/thoughtworks

Strategy. Design. Engineering.

Volume 29 | September 2023

Radar Tecnológico

Una guía de opinión sobre el entorno tecnológico actual

Sobre el Radar

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

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

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

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

Un vistazo al Radar

El Radar se dedica a rastrear cosas interesantes, a las que nos referimos como blips. Organizamos los blips en el Radar utilizando dos elementos de categorización: cuadrantes y anillos. Los cuadrantes representan los diferentes tipos de blips. Los anillos indican nuestra recomendación para utilizar esa tecnología.

Un blip es una tecnología o técnica que desempeña un papel en el desarrollo de software. Los blips son cosas que están en movimiento", es decir, que su posición en el Radar cambia a menudo, lo que suele indicar nuestra creciente confianza en recomendarlos a medida que avanzan por los anillos.

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

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

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

Resistir: Proceder con precaución.

Nuevo **Deplazado adentro /afuera** **Ningún cambio**

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

Contribuyentes

El Technology Advisory Board (TAB) es un grupo de 22 personas tecnólogas senior de Thoughtworks. El TAB se reúne dos veces al año en persona y virtualmente cada dos semanas. Su función principal es ser un grupo de asesoramiento para el CTO de Thoughtworks, Rachel Laycock, y CTO Emerita, Rebecca Parsons.

El TAB actúa como un organismo amplio que puede examinar los temas que afectan a la tecnología y a las personas tecnólogas en Thoughtworks. Esta edición del Radar Tecnológico de Thoughtworks es en base a la reunión virtual que TAB realizó remotamente en agosto del 2023.


Rebecca Parsons (CTO Emerita)


Rachel Laycock (CTO)


Martin Fowler (Chief Scientist)


Bharani Subramaniam


Birgitta Böckeler


Brandon Byars


Camilla Falconi Crispim


Erik Dörnenburg


Fausto de la Torre


Hao Xu


Ian Cartwright


James Lewis


Marisa Hoenig


Maya Ormaza


Mike Mason

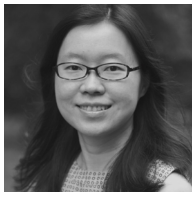

Neal Ford


Pawan Shah


Scott Shaw


Selvakumar Natesan


Shangqi Liu


Sofia Tania


Vanya Seth

Contribuyentes

Equipo de traducción:

David Corrales, Eduard Maura i Puig, Gustavo Chiriboga, María Montoza, Yago Pereiro, Norbs Rodriguez, Cristian Montero, Luis Maracara, Catalina Margozzini, Mayfe Yépez, Alex Ulloa, Sebastian Roman, Felipe Jorquera, Juan Romero Gómez, Natalia Rivera, Laura Eirás, Ari Handler, Andrea Obando, Angie Bracho, Raymi Salcedo, Erika Vacacela, Jenny Huang, Daniel Santibañez, Jorge Palacios, Irene Amo, Gisela Yumi, Antonella Castells, Jhosep Marin, Enrique Aracil, Catalina Solís, Ana Corral, Diana Pila, Jose Herdoiza, Elizabeth Cortez, Gabriel Villacis, Joaquin Hervas, Jesús Cardenal, Érika Dahi (Kika), Jose María Alonso Montero, Judit Navarro, Maria Valero, Lucia Parga, Cecilia Geraldo, Yeraldine Mendoza, Sendami Luque, Maria Jaramillo, Ernesto Fuentes, Carlos Carvajal, Cesar Abreu, Maura Yanez, Neysa Alarcón, Carolina Melgarejo, Tex Albuja, Isabel Hong, Milber Champutiz, Ana María Zúñiga, Pedro Carbonell, Paula Forero, Nahuel Delgado, Gabriela Marques, Paula Nieto, Eumir Ollarvez, Jorge Agudo Praena

Equipo de Edición:

Carlos Cavero, Maya Ormaza, Juan Infante Zumer, Joao Lucas Santana, Fernando Tamayo

Equipo de Marketing:

Elizabeth Parra, María José Lalama, Magdalena Grondona y Daniel Negrete.

Temas

Desarrollo de software asistido por IA

En absoluto sorprende que en la edición de este Radar predominen los tópicos relacionados con la IA. Por primera vez, necesitamos una guía visual para descifrar las diferentes categorías y capacidades (algo a lo que nunca tuvimos que recurrir ni en el apogeo del caos en el ecosistema JavaScript). Como consultora de software con un historial de prácticas de ingeniería pioneras como CI (integración continua) y CD (entrega continua), una de las categorías que nos interesa especialmente es el uso de la IA para ayudar al desarrollo de software. Como parte del Radar, debatimos muchas herramientas de asistencia en la codificación, como GitHub Copilot, Tabnine y Codeium. También estamos entusiasmados por cómo los LLMs de código abierto para la programación podrían sacudir el panorama de las herramientas, y vemos un gran potencial en la explosión de herramientas y capacidades de asistencia más allá de la codificación, como la asistencia en la redacción de historias de usuario, presentaciones cortas y otras tareas basadas en el lenguaje. Al mismo tiempo, esperamos que estas herramientas se utilicen con responsabilidad y se preste especial atención a riesgos de seguridad y calidad como por ejemplo las dependencias alucinantes.

