cable "over the wire" al navegador. La aplicación sólo tiene una pequeña cantidad de código JavaScript en el navegador para unir los fragmentos de HTML. Nuestros equipos, y sin duda otros también, utilizaron esta técnica después de que las peticiones web asíncronas fueran soportadas por los navegadores, allá por el año 2005, pero por diversas razones esta técnica nunca ganó mucha fama.
En la actualidad, Hotwire utiliza tanto las capacidades modernas de los navegadores web como las capacidades de HTTP para lograr la velocidad, capacidad de respuesta y la naturaleza dinámica de las aplicaciones de una sola página (SPA, por sus siglas en inglés). Esta técnica adopta un diseño de aplicación web más sencillo, localizando la lógica en el servidor y manteniendo simple el código del lado del cliente. El equipo de Basecamp ha lanzado algunos marcos de trabajo de Hotwire que potencian su propia aplicación, incluyendo Turbo y Stimulus. Turbo incluye un conjunto de técnicas y marcos de trabajo para acelerar la capacidad de respuesta de la aplicación evitando la recarga de la página completa, la previsualización de la página desde la caché y la descomposición de la página en fragmentos con mejoras progresivas bajo demanda. Stimulus está diseñado para mejorar el HTML estático en el navegador conectando objetos JavaScript a los elementos de la página en el HTML.

Importmaps para micro frontends

Evaluar

Cuando se compone una aplicación con varios micro frontends, algunas partes del sistema necesitan decidir qué micro frontend cargar y de dónde hacerlo. Hasta ahora, o bien construíamos una solución a medida, o bien dependíamos de un marco de trabajo más amplio como single-spa. Ahora existe un nuevo estándar denominado import-maps que ayuda en ambos casos. Nuestras primeras experiencias muestran que usar importmaps para micro frontends permite conseguir una separación limpia de responsabilidades. El código JavaScript indica qué importar y una etiqueta script al inicio de la respuesta HTML inicial especifica de dónde hay que importar el front-end. Ese HTML está generado obviamente en el lado servidor, lo que hace posible usar alguna configuración dinámica durante la renderización. En muchos casos esta técnica nos recuerda a las rutas del enlazador/cargador (linker/loader) para bibliotecas dinámicas Unix. Por ahora los importmaps solo están soportados por Chrome, pero con el polyfill SystemJS están listos para un uso más amplio.

Modelo Abierto de Aplicaciones (OAM)

Evaluar

El Open Application Model (OAM) es un intento para crear algunos estándares alrededor del modelamiento de plataformas de infraestructura como productos. Usando abstracciones para los componentes, las configuraciones de aplicaciones, sus ámbitos (scopes) y atributos (traits), los equipos de desarrollo pueden describir sus aplicaciones de forma agnóstica a la plataforma; mientras que el equipo que la implemente, lo puede hacer en términos de carga de trabajo, atributos y ámbitos. Desde la última vez que hablamos sobre OAM, hemos seguido con interés una de sus primeras implementaciones: KubeVela. KubeVela está cerca de publicar su versión 1.0 y tenemos curiosidad de ver si este tipo de implementaciones pueden sustentar la promesa de la idea de OAM.

Analítica web centrada en la privacidad

Evaluar

La analítica web centrada en la privacidad es una técnica para recopilar datos analíticos web sin comprometer la privacidad del usuario final, manteniéndolos verdaderamente anónimos. Una

consecuencia inesperada del cumplimiento de la Regulación General de Protección de Datos (RGPD o, GDPR en inglés) es la decisión tomada por varias organizaciones de degradar la experiencia del usuario con complejos procesos para aceptar cookies, especialmente cuando el usuario no acepta inmediatamente la configuración por defecto de "todas las cookies". La analítica web centrada en la privacidad tiene el doble beneficio de respetar el espíritu y la letra de la RGPD a la vez que evita la necesidad de introducir formularios intrusivos de consentimiento de cookies. Una implementación de esta técnica es Plausible.

Programación en grupo remota

Evaluar

La programación en grupo (mob programming) es una de esas técnicas que nuestros equipos han visto que son capaces de ejecutar más fácilmente de forma remota. La programación en grupo remota permite a los equipos agruparse alrededor de una tarea o una sección de código sin las limitantes físicas de solo poder ubicar un número limitado de personas en una estación de trabajo. Los equipos pueden colaborar rápidamente sin tener que conectar una gran pantalla, reservar una sala de reuniones física o encontrar una pizarra.

Computación segura entre múltiples partes

Evaluar

La computación segura entre múltiples partes (secure multiparty computing, MPC) soluciona el problema de computación colaborativa protegiendo la privacidad entre partes que no confían entre sí. Su objetivo es calcular de forma segura un problema acordado sin un tercero de confianza, mientras cada participante es requerido de formar parte del cálculo de un resultado, que no puede ser obtenido por las otras entidades. Un ejemplo simple de MCP es el problema de los millonarios: dos millonarios quieren conocer quién es más rico, pero ninguno quiere revelar el valor de su patrimonio al otro ni tampoco a un tercero. Los enfoques de implementación de MPC varían: los escenarios pueden incluir el intercambio de secretos, transferencia inconsciente, circuitos confusos o cifrado homomórfico. Algunas soluciones comerciales de MPC que han aparecido recientemente, como Antchain Morse, afirman que ayudan a resolver los problemas del intercambio de secretos y el aprendizaje automático seguro en escenarios como la investigación crediticia conformada por múltiples partes y el intercambio de datos de registros médicos. Aunque estas plataformas son atractivas desde una perspectiva de mercadotecnia, todavía tenemos que ver si son realmente útiles.

GitOps

Resistir

Sugerimos usar GitOps con cierto grado de cuidado, especialmente con lo que respecta a las estrategias de ramificación en repositorios de código. GitOps se puede ver como una forma de implementar infrastructura como código que implica la sincronización y aplicación continua de código de infraestructura desde Git en varios ambientes. Cuando se usa con una estrategia de "rama por ambiente", los cambios se promueven de un ambiente al siguiente mediante la combinación (merge) del código. Si bien tratar el código como única fuente de la verdad es un enfoque sólido, vemos que la estrategia de "rama por entorno" típicamente da pie a que aparezcan diferencias entre ambientes y que configuraciones específicas se propaguen a medida que las operaciones de combinación del código (merge) se vuelven problemáticas o se dejan de hacer. Esto es muy similar a lo que advertimos en el pasado respecto a las ramas de larga duración con GitFlow.

Equipos de plataforma en capas

Resistir

La explosión de interés en torno a las plataformas de software ha creado mucho valor para las organizaciones, pero el camino hacia la construcción de un modelo de entrega basado en plataformas está plagado de posibles callejones sin salida. Por la emoción que traen los nuevos paradigmas, es común ver un resurgimiento de técnicas antiguas marcadas con la nueva lengua vernácula, lo que hace que sea fácil perder de vista las razones por las que superamos esas técnicas en primer lugar. Para ver un ejemplo de este cambio de marca, revisa nuestro blip sobre los ESBs disfrazados de API Gateways en la edición anterior del Radar. Otro ejemplo que estamos viendo es repetir el enfoque de dividir a los equipos por capa de tecnología, pero llamándolos plataformas. En el contexto de la construcción de una aplicación, era común tener un equipo de front-end separado del equipo que se encarga de la lógica empresarial, separado del equipo de datos, y vemos análogos a ese modelo cuando las organizaciones segregan las capacidades de la plataforma entre los equipos dedicados a una capa de negocio o de datos. Gracias a la Ley de Conway, sabemos que organizar equipos de capacidades de plataforma en torno a capacidades comerciales es un modelo más eficaz, que permite al equipo empoderarse de la capacidad completamente, incluyendo los datos. Esto ayuda a evitar los dolores de cabeza de la gestión de dependencias de equipos de plataforma en capas , con el equipo de front-end esperando al equipo de la lógica de negocio, esperando al equipo de datos para hacer cualquier cosa.

Técnicas

Ciertas organizaciones consideran que la revisión por pares es equivalente a un pull request. Hemos visto que este enfoque crea importantes embotellamientos en el equipo y además degrada significativamente la calidad de la retroalimentación.

(Revisión por pares equivalente a pull request)

Requisitos ingenuos de complejidad de contraseñas

Resistir

En la actualidad, las políticas de contraseñas son un estándar por defecto para muchas organizaciones. Sin embargo, seguimos viendo que hay organizaciones que solicitan que las contraseñas incluyan una variedad de símbolos, números, letras mayúsculas y minúsculas, e incluso caracteres especiales. Estos son requisitos ingenuos de complejidad de contraseñas y provocan una falsa sensación de seguridad, ya que los usuarios optarán por contraseñas más inseguras porque cualquier otra cosa es difícil de recordar y escribir. De acuerdo a las recomendaciones del NIST, el factor principal en la fuerza de una contraseña es su longitud, y por lo tanto, los usuarios deberían elegir frases de contraseña largas con un requisito máximo de 64 caracteres (incluyendo espacios). Estas frases de contraseña son más seguras y fáciles de recordar.

Revisión por pares equivalente a pull request

Resistir

Parece que ciertas organizaciones consideran que la revisión por pares es equivalente a un pull request : creen que la única manera de lograr una revisión por pares del código es a través de un pull request. Hemos visto que este enfoque crea importantes embotellamientos en el equipo y además degrada significativamente la calidad de la retroalimentación cuando los revisores, sobrecargados, comienzan a rechazar las solicitudes casi sin mirar. Aunque se podría argumentar que esta es una forma de demostrar el "cumplimiento normativo" de la revisión de código, en uno de nuestros clientes se dijo que esto era inválido, ya que no había evidencia de que el código había sido leído por alguien antes de su aceptación. Los pull requests son solo una forma de administrar el flujo de trabajo de revisión de código por lo que instamos a las personas a considerar otros enfoques, especialmente cuando es necesario capacitar y transmitir retroalimentación con cuidado.

SAFe™

Resistir

Nuestro posicionamiento respecto a "ser ágil antes de hacer ágil" y nuestras opiniones sobre este tema no deberían causar sorpresa; pero desde que SAFe™ (Scaled Agile Framework®), según el informe de Gartner de Mayo de 2019, es el marco de referencia ágil empresarial más considerado y utilizado, y dado que vemos a más y más empresas emprender cambios organizacionales, hemos pensado que es el momento de volver a crear conciencia sobre este tema. Nos hemos encontrado con organizaciones luchando con los procesos sobre-estandarizados y de fases que tiene SAFe. Esos procesos crean fricción en la estructura organizacional y su modelo operativo. También, puede promover la creación de silos en la organización, impidiendo que las plataformas se conviertan en auténticas habilitadoras de las capacidades del negocio. El control "de arriba hacia abajo" genera desperdicio en la cadena de valor y desalienta la creatividad del talento de ingeniería, al tiempo que limita la autonomía y la experimentación en los equipos. En vez de medir el esfuerzo y focalizarse en ceremonias estandarizadas, recomendamos una aproximación y gobernanza más ligeras y enfocadas en la entrega de valor, para así ayudar a eliminar la fricción organizativa, como EDGE, y evaluar la carga cognitiva de los equipos para identificar sus tipos y determinar como deben interactuar mejor entre sí.