¿Qué tan productivo es medir la productividad?

El desarrollo de software a veces puede parecer mágico para las personas no técnicas, lo que lleva a los líderes a esforzarse por medir qué tan productivos son los desarrolladores en sus misteriosas tareas. Nuestro referente, Martin Fowler, escribió sobre este tema ya en 2003, pero el tema no ha desaparecido. Hablamos de muchas herramientas y técnicas modernas para este radar que adoptan enfoques más matizados para medir el proceso creativo de construcción de software, pero que aún siguen siendo inadecuadas. Afortunadamente, la industria ha dejado de utilizarla cantidad de líneas de código como medida de resultados. Sin embargo, las formas alternativas de medir la A (“Actividad”) del framework SPACE como la cantidad de pull requests o problemas resueltos, siguen siendo indicadores deficientes de productividad. En cambio, la industria ha comenzado a centrarse en la eficacia de la ingeniería: en lugar de medir la productividad, deberíamos medir las cosas que sabemos que contribuyen o restan valor al flujo. En lugar de centrarnos en las actividades de un individuo, deberíamos centrarnos en las fuentes de desperdicio en el sistema y las condiciones que podemos empíricamente demostrar que tienen un impacto en la percepción de “productividad” del desarrollador. Nuevas herramientas como DX DevEx 360 abordan esto centrándose en la experiencia del desarrollador en lugar de en alguna medida engañosa del resultado. Sin embargo, muchos líderes continúan refiriéndose a la “productividad” de los desarrolladores de manera vaga y cualitativa. Sospechamos que al menos parte de este resurgimiento del interés tiene que ver con el impacto del desarrollo de software asistido por IA, lo que plantea la pregunta inevitable: ¿está teniendo un impacto positivo? Si bien las mediciones pueden estar adquiriendo algunos matices, las mediciones reales de la productividad aún son difíciles de alcanzar.

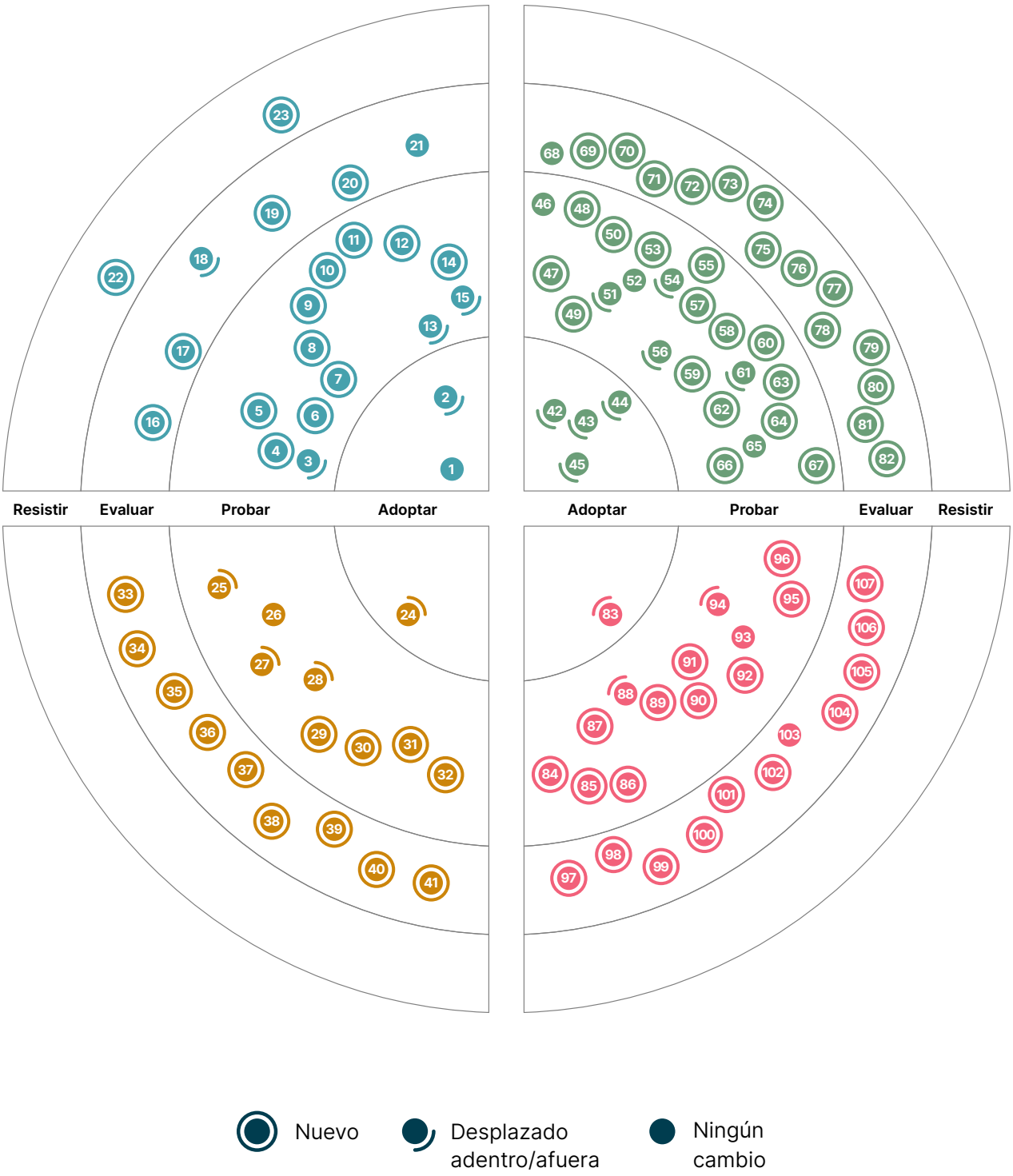
Numerosos LLMs

Los grandes modelos de lenguaje (LLMs en inglés) constituyen la base de muchos de los avances modernos en IA. Gran parte de la experimentación actual involucra interfaces de usuario parecidas a las de un chat como ChatGPT o Bard. Fundamentalmente, los principales ecosistemas competidores (ChatGPT de OpenAI, Bard de Google, LLaMA de Meta, Bedrock de Amazon, entre otros) estuvieron muy presentes en nuestras conversaciones. Más ampliamente, los LLMs son herramientas que pueden resolver una variedad de problemas, que van desde generación de contenido (texto, imágenes y vídeos) hasta la generación de código, resúmenes y traducción, por nombrar algunos. Teniendo el lenguaje natural como una poderosa capa de abstracción, estos modelos presentan un conjunto de herramientas universalmente atractivas y están siendo usadas por muchos trabajadores de la información. Nuestro discurso comprende varias facetas de los LLMs, incluyendo self-hosting, que permiten personalización y un mayor control en comparación a los LLMs que están hospedados en la nube. Con la creciente complejidad de los LLMs, deliberamos sobre la capacidad de cuantificarlos y ejecutarlos en pequeños espacios, especialmente en dispositivos nuevos y ambientes restringidos. Tocamos ReAct prompting, el cual promete un mejor rendimiento, así cómo agentes autónomos alimentados por LLMs que pueden ser usados para construir aplicaciones dinámicas que van más allá de las interacciones en forma de preguntas y respuestas. También mencionamos varias bases de datos de vectores (incluyendo Pinecone) que están teniendo un resurgimiento gracias a los LLMs. Las capacidades subyacentes de los LLMs, incluyendo las capacidades especializadas y auto hospedadas, continúan su crecimiento explosivo.

Madurez en soluciones alternativas de entrega remota

Aunque los equipos de desarrollo de software remotos han aprovechado la tecnología para superar las limitaciones geográficas durante años, el impacto de la pandemia impulsó la innovación en esta área, solidificando el trabajo totalmente remoto o híbrido como una tendencia duradera. Para este Radar, hemos analizado cómo las prácticas y herramientas de desarrollo de software remoto han madurado y los equipos siguen superando los límites con un enfoque en la colaboración efectiva en un entorno más distribuido y dinámico que nunca. Algunos equipos siguen ideando soluciones innovadoras utilizando nuevas herramientas colaborativas. Otros continúan adaptando y mejorando las prácticas presenciales existentes para actividades como programación en pareja en tiempo real o mob programming, talleres distribuidos (por ejemplo, remote Event Storming) y comunicaciones asíncronas y síncronas Aunque el trabajo remoto ofrece numerosos beneficios (incluyendo un grupo más diverso de talentos), es claro el valor de las interacciones cara a cara. Los equipos no deben dejar que se desvanezcan los ciclos de feedback críticos y deben ser conscientes de a lo que renuncian al realizar la transición a entornos remotos.

El Radar



El Radar

Técnicas

Adoptar

1. Sistemas de Diseño
2. Enfoque Ligero a los RFCs

Probar

3. Diseño de tests de componentes que tienen en cuenta la accesibilidad
4. Análisis de rutas de ataque
5. Incorporación automática de PRs con actualización de dependencias
6. Product Thinking de Datos FAIR
7. OIDC para GitHub Actions
8. Provisión de monitores y alertas con Terraform
9. ReAct prompting
10. Retrieval-Augmented Generation
11. Modelamiento de fallos basado en el riesgo
12. Lenguaje natural semi-estructurado para LLMs
13. Seguimiento de la salud sobre la deuda técnica
14. Tests unitarios para reglas de alerta
15. Seguridad de confianza cero para CI/CD

Evaluar

16. Health checks de dependencias para contrarrestar package hallucinations
17. Registro de decisiones del sistema de diseño
18. GitOps
19. Agentes autónomos impulsados por LLM
20. Orquestación de plataforma
21. LLMs autohospedados