Scaled Agile Framework® y SAFe™ son marcas registradas de Scaled Agile, Inc.

Propiedad separada del código y del pipeline

Resistir

Idealmente, el pipeline de despliegue y el código que se despliega deben ser propiedad del mismo equipo, especialmente cuando los equipos practican DevOps. Desafortunadamente, todavía vemos organizaciones donde existe una propiedad separada del código y del pipeline , donde este último pertenece al equipo de infraestructura. Esto se traduce en lentitud para aplicar cambios, en la existencia de barreras para las mejoras y en una falta de apropiación por parte del equipo de desarrollo, con menor involucramiento en los despliegues. Una causa de esto puede ser claramente el tener equipos separados; otra puede ser el deseo de conservar procesos y roles que funcionen como controladores y guardianes. Aunque puede haber razones legítimas para usar este enfoque (por ejemplo, por control regulatorio), en general encontramos que es doloroso e inútil.

Modelos operativos de plataforma basados en tickets

Resistir

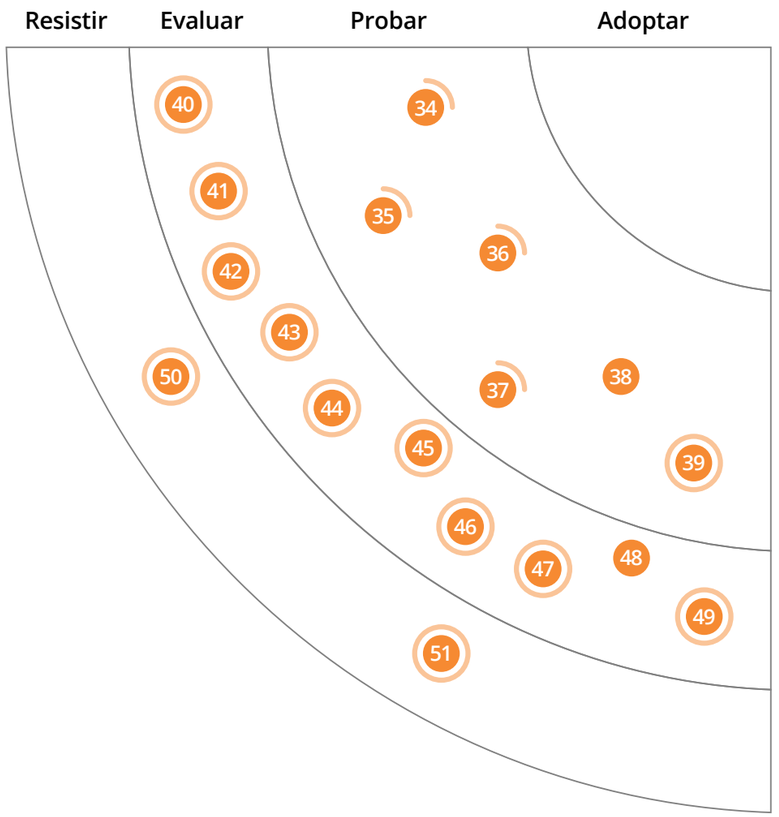
Uno de los objetivos finales de una plataforma debe ser reducir los procesos basados en tickets a un mínimo absoluto ya que crean colas en el flujo de valor. Lamentablemente, todavía vemos organizaciones que no presionan con la suficiente fuerza para conseguir esa meta, lo que resulta en modelos operativos de plataforma basados en tickets. Esto es particularmente frustrante cuando los procesos basados en tickets se colocan en frente de plataformas de los proveedores de la nube que tienen características de autoservicio y orientadas a las APIs. Es difícil pero no necesario lograr el autoservicio con muy pocos tickets desde el comienzo, pero debe ser el objetivo.

La dependencia excesiva en la burocracia y la falta de confianza son las causas de esta resistencia a alejarse de los procesos basados en tickets. Incorporar más verificaciones y alertas automáticas dentro de la plataforma ayuda a cortar el cordón de los procesos de aprobación con tickets. Por ejemplo, se puede dar visibilidad de los costos de ejecución a los equipos y colocar límites automáticos para evitar una explosión accidental de los costos. Recomendamos implementar políticas de seguridad como código y utilizar escáneres de configuración o analizadores como Recommender para ayudar a los equipos a hacer lo correcto.

TECHNOLOGY RADAR

Plataformas

Plataformas



Kit de desarrollo en la nube de AWS

Probar

Muchos de nuestros equipos que ya están en AWS han encontrado que el kit de desarrollo en la nube de AWS (CDK de AWS) es una elección razonable predeterminada para permitir el aprovisionamiento de la infraestructura. En concreto, a los equipos les gusta el uso de lenguajes de programación de primera clase en lugar de archivos de configuración, lo que les permite utilizar herramientas, habilidades y enfoques de pruebas ya existentes. Al igual que otras herramientas similares, se debe tener cuidado para garantizar que los despliegues continúan siendo fáciles de entender y mantener. El CDK es actualmente compatible con TypeScript, JavaScript, Python, Java, C# y .NET. Se están añadiendo nuevos proveedores al núcleo del CDK. Hemos utilizado con éxito tanto el CDK de AWS como el kit de desarrollo en la nube para Terraform de HashiCorp para generar configuraciones de Terraform y permitir el aprovisionamiento exitoso en esta plataforma.

Backstage

Probar

Seguimos viendo interés en Backstage y crecimiento en su uso, además de la adopción de portales para desarrolladoras, dada por la necesidad de las organizaciones de brindar soporte y optimizar sus ambientes de desarrollo. A medida que crece el número de herramientas y tecnologías, también aumenta la importancia de lograr alguna forma de estandarización para conseguir algo de consistencia, de forma que los equipos de desarrollo puedan concentrarse en la innovación y el desarrollo de los productos en lugar de reinventar la rueda. Backstage es una plataforma de código abierto para levantar portales para desarrolladoras creada por Spotify; se basa en plantillas de software, y permite unificar las herramientas de infraestructura con documentación técnica consistente y centralizada. Su arquitectura para complementos permite extenderla y adaptarla a las necesidades del ecosistema de infraestructura de una organización.

Delta Lake

Probar

Delta Lake es una capa de almacenamiento de código abierto implementada por Databricks, que intenta llevar transacciones ACID al procesamiento de big data. En proyectos de lago de datos o de malla de datos con soporte de Databricks, nuestros equipos siguen prefiriendo usar el almacenamiento Delta Lake en lugar del uso directo de mecanismos de almacenamiento de archivos como S3 o ADLS. Por supuesto que esto se limita

Adoptar

Probrar
34. Kit de desarrollo en la nube de AWS
35. Backstage
36. Delta Lake
37. Materialize
38. Snowflake
39. Fuentes variables

Evaluar
40. Apache Pinot
41. Bit.dev
42. DataHub
43. Feature Store
44. JuiceFS
45. API de Kafka sin Kafka
46. NATS
47. Opstrace
48. Pulumi
49. Redpanda

Resistir
50. Azure Machine Learning
51. Productos hechos en casa para infraestructura como código.

Plataformas

LinkedIn ha evolucionado WhereHows en DataHub, la plataforma de nueva generación que aborda la capacidad de descubrimiento de datos a través de un sistema extensible de metadatos.

(DataHub)

a proyectos que usan plataformas de almacenamiento que soportan Delta Lake cuando usan formatos de archivo Parquet. Delta Lake facilita casos de uso de lectura/escritura de datos concurrentes donde se requiere transaccionalidad a nivel de archivo. Encontramos de gran ayuda a la integración transparente de Delta Lake con las APIs de procesamiento en lotes o en micro lotes de Apache Spark, y particularmente, a funcionalidades como los viajes en el tiempo (acceder a los datos de un momento determinado o en la reversión de un commit) así como el soporte de evolución de esquema al momento de escritura, aunque hay algunas limitaciones en estas características.

Materialize

Probar

Materialize es una base de datos en streaming que permite realizar cálculos incrementales sin necesidad de pipelines de datos complicados. Se debe describir los cálculos mediante vistas SQL estándar y conectar Materialize al stream de datos. El motor de flujo de datos diferencial subyacente realiza cálculos incrementales para proporcionar resultados consistentes y correctos con mínima latencia. A diferencia de las bases de datos tradicionales, no hay restricciones para definir estas vistas y los cálculos se ejecutan en tiempo real. Nosotros hemos utilizado Materialize junto con Spring Cloud Stream y Kafka en un sistema de eventos distribuidos para hacer consultas sobre streams de eventos, y nos gustó esta configuración.

Snowflake

Probar

Desde la última vez que presentamos a Snowflake en el Radar, hemos ganado algo más de experiencia con él, así como con las mallas de datos como alternativa a los almacenes de datos (data warehouses) y lagos de datos. Snowflake continua impresionándonos con funcionalidades como viajes en el tiempo, clonación de copia cero, intercambio de datos y con su tienda (marketplace). No hemos encontrado nada que no nos guste, lo que ha llevado a los equipos a preferirlo por sobre otras alternativas. Redshift se está enfocando en la separación del almacenamiento y del procesamiento, que ha sido un punto fuerte de Snowflake, pero, incluso con Redshift Spectrum, no es tan fácil y ni flexible de usar, en parte por sus lazos previos con Postgres (por cierto, todavía nos gusta Postgres). Las consultas federadas pueden ser una razón para usar Redshift, pero, cuando se trata de operaciones, Snowflake es mucho más fácil de usar. BigQuery, que es otra alternativa, es muy fácil de operar pero Snowflake lo supera en ambientes de múltiples nubes (multi-cloud). También podemos decir que hemos usado Snowflake con éxito en GCP, AWS y Azure.

Fuentes variables

Probar

Las fuentes variables son una forma de evitar la necesidad de buscar e incluir archivos de fuentes distintos para diferentes pesos y estilos. Todo está en un solo archivo y puedes utilizar propiedades para seleccionar el tipo de estilo y el peso que se necesita. Si bien no es nuevo, aún vemos sitios y proyectos que se podrían beneficiar de este enfoque tan simple. Si tienes páginas que incluyen diferentes variaciones de la misma fuente, te sugerimos que pruebes fuentes variables.

Apache Pinot

Evaluar

Apache Pinot es un almacén de datos distribuidos OLAP creado para ofrecer analíticas en tiempo real y con baja latencia. Puede alimentarse de fuentes de datos por lotes (como Hadoop HDFS, Amazon S3, Azure ADLS o Google Cloud Storage), así como de orígenes de datos en streams (como Apache Kafka). Si se necesita ofrecer al usuario análiticas de baja latencia, las soluciones de SQL-en-Hadoop no ofrecen la latencia que se necesita. Los motores OLAP modernos, como Apache Pinot (o Apache Druid y Clickhouse, entre otros), pueden lograr una latencia mucho menor y son particularmente adecuados en contextos donde se necesitan analíticas rápidas en tiempo real y con datos inmutables, como las agregaciones. Originalmente construido por LinkedIn, Apache Pinot ingresó a la incubadora de Apache a finales de 2018 y desde entonces ha obtenido una arquitectura de complementos y compatibilidad con SQL, entre otras capacidades clave. Apache Pinot puede ser bastante complejo de operar y tiene muchas partes móviles, pero recomendamos evaluarlo si los volúmenes de datos son lo suficientemente grandes y se necesita una capacidad de consulta de baja latencia.

Bit.dev

Evaluar

Bit.dev es una plataforma colaborativa alojada en la nube para componentes de interfaz de usuario extraídos, modularizados y reutilizados con Bit. Los componentes web han existido ya por un tiempo, pero nunca ha sido sencillo construir aplicaciones frontend modernas ensamblando componentes independientes y pequeños, extraídos de otros proyectos. Bit fue

diseñada para hacer exactamente eso: extraer un componente de una biblioteca o proyecto existente. Tu puedes construir tu servicio propio para la colaboración de componentes o usar Bit.dev.

DataHub
Evaluar

Desde que mencionamos al descubrimiento de datos por primera vez en el Radar, LinkedIn ha evolucionado WhereHows en DataHub, la plataforma de nueva generación que aborda la capacidad de descubrimiento de datos a través de 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 un API o un stream hacia la plataforma central. Este modelo de integración traslada la propiedad de la entidad central a los equipos individuales, haciéndolos responsables de sus metadatos. A medida que más y más empresas intentan orientarse a los datos, es fundamental tener un sistema que ayude con el descubrimiento y la comprensión de la calidad y el linaje de los datos, por lo que recomendamos evaluar a DataHub en esa capacidad.

Feature Store
Evaluar

Feature Store es una plataforma de datos específica para aprendizaje automático que resuelve algunos de los principales retos que encontramos hoy en la ingeniería de características con tres capacidades fundamentales: (1) usa pipelines de datos gestionados para eliminar conflictos con los pipelines cuando llegan nuevos datos; (2) cataloga y almacena datos de las características para fomentar su visibilidad y colaboración entre modelos; y (3) consistentemente provee datos de características durante el entrenamiento y la interferencia.
Desde que Uber reveló su plataforma Michelangelo, muchas organizaciones y startups han construido sus propias versiones de almacenes de características, como Hopsworks, Feast y Tecton. Vemos potencial en Feature Store y recomendamos realizar un análisis cuidadoso.

JuiceFS
Evaluar

JuiceFS es un sistema de archivos POSIX distribuido, de código abierto, basado en Redis y, al mismo tiempo, un almacén de objetos. A la hora de crear nuevas aplicaciones, normalmente recomendamos interactuar directamente con el almacén de objetos, sin utilizar una capa de abstracción adicional. Sin embargo, JuiceFS podría ser una opción si se necesita migrar a la nube a aplicaciones legadas que dependen de sistemas de archivos POSIX tradicionales.

API de Kafka sin Kafka
Evaluar

A medida que más empresas recurren a los eventos como medio para compartir datos entre microservicios, recopilar analíticas o alimentar lagos de datos, Apache Kafka se ha convertido en la plataforma favorita para sostener una arquitectura basada en eventos. Aunque Kafka fue un concepto revolucionario en mensajería persistente y escalable, se requieren muchas partes móviles para que funcione, como ZooKeeper, brokers, particiones y réplicas. Si bien estos pueden ser particularmente difíciles de implementar y operar, ofrecen una gran flexibilidad y potencia cuando es necesario, especialmente a escala empresarial industrial. Debido a la alta barrera de entrada que presenta el ecosistema completo de Kafka, nos complace la reciente explosión de plataformas que ofrecen el API de Kafka sin Kafka. Entradas recientes como Kafka en Pulsar y Redpanda ofrecen arquitecturas alternativas y Azure Event Hubs para Kafka proporciona cierta compatibilidad con las APIs de producción y consumo de Kafka. Algunas características de Kafka, como la biblioteca cliente de streams, no son compatibles con estos brokers alternativos, por lo que todavía hay razones para elegir Kafka en su lugar. Sin embargo, queda por ver si las desarrolladoras realmente adoptan esta estrategia o si es simplemente un intento de los competidores para alejar a los usuarios de Kafka. En última instancia, quizás el impacto más duradero de Kafka podría ser su conveniente protocolo y el API que se brinda a los clientes.

NATS
Evaluar

NATS es un sistema de cola de mensajería rápido y seguro con una gama inusualmente amplia de características y potenciales sistemas objeto de despliegue. En principio, pasaremos por alto el que uno se pregunte por qué el mundo necesita otro sistema de colas de mensajería. Estas han estado presentes en varias formas casi todo el tiempo desde que las empresas han usado computadoras y han experimentado años de refinamiento y optimización para diversas tareas. Sin embargo, NATS tiene características interesantes y es único por su capacidad para escalar desde controladores integrados hasta super clusters globales alojados en la nube. Nos intriga particularmente la intención de NATS de soportar flujos continuos de datos desde dispositivos móviles e IoT, y a través de una red de sistemas interconectados.

Plataformas

A medida que mayor número de negocios recurren a los eventos como una forma de compartir datos entre microservicios, recopilar análisis o alimentar data lakes, Apache Kafka se ha convertido en la plataforma favorita. Debido a la alta barrera de entrada que presenta el ecosistema completo de Kafka, damos la bienvenida a la reciente explosión de plataformas que ofrecen la API de Kafka sin Kafka.

(Kafka API sin Kafka)

Plataformas

De vez en cuando las organizaciones tienden a construir frameworks o abstracciones encima de productos externos existentes para cubrir necesidades muy específicas, pensando que la adaptación conlleva más beneficios. Sin embargo, estas organizaciones subestiman el esfuerzo necesario para seguir evolucionando dichas soluciones al ritmo de sus necesidades, y tras un breve periodo de tiempo, se dan cuenta de que la versión original está en mucho mejor condición que la suya.

(Productos de infraestructura como código caseros)

Sin embargo, es necesario abordar algunos problemas delicados, uno de los cuales es garantizar que los consumidores solo vean los mensajes y tópicos a los que se les permite acceder, especialmente cuando la red atraviesa los límites de la organización. NATS 2.0 introdujo un esquema de seguridad y control de acceso que soporta clusters para múltiples propietarios (multitenant) donde las cuentas restringen el acceso de un usuario a las colas y a los tópicos. Escrito en Go, NATS ha sido adoptado principalmente por la comunidad de este lenguaje. Aunque existen clientes para casi todos los lenguajes de programación más comunes, el cliente Go es el más popular. Sin embargo, algunas de nuestras desarrolladoras han descubierto que todas las bibliotecas cliente tienden a reflejar sus orígenes en Go. El aumento del ancho de banda y la potencia de procesamiento de los dispositivos inalámbricos pequeños significa que el volumen de datos que las empresas deben consumir en tiempo real solo aumentará. Se puede evaluar a NATS como una posible plataforma para transmitir esos datos dentro y entre las empresas.

Opstrace

Evaluar

Opstrace es una plataforma de código abierto de observabilidad, destinada para ser desplegada en la propia red del usuario. Si no usamos soluciones comerciales como Datadog (por temas de costo o de almacenamiento de datos, por ejemplo), la solución que nos queda sería construir nuestra propia plataforma compuesta de herramientas de código abierto. Esto puede suponer mucho esfuerzo, así que Opstrace está diseñado para cubrir esta brecha. Utiliza APIs e interfaces de código abierto, como Prometheus y Grafana, y agrega funcionalidades adicionales como TLS y autenticación. En el corazón de Opstrace se encuentra un clúster de Cortex para proporcionar el API escalable de Prometheus, así como un cluster de Loki para los logs. Como es una herramienta relativamente nueva, aún carece de ciertas funcionalidades si se compara con soluciones como Datadog o SignalFX. No obstante, es una opción prometedora que contribuye a este espacio y vale la pena tomarla en cuenta.

Pulumi

Evaluar

Hemos visto un creciente pero lento interés en Pulumi. Esta utilidad llena una brecha en el mundo de la infraestructura como código, donde Terraform está muy afianzado. Si bien Terraform ha sido ampliamente usado y probado, su naturaleza declarativa presenta capacidades inadecuadas de abstracción y carece de la capacidad de ser probado. Terraform es adecuado cuando la infraestructura es completamente estática, pero para la definición de infraestructuras dinámicas se necesita un lenguaje de programación real. Pulumi se distingue por permitir que las configuraciones se escriban en TypeScript o JavaScript, Python y Go, sin ser necesario un lenguaje de marcado o de plantillas. Pulumi está altamente enfocado en arquitecturas nativas a la nube, incluyendo contenedores, funciones serverless y servicios de datos, proporcionando buen soporte para Kubernetes. Recientemente, el CDK de AWS se ha posicionado como contendor, pero Pulumi se mantiene como la única herramienta neutral del área, en lo que respecta a los proveedores. Anticipamos una amplia adopción de Pulumi en el futuro y esperamos la aparición de herramientas viables y de ecosistemas de conocimiento que le den soporte.

Redpanda

Evaluar

Redpanda es una plataforma de streaming de datos que proporciona un API compatible con Kafka, lo que permite aprovechar el ecosistema de Kafka sin tener que lidiar con las complejidades de su instalación. Por ejemplo, operar Redpanda es sencillo porque se distribuye como un archivo ejecutable solo y no requiere de una dependencia externa como ZooKeeper. En su lugar, implementa el protocolo Raft y realiza pruebas integrales para validar si ha sido implementado correctamente. Una de las capacidades de RedPanda (solo para clientes empresariales) son las transformaciones WASM en línea, utilizando su motor WebAssembly (WASM) integrado. Esto permite a las desarrolladoras crear transformadores de eventos en su lenguaje preferido y compilarlos a WASM. También ofrece latencias de cola muy reducidas y mayor rendimiento debido a una serie de optimizaciones. Redpanda es una alternativa fascinante a Kafka y vale la pena probarla.

Azure Machine Learning

Resistir

Ya hemos visto antes que los proveedores de la nube publican cada vez más y más servicios al mercado. También hemos documentado nuestras preocupaciones de que en ocasiones los servicios liberados al público no están del todo listos. Lamentablemente, en base a nuestra experiencia, Azure Machine Learning entra en esta última categoría. Como uno de los participantes más recientes en el campo de las plataformas delimitadas de poco código, Azure ML promete más facilidades para los científicos de datos. Sin embargo no cumple su promesa y, de hecho, nuestras científicas de datos sienten que sigue siendo más sencillo trabajar

en Python. A pesar de algunos esfuerzos significativos, tuvimos dificultades para hacer escalar la solución y la falta de documentación adecuada demostró ser otro problema. Por estas razones la movemos al anillo “Resistir”.