Resistir

22. Ignorando las listas top 10 de OWASP
23. Componentes web para aplicaciones web renderizadas en servidor

Plataformas

Adoptar

24. Colima

Probar

25. CloudEvents
26. DataOps.live
27. Google Cloud Vertex AI
28. Immuta
29. Lokalise
30. Orca
31. Trino
32. Wiz

Evaluar

33. ActivityPub
34. Azure Container Apps
35. Servicio OpenAI de Azure
36. ChatGLM
37. Chroma
38. Kraftful
39. pgvector
40. Pinecone
41. wazero

Resistir

—

Herramientas

Adoptar

42. dbt
43. Mermaid
44. Ruff
45. Snyk

Probar

46. AWS Control Tower
47. Bloc
48. cdk-nag
49. Checkov
50. Chromatic
51. Cilium
52. Cloud Carbon Footprint
53. Pruebas de estructura de contenedor
54. Devbox
55. DX DevEx 360
56. GitHub Copilot
57. Insomnia
58. Plugin de cliente HTTP para IntelliJ
59. KEDA
60. Kubeconform
61. mob
62. MobSF
63. Mocks Server
64. Defensa en la ejecución de Prisma
65. Terratest
66. Thanos
67. Yalc

Evaluar

68. ChatGPT
69. Codeium
70. GitHub merge queue
71. Google Bard
72. Google Cloud Workstations
73. Gradio
74. KWOK
75. Llama 2
76. Maestro
77. LLM de código abierto para codificación
78. OpenCost
79. OpenRewrite
80. OrbStack
81. Pixie
82. Tabnine

Resistir

—

Lenguajes y Frameworks

Adoptar

83. Playwright

Probar

84. API Minimalista .NET
85. Ajv
86. Armeria
87. AWS SAM
88. Dart
89. fast-check
90. Kotlin con Spring
91. Mockery
92. Netflix DGS
93. OpenTelemetry
94. Polars
95. Pushpin
96. Snowpark

Evaluar

97. Perfiles de Referencia
98. GGML
99. GPTCache
100. API Inflexión Gramatical
101. htmx
102. Kotlin Kover
103. LangChain
104. LlamaIndex
105. promptfoo
106. Semantic Kernel
107. Spring Modulith

Resistir

—

Técnicas

Adoptar

1. Sistemas de Diseño
2. Enfoque Ligero a los RFCs

Probar

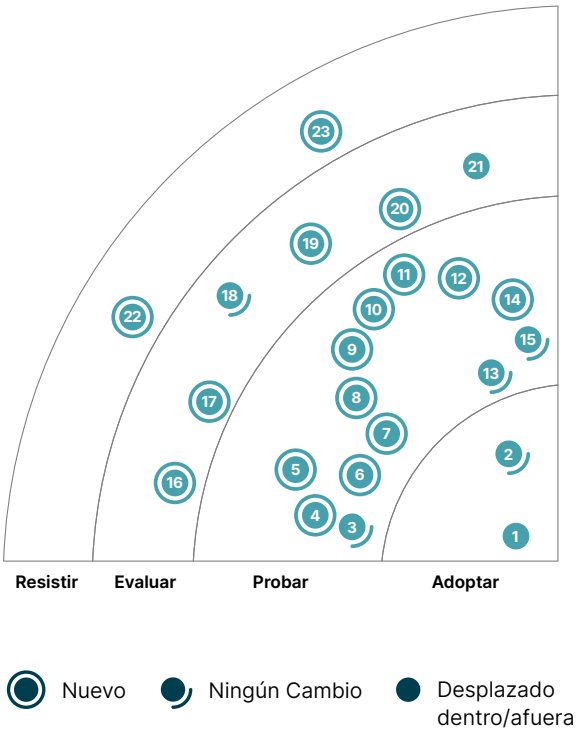
3. Diseño de tests de componentes que tienen en cuenta la accesibilidad
4. Análisis de rutas de ataque
5. Incorporación automática de PRs con actualización de dependencias
6. Product Thinking de Datos FAIR
7. OIDC para GitHub Actions
8. Provisión de monitores y alertas con Terraform
9. ReAct prompting
10. Retrieval-Augmented Generation
11. Modelamiento de fallos basado en el riesgo
12. Lenguaje natural semi-estructurado para LLMs
13. Seguimiento de la salud sobre la deuda técnica
14. Tests unitarios para reglas de alerta
15. Seguridad de confianza cero para CI/CD

Evaluar

16. Health checks de dependencias para contrarrestar package hallucinations
17. Registro de decisiones del sistema de diseño
18. GitOps
19. Agentes autónomos impulsados por LLM
20. Orquestación de plataforma
21. LLMs autohospedados

Resistir

22. Ignorando las listas top 10 de OWASP
23. Componentes web para aplicaciones web renderizadas en servidor



1. Sistemas de Diseño

Adoptar

A medida que el desarrollo de aplicaciones se vuelve cada vez más complejo y dinámico, es un desafío entregar productos usables y accesibles con un estilo consistente. Esto es particularmente más tangible en organizaciones más grandes con múltiples equipos trabajando en diferentes productos. Los sistemas de diseño definen una colección de patrones de diseños, librerías de componentes, buenas prácticas de diseño e ingeniería que aseguran productos digitales consistentes. Evolucionados a partir de los estilos de guías corporativas del pasado, los sistemas de diseños ofrecen librerías compartidas y documentos que son fáciles de encontrar y usar. Generalmente, la guía es escrita como código y se mantiene bajo control de versiones para que la guía sea menos ambigua y más fácil de mantener que simples documentos. Los sistemas de diseño se han convertido en un enfoque estándar cuando se trabaja con equipos y disciplinas en el desarrollo de productos porque permite que los equipos se concentren. Pueden abordar desafíos estratégicos relacionados al producto en sí sin tener que reinventar la rueda cada vez que necesita un nuevo componente visual.