Productos hechos en casa para infraestructura como código

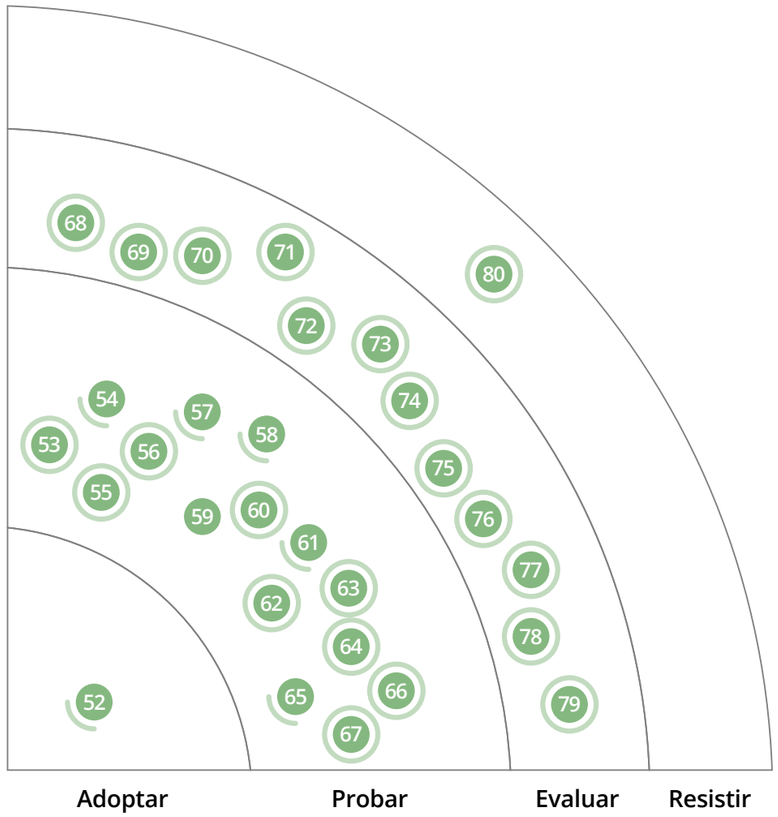
Resistir

Los productos respaldados por compañías o comunidades están en constante evolución, al menos aquellos que ganan tracción en la industria. De vez en cuando las organizaciones tienden a construir marcos de trabajo o abstracciones encima de productos externos existentes para cubrir necesidades muy específicas, pensando que sus adaptaciones traerán más beneficios que los productos por sí solos. Vemos a organizaciones intentando crear productos hechos en casa para infraestructura como código sobre tecnologías ya establecidas. Estas organizaciones subestiman el esfuerzo necesario para seguir evolucionando dichas soluciones al ritmo de sus necesidades, y tras un breve periodo de tiempo, se dan cuenta de que la versión original está en mucho mejor condición que la suya propia; incluso hay casos donde la abstracción encima del producto externo limita las funcionalidades originales. Aunque hemos visto casos de éxito de organizaciones construyendo soluciones hechas en casa, queremos advertir en contra de esta estrategia, ya que el esfuerzo que se requiere no es despreciable, y es necesario tener una visión de producto a largo plazo para conseguir las metas deseadas.

TECHNOLOGY RADAR

Herramientas

Herramientas



Sentry
Adoptar

Sentry se ha convertido en la elección predeterminada para muchos de nuestros equipos a la hora de reportar errores de frontend. La utilidad de características como la agrupación de errores o poder definir patrones para descartar errores con ciertos parámetros ayuda a gestionar la avalancha de errores que llegan de muchos dispositivos de usuarios finales. La integración de Sentry en el pipeline de despliegue continuo permite cargar mapas de orígenes para una depuración de errores más eficiente. Aunque Sentry se ofrece principalmente como un SaaS, también valoramos que su código fuente está disponible públicamente y se puede usar gratis en casos de uso más pequeños y en instalaciones auto-hospedadas.

axe-core
Probar

Hacer que la web sea inclusiva requiere de mucha atención para asegurar que la accesibilidad sea considerada y validada durante todas las fases de la entrega de software. Muchas de las herramientas populares para pruebas de accesibilidad son diseñadas para efectuar sus validaciones una vez que la aplicación está completa; como resultado, los problemas se detectan tarde y a menudo se dificulta su corrección, por lo que se acumulan como deuda. En nuestro reciente trabajo interno en los sitios web de Thoughtworks, hemos incluido como parte del pipeline a axe-core, un motor de pruebas de accesibilidad de software libre. Empezó a proporcionar una retroalimentación temprana a los miembros del equipo sobre la adherencia a las reglas de accesibilidad, incluso durante las primeras adiciones incrementales de funcionalidad. Sin embargo, no todos los problemas pueden ser encontrados mediante la inspección automatizada. Para extender a axe-core, existe la herramienta comercial axe DevTools, que incluye características que guían a los miembros del equipo en la realización de pruebas exploratorias para la mayoría de problemas de accesibilidad.

dbt
Probar

Desde la última vez que escribimos sobre dbt, lo hemos utilizado en algunos proyectos y nos gusta lo que hemos visto. Por ejemplo, dbt hace que la parte de transformación de los pipelines ETL sea más accesible para los consumidores de los datos en vez de solo para los ingenieros de datos que los construyen. Al mismo tiempo, fomenta la aplicación de buenas prácticas de ingeniería, como el versionamiento, las pruebas automatizadas y los despliegues. SQL sigue siendo la lengua franca del mundo de los datos (incluyendo bases de

Adoptar
52. Sentry

Probar
53. axe-core
54. dbt
55. esbuild
56. Flipper
57. Great Expectations
58. k6
59. MLflow
60. OR-Tools
61. Playwright
62. Prowler
63. Pyright
64. Redash
65. Terratest
66. Tuple
67. Why Did You Render

Evaluar
68. Buildah y Podman
69. GitHub Actions
70. Imagen nativa de Graal
71. HashiCorp Boundary
72. imgcook
73. Longhorn
74. Operator Framework
75. Recommender
76. Remote - WSL
77. Spectral
78. Yelp detect-secrets
79. Zally

Resistir
80. AWS CodePipeline

Herramientas

Las herramientas OR (en inglés, OR Tools) son un paquete de software de código abierto para resolver problemas de optimización combinatoria. Estos problemas de optimización tienen un gran conjunto de soluciones posibles y herramientas como esta son de gran ayuda al buscar la mejor solución.

(Herramientas OR)

datos, almacenes de datos, motores de consulta, lagos de datos y plataformas de analítica) y la mayoría de estos sistemas lo soportan en cierta medida. Esto permite que dbt sea utilizado frente a estos sistemas para realizar transformaciones con la sola construcción de adaptadores. El número de conectores nativos ha crecido e incluyen aquellos para Snowflake, BigQuery, Redshift y Postgres, al igual que la gama de complementos de la comunidad. Vemos que las herramientas como dbt ayudan a que las plataformas de datos brinden más capacidades de auto-servicio.

esbuild
Probar

Siempre nos ha llamado la atención encontrar herramientas que puedan acortar el ciclo de retroalimentación en el desarrollo de software, y esbuild es un buen ejemplo. A medida que la base de código de una aplicación de front-end crece más y más, generalmente experimentamos tiempos de empaquetado que se cuentan en minutos. esbuild, como un empaquetador de JavaScript optimizado para proveer mayor velocidad, puede reducir estos tiempos en factores de 10 a 100. Está escrito en Golang y utiliza un enfoque más eficiente en los procesos de análisis sintáctico, escritura y generación de mapas de código fuente, que superan significativamente a los de herramientas como Webpack y Parcel. Puede ser que esbuild no sea tan completa como esas herramientas para la transformación de sintaxis de JavaScript, sin embargo esto no impide que muchos de nuestros equipos se cambien a esbuild y la usen como su herramienta preferida.

Flipper
Probar

Flipper es un depurador extensible para aplicaciones móviles. De manera predeterminada, soporta la elaboración de perfiles, inspección interactiva del diseño, visor de registros y un inspector de red para aplicaciones iOS, Android y React Native. En comparación a otras herramientas de depuración para aplicaciones móviles, encontramos que Flipper es ligero, tiene valiosas características y es de fácil configuración.

Great Expectations
Probar

Habíamos escrito sobre Great Expectations en la edición anterior del Radar. Nos encanta y hemos decidido moverlo al anillo “Probar” en esta edición. Great Expectations es un marco de trabajo que permite crear controles que etiquetan anomalías o problemas de calidad en los pipelines de datos. Igual que la ejecución de una prueba unitaria en un pipeline de compilación, Great Expectations realiza verificaciones durante la ejecución del pipeline de datos. Nos gusta su simplicidad y facilidad de uso: las reglas almacenadas en formato JSON pueden ser modificadas por nuestros expertos del dominio de datos sin necesidad de tener habilidades de ingeniería de datos.

k6
Probar

Hemos trabajado un poco más en pruebas de rendimiento con k6 desde que lo cubrimos por primera vez en el Radar, y hemos conseguido buenos resultados. Nuestros equipos han apreciado el enfoque en la experiencia para las desarrolladoras y la flexibilidad de la herramienta. Aunque es fácil comenzar con k6 por sí solo, realmente destaca por su facilidad de integración en un ecosistema de desarrollo. Por ejemplo, utilizando el adaptador Datadog, un equipo pudo visualizar rápidamente el rendimiento en un sistema distribuido e identificar importantes problemas antes de lanzar el sistema a producción. Otro equipo, con la versión comercial de k6, pudo usar la extensión de Azure pipelines marketplace para realizar pruebas de rendimiento en su pipeline de integración continua y obtener informes de Azure DevOps con poco esfuerzo. Dado que k6 admite umbrales que permiten tener verificaciones de pruebas automatizadas listas para usar, es relativamente fácil agregar una etapa al pipeline que detecte la degradación del rendimiento de los nuevos cambios, agregando un poderoso mecanismo de retroalimentación para el equipo de desarrollo.

MLflow
Probar

MLflow es una herramienta de código abierto para el seguimiento de experimentos de aprendizaje automático y la gestión del ciclo de vida. El flujo de trabajo para desarrollar y evolucionar continuamente un modelo de aprendizaje automático incluye una serie de experimentos (una colección de ejecuciones), el seguimiento del rendimiento de estos experimentos (una colección de métricas) y el seguimiento y ajuste de modelos (proyectos). MLflow facilita este flujo de trabajo muy bien al apoyar los estándares abiertos existentes y se integra bien con muchas otras herramientas en el ecosistema. MLflow, como servicio gestionado por Databricks en la nube, disponible en AWS y Azure, está madurando rápidamente y lo hemos utilizado con éxito en nuestros proyectos. Consideramos que MLflow es una gran herramienta para la gestión y el seguimiento de modelos, que

soporta tanto modelos de interacción basados en UI como en API. Nuestra única y creciente preocupación es que MLflow está tratando de entregar respuestas a demasiados temas vinculados, como una sola plataforma, como el servicio de modelos y la puntuación.