Nuestras experiencias demuestran que los equipos rara vez aplican esta mentalidad centrada en el producto cuando desarrollan sistemas de diseño. Los principales consumidores de estas librerías y documentos compartidos son los equipos de desarrollo de productos. Cuando se aplica la mentalidad de producto, los dueños del sistema deben establecer empatía con los consumidores internos (los equipos de desarrollo) y colaborar con ellos. Encontramos que la razón por la que muchas librerías de componentes son difamadas es porque el equipo propietario no pudo brindarles a los consumidores lo que ellos necesitaban lo suficientemente rápido y no estaba preparado para aceptar contribuciones externas. Una mentalidad centrada en el producto también requiere que las organizaciones piensen en si y como las contribuciones deben ser realizadas al diseño de sistemas y cómo deben regirse estas contribuciones — sobre este tema, recomendamos aplicar la técnica de registros de decision de los sistemas de diseño Para nosotros, ejecutar un buen sistema de diseño o una librería de componentes requiere tanto trabajo social como trabajo técnico.

2. Enfoque Ligero a los RFCs

Adoptar

Un RFC (del inglés Request for Comments) es un documento formal que incluye diseños atados a contexto e ideas de arquitectura que facilitan la colaboración de un equipo y la toma de decisiones. Casi todas las organizaciones nativamente digitales y en expansión, utilizan RFCs para tomar decisiones sobre diseño, arquitectura, técnicas y las maneras en las que colaboran sus equipos. Organizaciones maduras han utilizado RFCs en equipos autónomos para mejorar la comunicación y colaboración, especialmente en la toma de decisiones entre equipos. Son usualmente utilizados como un proceso para examinar y ratificar los registros de decisiones de arquitectura. El resultado es un proceso transparente y colaborativo que permite dar a las personas impactadas por una decisión, la oportunidad de involucrarse y colaborar antes de que esta decisión sea ratificada. En entornos altamente dinámicos, es muy común que el razonamiento que lleva a tomar decisiones de diseño se pierde y los equipos que son responsables por implementar dicha decisión se queden con muchas interrogantes. Un RFC proporciona un registro de auditoría de decisiones que beneficia a los futuros miembros del equipo y recopila la evolución técnica y empresarial de una organización. Un RFC puede ser una herramienta valiosa para facilitar la arquitectura evolutiva. No obstante, para un mejor resultado, recomendamos tomar un enfoque ligero a los RFCs. Si no son enfocados y al grano, estos documentos tienden a crecer en extensión con el tiempo y comienzan a parecerse a documentos tradicionales de soluciones arquitecturales que son archivados y olvidados.

3. Diseño de tests de componentes que tienen en cuenta la accesibilidad

Probar

Los requisitos de accesibilidad se deberían tener en cuenta durante las pruebas de componentes web. Si bien plugins de frameworks de testing como chai-a11y-axe permiten comprobar una API para revisar que funcionan, el diseño de pruebas de componentes que tienen en cuenta la accesibilidad puede ayudar aún más ofreciendo todos los elementos semánticos que los lectores de pantalla y otras tecnologías de accesibilidad requieren. Primero, en lugar de usar test-ids o clases para encontrar y seleccionar elementos que se quieren validar, se usa el principio de identificar elementos usando roles ARIA u otros atributos semánticos que usan las tecnologías de accesibilidad. Algunas librerías de pruebas como Testing Library, incluso lo recomiendan en su documentación. Segundo, no hay que limitarse solo a probar interacciones de click; sino también considerar a las personas que no pueden usar el ratón o ver la pantalla, así como añadir pruebas adicionales para el teclado y otras interacciones. La técnica descrita está muy establecida entre nuestros equipos y deberíamos de haberla puesto en el área de Probar hace tiempo.

4. Análisis de rutas de ataque

Probar

El análisis de rutas de ataque es una técnica de análisis de seguridad que identifica y evalúa las potenciales rutas que un atacante podría tomar para aprovechar las vulnerabilidades en los sistemas y redes de una organización. Anteriormente, la mayoría de las estrategias de análisis de seguridad o herramientas relacionadas se centraban en áreas de riesgo concretas, como configuraciones erróneas, contenedores con vulnerabilidades o alertas CVE. Este enfoque aislado conlleva a que los equipos no puedan ver cómo se pueden combinar los riesgos con las debilidades existentes en otras capas del stack de tecnología para crear rutas de ataque peligrosas. Aunque esta técnica no es novedosa, los avances recientes en las herramientas de análisis de seguridad la han hecho más accesible para los equipos de seguridad. Orca y Wiz son dos de esas herramientas. Sugerimos que los equipos que administran infraestructuras complejas consideren el uso de esta técnica al planificar una estrategia de seguridad o al seleccionar las herramientas de análisis de seguridad para su organización.

5. Incorporación automática de PRs con actualización de dependencias

Probar

La complejidad en la cadena de suministro de software representa un riesgo significativo, y lo hemos cubierto de forma amplia, por ejemplo, en nuestros escritos en SBOM y SLSA. El talón de Aquiles para la mayoría de los equipos, sigue siendo la presencia de vulnerabilidades en las dependencias, usualmente dependencias indirectas que existen en varios niveles inferiores. Herramientas como Dependabot ayudan mediante la creación de pull requests (PRs) para actualizar dependencias. La pronta gestión de los PRs, especialmente cuando se trata de aplicaciones o servicios que no están en un proceso activo de desarrollo, demanda un alto nivel de disciplina en el ámbito de ingeniería.

Bajo las circunstancias adecuadas, ahora recomendamos la incorporación automática de PRs con actualización de dependencias. Para esto, el sistema necesita tener una cobertura de pruebas exhaustivas – no solo pruebas unitarias sino que también pruebas funcionales y de performance. La etapa de construcción del pipeline debe ejecutar todos estos test, incluyendo escaneo de seguridad. En pocas palabras, el equipo debe tener total confianza que cuando el pipeline se ejecuta correctamente, entonces el software está listo para llegar a producción. En estos casos, la actualización de los PRs con dependencias, deberían poder ser incorporados automáticamente, aún cuando incluyan actualizaciones mayores de versión en dependencias indirectas.

6. Product Thinking de Datos FAIR

Probar

El Product Thinking de Datos prioriza el tratar a los consumidores de datos como clientes, de esta manera asegura que tengan una experiencia fluida a lo largo de la cadena de valor. Esto comprende la facilidad en el descubrimiento, entendimiento, confianza, acceso y consumo de la información. El "Product Thinking" no es un concepto nuevo. En el pasado lo hemos integrado en el mundo operacional mientras construimos productos operacionales o microservicios. Esto también sugiere una nueva manera de formar equipos de larga duración y multifuncionales que puedan poseer y compartir información a lo largo de la organización. Para tener una mentalidad de producto a Datos, creemos que las organizaciones pueden poner en funcionamiento los principios FAIR (findable, accessible, interoperable and reusable) Nuestros equipos utilizan catálogos de datos como Collibra y DataHub para habilitar el descubrimiento de datos. Para garantizar la confianza publicamos métricas de calidad de datos y SLI como su vigencia, integridad y consistencia para cada producto de datos, y herramientas como Soda Core y Great Expectations automatizan los controles de calidad. La Observabilidad de Datos, se puede alcanzar mediante la ayuda de plataformas como Monte Carlo.

Hemos visto productos de datos convertirse en la base de múltiples casos de uso durante un tiempo. Esto viene acompañado de un tiempo de lanzamiento más rápido para posteriores casos de uso, a la vez que progresamos en la identificación e implementación de productos de datos enfocados a la entrega de valor. Es por esto que nuestro consejo es adoptar el product thinking de datos para datos FAIR.

7. OIDC para GitHub Actions

Probar

Una de las técnicas que recomendamos para implementar seguridad de confianza cero para CI/CD es autenticar tus pipelines de entrega contínua para el acceso a servicios en la nube via mecanismos federados de identificación como OpenID Connect (OIDC). Dado que GitHub Actions es ampliamente utilizado — y que esta importante técnica permanece poco utilizada — queremos llamar la atención sobre OIDC por GitHub Actions. De esta manera puedes evitar almacenar tokens de acceso de larga duración para tus recursos en la nube, y tus pipelines de entrega contínua no tendrán acceso directo a secretos. No obstante, asegúrate de limitar el acceso cuidadosamente de modo que las acciones se ejecuten realmente con el menor privilegio.

8. Provisión de monitores y alertas con Terraform

Probar

Infrastructure as code (IaC) es ahora un enfoque ampliamente aceptado para definir y provisionar entornos de alojamiento. Incluso con la continua evolución de herramientas y técnicas en esta área, Terraform sigue siendo la herramienta dominante para hacer IaC en recursos nativos de la nube. Sin embargo, la mayoría de los entornos de alojamiento actuales son combinaciones complejas de servicios nativos del proveedor de la nube, servicios de terceros y código personalizado. En estos entornos, hemos descubierto que los ingenieros a menudo recurren a una mezcla de Terraform para los recursos en la nube y scripts personalizados para el resto. Esto puede conducir a una falta de coherencia y repetibilidad en el proceso de aprovisionamiento. De hecho, muchos de los servicios de terceros que se utilizan comúnmente en entornos de alojamiento — incluyendo Splunk, Datadog, PagerDuty y New Relic — tienen proveedores Terraform que se pueden utilizar para aprovisionar

y configurar estos servicios. Por eso recomendamos que, además de los recursos en la nube, los equipos también suministren monitores y alertas con Terraform. Esto conduce a IaC con una mejor modularidad que es más fácil de entender y mantener. Como ocurre con todos los IaC, existe el riesgo de introducir inconsistencias cuando la configuración se modifica a través de otras interfaces. Para asegurar que el código de Terraform sigue siendo la fuente de la verdad, recomendamos que se deshabiliten los cambios de configuración a través de interfaces de usuario y APIs.