OR-Tools
Probar

OR-Tools es una suite de software de código abierto para resolver problemas de optimización combinatoria. Estos problemas de optimización tienen un conjunto muy grande de posibles soluciones y las herramientas como OR-Tools son de gran ayuda para buscar la mejor solución. Se puede modelar el problema en cualquiera de los lenguajes soportados (Python, Java, C# o C++) y elegir el solucionador de entre los muchos solucionadores soportados, tanto de código abierto como comerciales. Hemos usado con éxito OR-Tools en muchos proyectos de optimización con programación de enteros y de enteros mixta.

Playwright
Probar

Con Playwright es posible escribir pruebas de interfaz de usuario web para Chromium, Firefox y Webkit, todas a través de la misma API. Esta herramienta ha obtenido notoriedad gracias a su compatibilidad con los principales motores de navegación, que consigue por incluir versiones adaptadas de Firefox y Webkit. Seguimos recibiendo informes de experiencias positivas, sobre todo en cuanto a su estabilidad. Además, algunos equipos han podido migrar fácilmente de Puppeteer, que presenta una API muy similar.

Prowler
Probar

Nos complace ver que la disponibilidad y madurez de herramientas para el análisis de la configuración de la infraestructura ha aumentado: Prowler ayuda a los equipos a escanear sus configuraciones de infraestructura de AWS y a mejorar la seguridad en función de los resultados. Aunque Prowler ha estado presente por un tiempo, ha evolucionado mucho en los últimos años, y encontramos muy valioso que permite a los equipos asumir responsabilidades para alcanzar niveles apropiados de seguridad con ciclos cortos de retroalimentación. Prowler clasifica a las verificaciones del AWS CIS benchmarking en diferentes grupos (Identity and Access Management, Logging, Monitoring, Networking, CIS Level 1, CIS Level 2, EKS-CIS) e incluye numerosas validaciones que permiten comprender mejor aspectos sobre el cumplimiento de las normativas PCI DSS y GDPR.

Pyright
Probar

Aunque duck typing es visto como una ventaja por muchas personas programadoras de Python, algunas veces, especialmente en grandes repositorios de código, la verificación de tipado también puede ser útil. Por esa razón, algunas anotaciones de tipos han sido incluidas en las Propuestas de Mejora de Python (Python Enhancement Proposals, PEP) y Pyright es un verificador para dichas anotaciones. Además, proporciona algun nivel de inferencia de tipos y protecciones que son capaces de entender código con estructuras condicionales. Diseñado pensando en grandes repositorios de código, Pyright es rápido y sus verificaciones en modo observador suceden incrementalmente a medida que los archivos cambian para disminuir aún más los ciclos de retroalimentación. Pyright puede usarse directamente en la línea de comandos y también existen integraciones disponibles para VS Code, Emacs, vim, Sublime y posiblemente otros editores. En nuestra experiencia, Pyright es mejor que alternativas como mypy.

Redash
Probar

Adoptar una filosofía de DevOps del estilo "tú lo construyes, tú lo ejecutas" significa que los equipos deben prestar más atención a las métricas técnicas como a las de negocio, las cuales pueden ser tomadas de los sistemas que despliegan. A menudo vemos que las herramientas para analíticas son difíciles de usar para la mayoría de las desarrolladoras, así que la tarea de capturar y presentar métricas queda para otros equipos, incluso tiempo después de que las funcionalidades han sido entregadas a los usuarios finales. Nuestros equipos encontraron que Redash es muy útil para consultar métricas del producto y para crear paneles y tableros, en formas que pueden ser autogestionadas por la mayoría de las desarrolladoras, acortando los ciclos de retroalimentación y enfocando al equipo en los resultados del negocio.

Terratest
Probar

Terratest nos llamó la atención en el pasado como una opción interesante para pruebas de infraestructura. Desde entonces, nuestros equipos lo han utilizado y están muy entusiasmados por su estabilidad y la experiencia que proporciona. Terratest es una biblioteca de Golang que hace más fácil escribir pruebas automáticas para el código de infraestructura. Utilizando herramientas

Herramientas

Tuple es una herramienta relativamente nueva optimizada para la programación en pareja en remoto, diseñada para llenar el vacío que dejó Slack en el mercado después de abandonar Screenhero..

(Tuple)

Herramientas

imgcook es un producto SaaS de Alibaba que permite transformar varios archivos de diseño (Sketch, PSD, imágenes estáticas) a código front-end de manera inteligente.

(imgcook)

de infraestructura como código, como Terraform, es posible crear componentes de infraestructura reales (como servidores, firewalls o balanceadores de carga) para desplegar aplicaciones en ellos y después validar el comportamiento esperado utilizando Terratest. Al finalizar las pruebas, Terratest puede retirar las aplicaciones y limpiar los recursos. Esto lo hace muy útil para pruebas de infraestructura de extremo a extremo en un entorno real.

Tuple
Probar

Tuple es una herramienta relativamente nueva optimizada para la programación en pareja en remoto, diseñada para llenar el vacío que dejó Slack en el mercado después de abandonar a Screenhero. Aunque todavía presenta algunos problemas por su crecimiento (por ejemplo, su disponibilidad está limitada a macOS por ahora, con el soporte para Linux próximamente, y tiene algunas anomalías por resolver en su interfaz de usuario), hemos tenido una buena experiencia usándola dentro de esas restricciones. A diferencia de las herramientas de videoconferencia de propósito más general con capacidades para compartir pantallas, como Zoom, Tuple permite el control dual mediante dos cursores de ratón y, a diferencia de opciones como Visual Studio Live Share, no está vinculado a un IDE. Tuple admite llamadas de voz y video, uso compartido de portapapeles y menor latencia que las herramientas comunes. La capacidad de Tuple para permitir dibujar y borrar con facilidad en la pantalla de la pareja hace sea una herramienta muy intuitiva y de fácil manejo para las personas desarrolladoras.

Why Did You Render
Probar

Al trabajar con React, a menudo nos encontramos con situaciones en las cuales una página se ralentiza debido a que ciertos componentes se están volviendo a renderizar cuando no deberían. Why Did You Render es una librería que ayuda a detectar por qué un componente se está volviendo a renderizar; lo consigue al modificar el comportamiento del código en tiempo de ejecución mediante monkey patching React. Nosotros lo hemos utilizado con gran efectividad en algunos de nuestros proyectos para investigar problemas de rendimiento.

Buildah y Podman
Evaluar

A pesar de que Docker se ha convertido en una elección razonable predeterminada para contenerización, vemos que hay nuevos actores en este espacio llamando la atención. Ese es el caso de Buildah y Podman, que son proyectos complementarios para construir imágenes (Buildah) y correr contenedores (Podman) sin privilegios de usuario root en múltiples distribuciones de Linux. Podman presenta un motor sin procesos demonios para manejar y correr los contenedores, lo que es una propuesta interesante en comparación a lo que hace Docker. El hecho de que Podman puede usar ya sea imágenes de tipo Open Container Initiative (OCI) construidas por Buildah o imágenes Docker, hacen de esta herramienta atractiva y fácil de usar.

GitHub Actions
Evaluar

Los servidores de integración contínua (CI) y las herramientas de compilación son algunos de los elementos más antiguos y más usados de nuestro kit. Cubren toda la gama, desde simples servicios alojados en la nube hasta complejos servidores de pipelines definidos por código soportados por flotas de agentes de compilación. Dada nuestra experiencia y la amplia gama de opciones disponibles, inicialmente estábamos escépticos cuando GitHub Actions fue presentado como otro mecanismo para administrar los procesos de compilación e integración. Pero la oportunidad para que las desarrolladoras comiencen con algo pequeño y personalicen fácilmente el comportamiento significa que GitHub Actions se está volviendo la opción predeterminada para los proyectos más pequeños. Es difícil discutir la conveniencia de tener la herramienta de compilación integrada directamente en el repositorio de código fuente. Ha surgido una comunidad entusiasta en torno a esta función y eso significa que existe una amplia gama de herramientas y flujos de trabajo aportados por los usuarios para comenzar. Los proveedores de herramientas también se están incorporando a través del GitHub Marketplace. Sin embargo, recomendamos proceder con cautela. Aunque el código y el historial de Git se pueden exportar a servicios alternos, no se puede hacer lo mismo con un flujo de trabajo de desarrollo basado en GitHub Actions. Además, es necesario usar el mejor criterio para determinar cuándo un proyecto es lo suficientemente grande o complejo como para justificar una herramienta de pipelines con soporte independiente. Sin embargo, para comenzar a trabajar rápidamente en proyectos más pequeños, vale la pena considerar GitHub Actions y el ecosistema que está creciendo a su alrededor.

Imagen nativa de Graal
Evaluar

La Imagen Nativa de Graal es una tecnología que compila código Java en un binario nativo de un sistema operativo, en la forma de un ejecutable enlazado estáticamente o de una biblioteca compartida. Una imagen nativa está optimizada para reducir la huella de memoria y el tiempo de inicio de una aplicación. Nuestros equipos han usado con éxito las imágenes nativas de Graal, ejecutándolas como pequeños contenedores Docker, en la arquitectura sin servidor (serverless architecture) donde reducir el tiempo de inicio es importante. Aunque se diseñó para utilizarse con lenguajes de programación como Go o Rust que se compilan nativamente y requieren de tamaños pequeños de binarios y de tiempos de inicio cortos, la imagen nativa de Graal puede ser igualmente útil para equipos que tienen otros requerimientos y que quieren usar lenguajes basados en la JVM.
El generador de imágenes nativas de Graal, native-image, soporta lenguajes basados en la JVM, como Java, Scala, Clojure y Kotlin, y genera ejecutables para múltiples sistemas operativos como macOS, Windows y varias distribuciones de Linux. Ya que se trabaja en un "mundo" que se asume "cerrado", donde todo el código se conoce en tiempo de compilación, se necesitan configuraciones adicionales para que características como la reflexión o la carga dinámica de clases funcionen, ya que no se pueden deducir los tipos en tiempo de compilación a partir del código mismo.

HashiCorp Boundary
Evaluar

HashiCorp Boundary combines the secure HashiCorp Boundary combina las capacidades de gestión de identidad y redes seguras necesarias para la intermediación del acceso a sus hosts y servicios en un solo lugar y en una combinación de nube y recursos locales si es necesario. La administración de claves se puede realizar integrando cualquier servicio de administración de claves, ya sea de un proveedor en la nube o algo como HashiCorp Vault. HashiCorp Boundary admite un número creciente de proveedores de identidad que pueden integrarse con partes de su entorno de servicios para ayudar a definir los permisos, no solo en el host sino también a nivel de servicio. Por ejemplo, permite controlar el acceso detallado a un clúster de Kubernetes y la obtención de catálogos de servicios de forma dinámica de diversas fuentes es algo que está en el radar.
Todo esto queda fuera del camino de los usuarios finales de ingeniería que obtienen la experiencia de línea de comandos a la que están acostumbrados, conectados de forma segura a través de la capa de administración de red del Boundary.

imgcook
Evaluar

¿Recuerdan el proyecto pix2code que mostraba cómo generar código automáticamente a partir de las capturas de pantallas de una interfaz gráfica? Ahora existe una versión en forma de producto de esta técnica llamada imgcook, un producto SaaS de Alibaba que permite transformar archivos de diseño (Sketch, PSD, imágenes estáticas) a código front-end de manera inteligente. Alibaba necesita personalizar muchas páginas de campañas durante su festival de compras llamado 11:11. Generalmente, son pantallas de un solo uso que deben ser construidas rápidamente.
A través de un método de aprendizaje profundo, el diseño de la experiencia de usuario es primero procesado a código front-end y luego es ajustado por alguna desarrolladora. Nuestro equipo está evaluando esta tecnología: aunque el procesamiento de imágenes se realiza en el lado del servidor y la interfaz principal está en la web, imgcook provee herramientas que se pueden integrar en el ciclo de diseño y desarrollo, además, puede generar código estático y también código de enlace a datos si se define un DSL. Esta tecnología no es perfecta aún, las diseñadoras deben sujetarse a ciertas especificaciones para mejorar la precisión de la generación del código (que de todas maneras deberá ser ajustado por las desarrolladoras después). Siempre hemos sido cautelosos con la generación "mágica" de código, porque generalmente suele ser un código difícil de mantener a largo plazo e imgcook no es la excepción. Pero si se limita su uso a un contexto específico, como páginas de campañas de un solo uso, vale la pena probarlo.

Longhorn
Evaluar

Longhorn es un sistema distribuido de almacenamiento en bloques para Kubernetes. Hay muchas opciones para el almacenamiento persistente para Kubernetes, pero a diferencia de la mayoría, Longhorn está construido desde cero para proveer instantáneas y copias de seguridad incrementales, reduciendo las molestias de poner en marcha un almacenamiento replicado para instancias de Kubernetes que no están alojadas en la nube. Con el reciente soporte experimental para ReadWriteMany (RWX) incluso se puede montar un mismo volumen para lectura y para escritura a través de distintos nodos. Escoger el sistema de almacenamiento correcto para Kubernetes no es tarea fácil y recomendamos evaluar Longhorn en función de tus necesidades.

Herramientas

Recommender es un servicio de Google Cloud que analiza los recursos que usas y ofrece recomendaciones de cómo optimizarlos en base al uso actual.

(Recommender)

Herramientas

Zally es un linter minimalista de OpenAPI que ayuda a garantizar que una API se ajuste a la guía de estilo utilizada por el equipo.

(Zally)

Operator Framework
Evaluar

Operator Framework es un conjunto de herramientas de código abierto que simplifica la construcción y la administración del ciclo de vida de Kubernetes operators. El patrón operador de Kubernetes, originalmente introducido por CoreOS, es un enfoque para encapsular el conocimiento de operar una aplicación utilizando las capacidades nativas de Kubernetes; incluye resources para gestionar y controller code que garantiza que los recursos coincidan con el estado esperado. Este enfoque ha sido usado para ampliar Kubernetes y administrar varias aplicaciones de forma nativa, particularmente, las que manejan estado . Operator Framework cuenta con tres componentes: Operator SDK, que simplifica la construcción, pruebas y empaquetamiento de operadores Kubernetes; Operator lifecycle manager para instalar, administrar y actualizar los operadores; y un catalog para publicar y compartir operadores de terceros. Nuestros equipos han descubierto que Operator SDK es particularmente poderoso para desarrollar rápidamente aplicaciones nativas de Kubernetes.

Recommender
Evaluar

El número de servicios ofrecidos por los proveedores grandes de la nube continúa creciendo, y también lo hace la conveniencia y la madurez de las herramientas que ayudan a usarlos de manera segura y eficiente. Recommender es un servicio de Google Cloud que analiza los recursos que usas y ofrece recomendaciones de cómo optimizarlos en base al uso actual. El servicio consiste de una serie de "recomendadores" en áreas como seguridad, uso de cómputo o ahorro de costos. Por ejemplo, el IAM Recommender ayuda a implementar mejor el principio de menor privilegio señalando los permisos que nunca se han usado y que son potencialmente muy amplios.

Remote - WSL
Evaluar

En los últimos años, el Windows Subsystem for Linux (WSL) ha sido mencionado algunas veces en nuestras discusiones. Aunque nos gustó lo que vimos, incluyendo las mejoras que introdujo WSL 2, no ha logrado entrar en el Radar. En esta edición queremos destacar una extensión para Visual Studio Code que ha mejorado en gran medida la experiencia de trabajo con WSL. Si bien los editores basados en Windows siempre podían acceder a los archivos del sistema de archivos de WSL, no sabían de la existencia de este entorno Linux aislado. Con la extensión Remote - WSL, Visual Studio Code se vuelve consciente de WSL, permitiendo a las personas desarrolladoras ejecutar un shell de Linux. Esta extensión también permite depurar binarios que se ejecutan dentro de WSL desde Windows. IntelliJ, de JetBrains, también ha trabajado en mejorar notablemente su soporte para WSL.

Spectral
Evaluar

Uno de los patrones que hemos visto repetirse en esta publicación es que las herramientas estáticas de comprobación de errores y estilo surgen rápidamente después de que un nuevo lenguaje gana popularidad. Estas herramientas se conocen genéricamente como linters, en honor a la clásica y querida utilidad lint de Unix, que analiza estáticamente el código C. Nos gustan estas herramientas porque detectan errores tempranamente, incluso antes de que se compile el código. La instancia más nueva de este patrón es Spectral, un linter para YAML y JSON. Aunque Spectral es una herramienta genérica para estos formatos, su principal objetivo es OpenAPI (la evolución de Swagger) y AsyncAPI. Spectral viene con un conjunto completo de reglas listas para usar para estas especificaciones que pueden ahorrarle dolores de cabeza a las personas desarrolladoras al diseñar e implementar APIs o colaboración basada en eventos. Estas reglas verifican las especificaciones adecuadas de los parámetros de la API o la existencia de una declaración de licencia en la especificación, entre otras cosas. Si bien esta herramienta es una adición bienvenida al flujo de trabajo de desarrollo de APIs, plantea la pregunta de si una especificación no ejecutable debe ser tan compleja como para requerir una técnica de verificación de errores diseñada para lenguajes de programación. ¿Quizás las personas desarrolladoras deberían escribir código en lugar de especificaciones?

Yelp detect-secrets
Evaluar

Yelp detect-secrets es un módulo de Python para detectar secretos dentro de una base de código fuente: analiza los archivos dentro de un directorio en búsqueda de secretos. Se puede utilizar con Git mediante un hook de pre-commit o como parte de un pipeline de CI/CD para realizar un escaneo en múltiples ubicaciones. Viene con una configuración predeterminada que simplifica su uso, y también puede ser modificada para adaptarse a diversas necesidades. También se pueden instalar complementos personalizados y añadirlos a sus búsquedas heurísticas predeterminadas. En comparación con ofertas similares, descubrimos que esta herramienta detecta más tipos de secretos con sus configuraciones predeterminadas.

Zally
Evaluar

A medida que madura el ecosistema para la especificación de APIs, estamos viendo que aparecen más herramientas para automatizar la validación de estilos. Zally es un linter minimalista para OpenAPI que ayuda a garantizar que un API se ajusta a la guía de estilo utilizada por el equipo. De manera predeterminada, esta herramienta validará usando las reglas desarrolladas para la guía de estilos de Zalando, pero también soporta un mecanismo de extensión basado en Kotlin para la creación de reglas personalizadas. Zally incluye una interfaz de usuario web muy intuitiva para comprender las violaciones a las reglas de estilos e incluye una interfaz de línea de comando (CLI) que hace fácil su integración a los pipelines de CD.

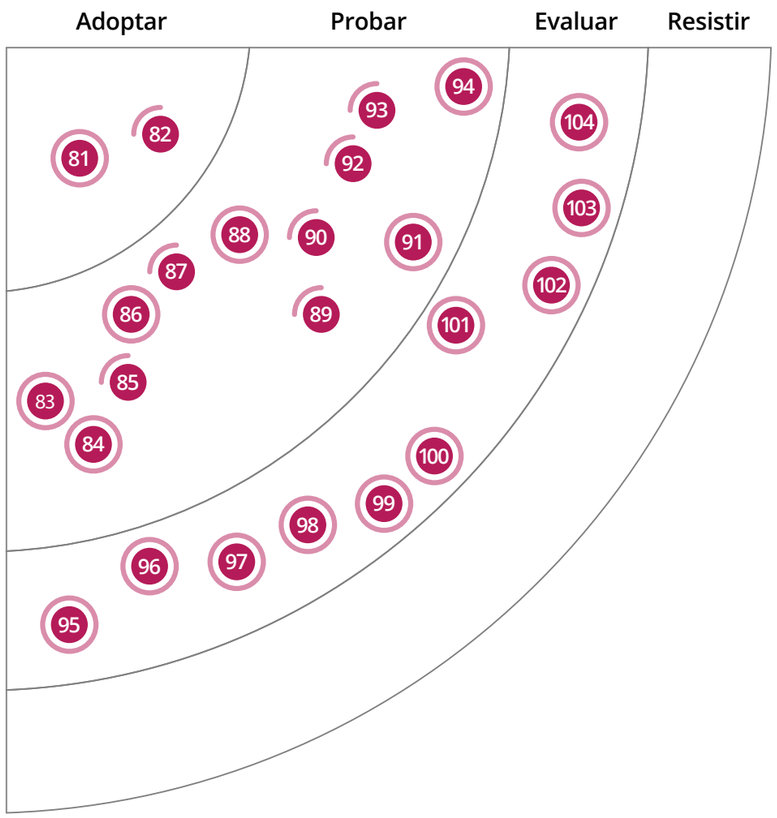
AWS CodePipeline
Resisitir

En base a la experiencia de múltiples equipos de Thoughtworks, sugerimos tener cuidado al considerar el uso de AWS CodePipeline. Específicamente, hemos notado que una vez que los equipos evolucionan y necesitan pipelines algo más complejos, esta herramienta puede empezar a ser difícil de usar. A pesar de que su uso podría parecer como una victoria rápida al comenzar con AWS, les aconsejamos dar un paso atrás y revisar con detenimiento si AWS CodePipeline ayudará a solventar algunas necesidades que podrían aparecer a largo plazo, como por ejemplo, flujos con tareas paralelas (pipeline fan-out & fan-in) o escenarios más complejos de despliegues y de ejecución de pruebas, que hacen uso de dependencias y activadores (triggers) no triviales.

TECHNOLOGY RADAR

Lenguajes & Frameworks

Lenguajes & Frameworks



Combine
Adoptar

Hace bastante tiempo posicionamos a ReactiveX, una familia de marcos de trabajo de código abierto para la programación reactiva, dentro del anillo "Adoptar" del Radar. En 2017 mencionamos la adición de RxSwift, que trajo la programación reactiva al desarrollo para iOS utilizando Swift. Desde entonces, Apple ha introducido su propia solución de programación reactiva en la forma del marco de trabajo Combine. Este se ha convertido en nuestra elección por defecto para aplicaciones que soportan iOS 13 como entorno de despliegue. Es más fácil de aprender que RxSwift y se integra muy bien con SwiftUI. Si se está planificando convertir una aplicación existente con RxSwift a Combine o hacerlos coexistir, se debería evaluar a RxCombine.