9. ReAct prompting

Probar

ReAct prompting es un método de prompting para LLMs destinado a mejorar la precisión de sus respuestas sobre otros métodos como chain-of-thought (CoT). IPublicado en un paper del 2022, funciona unificando razonamiento y acción (de ahí ReAct). Este enfoque ayuda a que las respuestas de los LLMs sean más entendibles y reducir las alucinaciones en comparación con CoT, dándole a los prompters una mejor oportunidad de conseguir lo que quieren. LangChain se desarrolló originalmente para soportar este estilo de prompting. Los agentes autónomos basados en ReAct prompting han demostrado ser algunas de las aplicaciones más ampliamente utilizadas de LLMs que nuestros equipos han estado construyendo. Recientemente, OpenAI ha introducido function calling en sus APIs para facilitar la implementación de ReAct y otros estilos de prompting similares sin tener que recurrir a herramientas externas como LangChain. Aún estamos en las primeras fases de definición de esta disciplina, pero hasta ahora, ReAct y sus descendientes han señalado el camino hacia algunas de las aplicaciones más interesantes de los LLMs.

10. Retrieval-Augmented Generation

Probar

Retrieval-Augmented Generation (RAG) es una técnica que combina memoria paramétrica y no-paramétrica, previamente entrenadas, para la generación de lenguaje. Nos permite tomar el conocimiento que poseen los LLMs pre-entrenados y aumentarlo con el conocimiento privado y contextual de nuestro dominio o industria. Con RAG, primero obtenemos un conjunto de documentos relevantes a partir de la memoria no-paramétrica (usualmente a través de una búsqueda similar en un almacenamiento de datos vectorial) para después usar la memoria paramétrica de los LLMs y generar una salida que es consistente con los documentos obtenidos inicialmente. Concluimos que RAG es una técnica efectiva para una variedad de tareas que requieren un intenso conocimiento sobre procesamiento de lenguaje natural – incluyendo responder preguntas, resumir y generar historias.

11. Modelamiento de fallos basado en el riesgo

Probar

El modelamiento de fallos basado en el riesgo es un proceso utilizado para entender el impacto, probabilidad y la habilidad de detectar las diferentes maneras en que un sistema puede fallar. Los equipos de entrega están empezando a utilizar esta metodología para diseñar y evaluar los controles necesarios para evitar dichos fallos. El enfoque se basa en la práctica de Análisis Modal de Fallos y Efectos (FMEA), , una técnica de puntuación de riesgo que existe desde los años 40 y con un historial de éxito en industrias que construyen sistemas físicos complejos como la aeroespacial y la automovilística. Como en estas industrias, las fallas de software también puede tener graves consecuencias — comprometiendo, por ejemplo, la salud y la privacidad — razón por lo que se ve un incremento en la necesidad de que los sistemas sean sometidos a rigurosos análisis. El proceso inicia identificando los posibles modos de fallo. Luego el equipo realiza un análisis a la causa principal

y asigna puntaje a la probabilidad de que la falla ocurra, la magnitud del impacto y la probabilidad de detectar la causa del fallo. Hemos comprobado que esto es más eficaz cuando los equipos interfuncionales repiten este proceso a medida que el sistema evoluciona. Hablando de seguridad, el modelamiento de fallos basado en el riesgo puede ser un gran complemento para modelado de amenazas y del análisis de rutas de ataque.

12. Lenguaje natural semi-estructurado para LLMs

Probar

Hemos tenido éxito en varias aplicaciones que hacen uso de lenguaje natural semi-estructurado para LLMs. Las entradas de datos estructuradas, como documentos JSON, son claras y precisas, y le dan al modelo una indicación del tipo de respuestas que se buscan. Limitar la respuesta de esta forma ayuda a estrechar el problema y puede producir resultados más certeros, particularmente cuando la estructura se ajusta a lenguaje específico de dominio (DSL) cuya sintaxis o esquema se proporciona al modelo. También hemos visto que acompañando la entrada estructurada de datos con comentarios en lenguaje natural o anotaciones produce una mejor respuesta que procesándolos de forma separada. Normalmente, el lenguaje natural simplemente se intercala con contenido estructurado al construir la instrucción. Al igual que con otros comportamientos de LLMs, no sabemos exactamente por qué funciona, pero la experiencia nos ha demostrado que añadir comentarios de lenguaje natural en código escrito por humanos también mejora la calidad de la salida de datos para los asistentes de código basados en LLMs.