LeakCanary
Adoptar

Nuestros equipos de desarrollo móvil ven que LeakCanary es una buena elección predeterminada para el desarrollo en Android. Esta utilidad detecta las molestas fugas de memoria en Android, es extremadamente simple de integrar y proporciona notificaciones con trazas claras hacia las causas de las fugas. LeakCanary puede ahorrar horas de trabajo tedioso para solucionar problemas de "memoria insuficiente" en múltiples dispositivos. Es por esto que recomendamos añadirla en el juego de herramientas.

Angular Testing Library
Probar

Como estamos permanentemente desarrollando aplicaciones web en JavaScript, seguimos apreciando el enfoque de Testing Library para probar aplicaciones; continuamos explorando y ganando experiencia con sus paquetes, más allá de React Testing Library. Angular Testing Library trae todos los beneficios de su familia cuando se prueban componentes de interfaz de usuario de una manera centrada en el usuario, en favor de pruebas más fáciles de mantener enfocadas principalmente en el comportamiento, en vez de probar los detalles de implementación de la interfaz de usuario. Aunque su documentación es limitada, Angular Testing Library proporciona buenas pruebas de muestra que nos ayudaron a comenzar más rápido en varias ocasiones. Hemos tenido un gran éxito con esta biblioteca en nuestros proyectos Angular y aconsejamos experimentar con este sólido enfoque de pruebas.

Adoptar
81. Combine
82. LeakCanary

Probar
83. Angular Testing Library
84. AWS Data Wrangler
85. Blazor
86. FastAPI
87. io-ts
88. Kotlin Flow
89. LitElement
90. Next.js
91. Módulos bajo demanda
92. Streamlit
93. SWR
94. TrustKit

Evaluar
95. .NET 5
96. bUnit
97. Dagster
98. Flutter for Web
99. Jotai and Zustand
100. Kotlin Multiplatform Mobile
101. LVGL
102. React Hook Form
103. River
104. Federación de Módulos de Webpack 5

Resistir

Lenguajes & Frameworks

FastAPI un framework web moderno, rápido (alto rendimiento) para la construcción de APIs con Python 3.6 o mayor.

(FastAPI)

AWS Data Wrangler
Probar

AWS Data Wrangler es una biblioteca de código abierto que amplía las capacidades de Pandas a AWS al conectar marcos de datos a los servicios de datos de AWS. Además de Pandas, esta biblioteca aprovecha las capacidades de Apache Arrow y Boto3 para exponer varias APIs para cargar, transformar y guardar datos provenientes de lagos y almacenes de datos. Una limitación importante de esta biblioteca es que no permite realizar pipelines distribuidos para grandes volúmenes de datos. Sin embargo, es capaz de aprovechar servicios de datos nativos, como Athena, Redshift y Timestream, para hacer el trabajo pesado y extraer datos y así expresar transformaciones complejas que se adapten bien a los marcos de datos. Hemos utilizado AWS Data Wrangler en producción y como tal, permite concentrarse en escribir transformaciones sin perder demasiado tiempo en la conectividad a los servicios de datos de AWS.

Blazor
Probar

A pesar de que JavaScript y su ecosistema dominan el espacio de desarrollo de interfaces de usuario para la web, están surgiendo nuevas oportunidades tras la aparición de WebAssembly. Blazor continúa llamando nuestra atención. Está produciendo buenos resultados para aquellos equipos que construyen interfaces de usuario sofisticadas en C# con WebAssembly. El hecho de que nuestros equipos también puedan utilizar C# en el front-end les permite compartir el código y reutilizar bibliotecas existentes. Eso, junto a las herramientas actuales para depuración y la ejecución de pruebas, como bUnit, hacen que esta tecnología de código abierto merezca ser probada.

FastAPI
Probar

Estamos viendo que más equipos adoptan Python como el lenguaje de preferencia para el desarrollo de soluciones, no sólo para ciencia de datos, sino de servicios de back-end también. En estos casos, estamos teniendo buenas experiencias con FastAPI, un marco de trabajo web moderno y rápido (de alto rendimiento) para la construcción de APIs con Python 3.6 o superior. Además, este marco de trabajo y su ecosistema incluyen funcionalidades como la documentación del API a través de OpenAPI, que permite que los equipos puedan centrar su atención en las funcionalidades de negocio y crear APIs REST rápidamente. Esto convierte a FastAPI en una buena alternativa frente a las soluciones ya existentes.

io-ts
Probar

Hemos disfrutado mucho de usar TypeScript desde hace ya un buen tiempo y nos encanta la seguridad que el tipado fuerte proporciona. Sin embargo, introducir datos dentro de los límites del sistema de tipos (por ejemplo, a partir de la respuesta de una llamada a un servicio de back-end) puede provocar errores en tiempo de ejecución. Una biblioteca que ayuda a resolver este problemas es io-ts: cierra la brecha entre la revisión de tipos en tiempo de compilación y el consumo de datos externos en tiempo de ejecución, proporcionando funciones de codificación y decodificación. También se puede usar como un verificador personalizado del tipado. A medida que adquirimos más experiencia con io-ts en nuestro trabajo, nuestras impresiones iniciales positivas se confirman y nos sigue gustando la elegancia de su enfoque.

Kotlin Flow
Probar

La introducción de las co-rutinas de Kotlin abrió la puerta a varias innovaciones; Kotlin Flow es una de ellas, directamente integrada en la biblioteca de co-rutinas. Es una implementación de Reactive Streams basada en co-rutinas. A diferencia de RxJava, los flujos son un API nativa de Kotlin, similar a la familiar API de secuencias con métodos que incluyen map y filter. De manera similar a las secuencias, los flujos son “fríos”, lo que significa que los valores de las secuencias se construyen únicamente cuando se necesitan. Todo esto hace que el código con múltiples hilos de ejecución sea más fácil de escribir y entender que con otros enfoques. El método toList convierte un flujo en una lista de forma predecible, lo cual es un patrón recurrente en las pruebas.

LitElement
Probar

Se ha producido un progreso constante desde que escribimos por primera vez sobre Web Components en 2014. LitElement, parte del Proyecto Polymer, es una biblioteca simple que se puede usar para crear componentes web ligeros. Realmente es solo una clase base que elimina la necesidad de un montón de código repetitivo, haciendo que escribir componentes web sea mucho más fácil. Hemos tenido éxito con el uso de LitElement en proyectos y la estamos usando con más frecuencia en nuestros proyectos basados en Web Components, ya que vemos que la tecnología sigue madurando y la biblioteca está siendo bien apreciada.

Next.js
Probar

Pudimos obtener un poco más de experiencia usando Next.js para bases de código React desde la última vez que escribimos sobre él. Next.js es un marco de trabajo de configuración cero que incluye enrutamiento simplificado, compilación automática y empaquetado con Webpack y Babel, recarga en caliente rápida para conseguir un flujo de trabajo de desarrollo conveniente, entre otras características. Proporciona renderizado del lado del servidor de forma predeterminada, mejora las optimizaciones para motores de búsqueda y los tiempos de carga inicial y soporta la generación estática incremental. Hemos tenido informes de experiencias positivas de los equipos que utilizan Next.js y, dada su gran comunidad, seguimos entusiasmados con la evolución de este marco de trabajo.

Módulos bajo demanda
Probar

Los módulos bajo demanda para Android son un marco de trabajo que permite, en apps debidamente estructuradas, descargar e instalar APKs personalizados que contienen solo las funcionalidades requeridas. Podría ser buena idea evaluar esta estrategia para apps muy grandes, donde la velocidad de descarga puede ser un problema, o si es probable que el usuario solo requiera usar partes específicas de la funcionalidad en la instalación inicial. También puede ayudar a simplificar el soporte de múltiples dispositivos sin necesidad de diferentes APKs. Un marco de trabajo similar también existe para iOS.

Streamlit
Probar

Streamlit es un marco de trabajo de código abierto para Python usado por científicas de datos para construir aplicaciones interactivas de datos. La personalización de los modelos de aprendizaje automático requiere de tiempo; en vez de tener que lidiar con la aplicación principal (la que usa los modelos), hemos visto que es valioso crear rápidamente prototipos autónomos en Streamlit y obtener retroalimentación durante los ciclos de experimentación. Streamlit se destaca de competidores como Dash por su foco en el prototipado rápido y por el soporte a una amplia variedad de bibliotecas de visualización, como Plotly y Bokeh. Lo estamos usando en algunos proyectos y nos gusta cómo podemos crear visualizaciones interactivas con muy poco esfuerzo.

SWR
Probar

Nuestros equipos han descubierto que la biblioteca SWR para React Hooks permite obtener un código más limpio y con mejor desempeño cuando se la utiliza en circunstancias apropiadas. SWR implementa la estrategia para el almacenamiento de datos en caché de HTTP denominada *stale-while-revalidate*, primero retornando datos desde la caché (obsoletos), luego enviando la petición para obtener nueva información (revalidar) y finalmente refrescando los valores a partir de la respuesta recibida. Advertimos a los equipos a solo usar la estrategia de almacenamiento en caché SWR cuando se espera que una aplicación devuelva datos obsoletos. Además, nótese que HTTP requiere que los cachés respondan a una petición con la respuesta más actualizada, y que solamente en circunstancias cuidadosamente consideradas se permite devolver respuestas obsoletas.

TrustKit
Probar

La asignación de claves SSH públicas es un tema delicado. Si para una aplicación se selecciona una política incorrecta o si no se dispone de una asignación de reserva, esta podría dejar de funcionar de forma inesperada. Estos son los escenarios donde TrustKit es útil. Este es un marco de trabajo de código abierto que facilita la asignación de claves SSH públicas para aplicaciones iOS. También existe un marco de trabajo equivalente para Android. Elegir la estrategia de asignación idónea es cuestión de matices, y en esta guía de primeros pasos es posible encontrar más información al respecto. En cuanto a nuestra experiencia, podemos decir que hemos utilizado TrustKit en varios proyectos de producción con éxito.

.NET 5
Evaluar

No acostumbramos a presentar cada nueva versión de .NET en el Radar, pero .NET 5 representa un paso significativo para unir .NET Core y .NET Framework en una sola plataforma. Las organizaciones deben empezar a desarrollar una estrategia de migración de sus ambientes de desarrollo (una combinación fragmentada de marcos de trabajo dependiendo del objetivo de despliegue) a una sola versión de .NET 5 (o 6, cuando esté disponible). La ventaja de este enfoque será la de tener una plataforma de desarrollo común sin importar el ambiente deseado: Windows, Linux, dispositivos móviles multiplataforma (vía Xamarin) o navegadores (utilizando Blazor). Si bien el desarrollo políglota se mantendrá como el enfoque preferido en compañías que cuenten con una cultura de ingeniería que pueda soportarlo, otras encontrarán más eficiente estandarizarse

Lenguajes & Frameworks

Streamlit es un marco de trabajo de código abierto para Python usado por científicos de datos para construir aplicaciones interactivas de datos.

(Streamlit)

Lenguajes & Frameworks

Estas bibliotecas de manejo de estado para React apuntan a ser pequeñas y fáciles de usar. Tal vez, casi por coincidencia, ambos nombres son traducciones de la palabra estado a Japonés y Alemán respectivamente.

(Jotai y Zustand)

en una sola plataforma para el desarrollo en .NET. Por ahora, queremos mantenerlo en el anillo “Evaluar” para ver qué tan bien el marco de trabajo unificado final se desempeña en .NET 6.

bUnit
Evaluar

bUnit es una biblioteca para Blazor que facilita la creación de pruebas para componentes Blazor en los marcos de trabajo existentes de pruebas unitarias como NUnit, xUnit o MSTest. Proporciona una fachada alrededor del componente que permite ejecutarlo y probarlo siguiendo el paradigma familiar de pruebas unitarias: esto permite obtener retroalimentación muy rápidamente y además probar los componentes de forma aislada. Si desarrollas utilizando Blazor, recomendamos agregar bUnit a tu juego de herramientas.

Dagster
Evaluar

Dagster es un marco de trabajo de código abierto para la orquestación de datos para aprendizaje automático, analíticas y pipelines de datos simples de tipo ETL. A diferencia de otros marcos de trabajo basados en tareas, Dagster tiene en cuenta a los datos que fluyen por el pipeline y puede proporcionar seguridad de tipos. Con la vista unificada de pipelines y activos producidos, Dagster puede programar y orquestar a Pandas, Spark, SQL, o cualquier otra cosa que Python pueda invocar. Dagster es relativamente nuevo por lo que recomendamos evaluar sus capacidades para tus pipelines de datos.

Flutter for Web
Evaluar

Hasta ahora, Flutter ha brindado principalmente soporte para construir aplicaciones nativas en iOS y Android. Sin embargo, la visión del equipo de Flutter es soportar la construcción de aplicaciones en todas las plataformas. Flutter para la Web es un paso en esa dirección: permite construir aplicaciones para iOS, Android y navegadores utilizando la misma base de código. Ha estado disponible por alrededor de un año en el canal “Beta” y recientemente ha alcanzado su forma estable con el lanzamiento de Flutter 2.0. En esta versión del soporte para la web, el equipo de Flutter se ha enfocado en las aplicaciones web progresivas, aplicaciones de página única (single page apps) y en expandir las aplicaciones móviles ya existentes hacia la web. La aplicación y el código del marco de trabajo (que se escriben ambos en Dart) son compilados a JavaScript, en vez de a código de maquina ARM, que es utilizado en las aplicaciones móviles. El motor web de Flutter permite elegir entre dos renderizadores: HTML (que utiliza HTML, CSS, Canvas y SVG) y CanvasKit (que usa WebAssembly y WebGL para renderizar comandos “paint” de Skia al canvas del navegador). Algunos de nuestros equipos han empezado a usar Flutter para la Web y gustan de los resultados iniciales.

Jotai y Zustand
Evaluar

En la edición anterior del Radar, comentamos sobre el inicio de una etapa de experimentación con el manejo de estado en aplicaciones React. Movimos a Redux de vuelta al anillo “Probar”, indicamos que ya no es nuestra elección por defecto, y mencionamos a Recoil, de Facebook. Ahora queremos destacar a Jotai y Zustand, otras bibliotecas para el manejo de estado para React. Su objetivo es ser pequeñas y fáciles de usar; y (tal vez por casualidad) sus nombres son las traducciones de la palabra “estado” en japonés y alemán respectivamente. Mas allá de estas similitudes, hay diferencias en su diseño: el de Jotai se acerca más a Recoil, por cuanto el estado consiste de átomos que se almacenan dentro del árbol de componentes de React, mientras que Zustand almacena el estado fuera de React en un solo objeto de estado, muy parecido a como lo hace Redux. Los autores de Jotai proveen una lista muy útil para decidir cuándo usar cuál

Kotlin Multiplatform Mobile
Evaluar

Siguiendo la tendencia del desarrollo multiplataforma para móviles, Kotlin Multiplatform Mobile (KMM) es una nueva entrada en este espacio. KMM es un SDK proporcionado por JetBrains que potencia las capacidades multiplataforma de Kotlin y que incluye herramientas y características diseñadas para hacer que la experiencia completa de construcción de aplicaciones multiplataforma para dispositivos móviles sea más agradable y eficiente. Con KMM escribes por una sola vez el código de la lógica del negocio y de la aplicación base y puedes compartirlo con aplicaciones Android e iOS. La escritura de código específico para una plataforma queda solo para cuando es estrictamente necesario (por ejemplo, para aprovechar elementos nativos de la interfaz de usuario) y el código específico se mantiene en vistas diferentes para cada plataforma. Aunque está aún en estado Alpha, Kotlin Multiplatform Mobile está evolucionando rápidamente. Por supuesto, no lo perderemos de vista y esperamos que tu tampoco lo hagas.

LVGL
Evaluar

Con la creciente popularidad de las casas inteligentes y dispositivos para llevar puesto (wearable devices), la demanda de interfaces de usuario intuitivas está aumentando. Sin embargo, para quien está involucrado en el desarrollo de dispositivos integrados, en lugar de Android o iOS, el desarrollo de interfaces de usuario puede requerir de mucho esfuerzo. Como una biblioteca de gráficos integrados de código abierto, LVGL se ha vuelto cada vez más popular. LVGL se ha adaptado a las plataformas integradas convencionales como NXP, STM32, PIC, Arduino y ESP32. Requiere de muy poca memoria para operar: 64 kB de memoria flash y 8 kB de RAM son suficientes y se puede ejecutar sin problemas en varios MCUs de tipo Cortex-M0 de bajo consumo. LVGL soporta medios de entrada como la pantalla táctil, ratón y botones, y contiene más de 30 controles, incluido TileView, adecuado para relojes inteligentes. La licencia MIT seleccionada no restringe su uso empresarial o comercial. Las opiniones de nuestros equipos respecto a esta herramienta han sido positivas y uno de nuestros proyectos que utiliza LVGL ya se encuentra en producción, específicamente, en la manufactura de lotes pequeños.

React Hook Form
Evaluar

La creación de formularios para la web continúa siendo uno de los desafíos perpetuos del desarrollo de front-ends, en particular con React. Varios de nuestros equipos han utilizado Formik para facilitar la creación de formularios, sin embargo, algunos están ahora evaluando React Hook Form como una potencial alternativa. React Hooks ya existía cuando React Hook Form fue creado, por lo que se lo puede utilizar como un componente de primera clase: el marco de trabajo registra y monitorea los elementos del formulario como componentes no controlados mediante un hook, reduciendo significativamente la necesidad de volver a renderizar. También es bastante ligero en tamaño y en la cantidad de código de relleno requerida para utilizarlo.

River
Evaluar

La creación de modelos a partir de un conjunto de datos de entrenamiento se encuentra en el corazón de muchos métodos de aprendizaje automático. Una vez que el modelo ha sido creado, se puede utilizar muchas veces. Sin embargo, el mundo no es estacionario y con frecuencia los modelos necesitan ser modificados a medida que se hacen disponibles nuevos datos. Solamente volver a ejecutar el paso de creación del modelo puede ser lento y costoso. El aprendizaje incremental soluciona este problema, haciendo posible aprender de flujos de datos incrementalmente para reaccionar más rápido ante los cambios. Como punto adicional, los requerimientos tanto de cómputo como de memoria son menores y predecibles. En nuestras implementaciones hemos tenido buenas experiencias con el marco de trabajo denominado River, aunque hemos añadido verificaciones, a veces manuales, luego de actualizar el modelo.

Federación de Módulos de Webpack 5
Evaluar

El lanzamiento de la funcionalidad para la federación de módulos de Webpack 5 ha sido muy esperado por las desarrolladoras de arquitecturas micro-frontend. Esta característica introduce una forma más estandarizada de optimizar la carga y la administración de las dependencias de los módulos y el código compartido. La federación de módulos permite especificar aquellos que son compartidos, lo que ayuda a eliminar dependencias duplicadas entre múltiples micro-frontends al cargar una sola vez el código utilizado por varios módulos. También permite distinguir entre módulos locales y remotos: los remotos se cargan de forma asíncrona y no son parte del resultado compilado en sí. A diferencia de las dependencias de tiempo de compilación (como los paquetes npm) esto puede simplificar significativamente el despliegue de una actualización de un módulo con varias dependencias. Hay que tener en cuenta que esto requiere empacar todos los micro-frontends con Webpack, en vez de aplicar técnicas como los import maps, que podrían llegar a convertirse en parte del estándar W3C.

Lenguajes & Frameworks

Los modelos de aprendizaje necesitan ser modificados a medida que se hacen disponibles nuevos datos. El aprendizaje incremental hace posible aprender de flujos de datos incrementalmente para reaccionar más rápido ante los cambios. En nuestras implementaciones hemos tenido buenas experiencias con el marco de trabajo denominado River, una librería basada en Python para aprendizaje automático en línea.

(River)

/thoughtworks

Somos una consultora de software global y una comunidad de individuos apasionados guiados por propósitos, 9,000+ personas en 48 oficinas en 17 países. A lo largo de nuestros más de 27+ años de historia, hemos ayudado a nuestros clientes a resolver problemas comerciales complejos donde la tecnología es el diferenciador. Cuando la única constante es el cambio, te preparamos para lo impredecible.

¿Quieres estar al tanto con toda las noticias e insights relacionados al Radar?

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

suscríbete ahora

Coordinadoras de traducción: Magdalena Grondona, María José Lalama Paredes y Elizabeth Parra

Traductores: Abel Guillen, Aldemaro Díaz, Alejandro Batanero, Alexander López Lapo, Alexandra Granda, Andrés Negrete, Araceli Correa, Bárbara Deluchi, Cecilia Barudi, Cintya Aguirre, Daniel Santibañez, Diego Llerena, Eduard Maura i Puig, Eduardo Winpenny Tejedor, Fernanda Pérez, Francisco Trujillo, Giovanny Sayas, Gonzalo Rodríguez, Inigo Crespo, Ivonne Burgos, Javier Paez, Jesús Cardenal Escribano, Johanna Cabrera, Jorge Agudo, Juan Pablo Blanco, Kelly Landázuri, Lorena Campos, Magda Sbant, Manu Escudero, Marcelo Cartgena, Marco Montilla, María Cordova, Marianela Ortiz, Milber Champutiz, Nicol Rafalowski, Paola Cajilema, Paolo Arrata, Paula Marín, Pedro Grijalva, Raymi Salcedo, René Pérez, Reynier Pupo, Ricardo Rodríguez, Rosa Palli, Rubén Trujillo, Sol Alonso y Zully Arellano.

Editores técnicos: Carlos Oquendo y Fausto De La Torre

/thoughtworks

thoughtworks.com/radar

#TWTechRadar