13. Seguimiento de la salud sobre la deuda técnica

Probar

Seguimos observando las mejoras de los equipos en sus ecosistemas al tratar el índice de salud de la misma forma que los otros objetivos de nivel de servicio (SLOs) y priorizando mejoras en consecuencia, en vez de solamente enfocarse en hacer seguimiento de la technical debt. Al asignar recursos eficientemente para abordar los problemas con mayor impacto en la salud, equipos y organizaciones pueden reducir costos de mantenimiento a largo plazo y evolucionar productos de manera más eficiente. Este enfoque también mejora la comunicación entre stakeholders tanto técnicos como no técnicos, fomentando un entendimiento común del estado del sistema. A pesar de que las métricas pueden variar entre organizaciones (Ver este blog post para ejemplos) al final contribuyen a una sostenibilidad a largo plazo y se aseguran de que el software se mantenga adaptable y competitivo. En un panorama digital que cambia rápidamente, enfocarse en el seguimiento de la salud sobre la deuda técnica de los sistemas proporciona una estrategia estructurada y basada en evidencias para mantenerlos y mejorarlos.

14. Tests unitarios para reglas de alerta

Probar

La observabilidad y monitorización son esenciales para los equipos de desarrollo de software. Dada la naturaleza impredecible de ciertos eventos, es crucial crear mecanismos de alerta precisos con reglas complejas. Sin embargo, la validación de estas reglas sólo se produce cuando estos escenarios suceden. La técnica de tests unitarios para reglas de alerta permite a los equipos mejorar la definición de reglas mediante el testeo proactivo y redefinirlas incrementando la confianza en la forma en que las reglas se crean. Esto ayuda a reducir falsas alarmas y asegura que los verdaderos incidentes son identificados. Herramientas como Prometheus dan soporte a los tests unitarios para reglas; nuestros equipos ya están reportando sus beneficios en entornos del mundo real.

15. Seguridad de confianza cero para CI/CD

Probar

Si no está debidamente asegurada, la infraestructura y las herramientas que ejecutan nuestros pipelines de build y despliegue, pueden convertirse en una gran riesgo. Las pipelines requieren acceso a datos críticos y sistemas como el código fuente, credenciales y secretos para compilar y desplegar software. Esto hace que estos sistemas sean muy atractivos para actores maliciosos. Por lo tanto, recomendamos aplicar seguridad de confianza cero para pipelines CI/CD e infraestructura — confiando en ellas tan poco como sea posible. Esto abarca una serie de técnicas: Si está disponible, autentifica tus pipelines con tu proveedor en la nube a través de mecanismos de identidad federada como OIDC, en lugar de darles acceso directo a los secretos; implementa el principio de mínimo privilegio minimizando el acceso de cuentas individuales de usuario o runners, en lugar de emplear "cuentas de usuario god" con acceso ilimitado; utiliza tus ejecutores de forma efímera en lugar de reutilizarlos, para reducir el riesgo de exponer secretos de trabajos anteriores o ejecutar trabajos en ejecutores comprometidos; mantén el software de tus agentes y ejecutores actualizado; y monitoriza la integridad, confidencialidad y disponibilidad de tus sistemas CI/CD de la misma forma que monitorizarías tu software de producción.

Hemos visto que los equipos se olvidan de este tipo de prácticas particularmente cuando están acostumbrados a trabajar con infraestructura de CI/CD auto-gestionada en redes internas. Si bien todas estas prácticas son importantes en tus redes internas, estas se vuelven incluso más cruciales cuando se usa un servicio gestionado, ya que amplía el área de ataque y el radio de impacto aún más.

16. Health checks de dependencias para contrarrestar package hallucinations

Evaluar

Asegurar la cadena de distribución de software se ha convertido en una preocupación común entre los equipos de delivery, esto se refleja en el crecimiento del número de herramientas y técnicas en este ámbito, varias de las cuales, ya hemos cubierto anteriormente en el Radar. La creciente popularidad del uso de herramientas basadas en GenAI para ayudar en el proceso de desarrollo de software, ha introducido un nuevo vector de ataque a la cadena de distribución del software: package hallucinations. Creemos que es importante que los equipos que utilizan este tipo de herramientas, como GenAI, en su proceso de desarrollo, se mantengan alertas contra estos riesgos. Para lograr esto, los equipos pueden realizar Health checks de dependencias para contrarrestar los package hallucinations: mirar las fechas en las que fueron creadas, número de descargas, comentarios y calificaciones, el número de personas contribuidoras, el historial de la actividad, etc, antes de elegir adoptarlos. Algunas de estas revisiones pueden ser ejecutadas en los repositorios de dependencias y en GitHub, herramientas como deps.dev y Snyk advisor también pueden proporcionar información adicional. A pesar de que esto no es una técnica nueva, está adquiriendo una relevancia cada vez mayor a medida que los equipos experimentan cada vez más con herramientas GenAI en el ciclo de desarrollo de software.

17. Registro de decisiones del sistema de diseño

Evaluar

En un entorno acelerado de desarrollo de productos en donde las necesidades del usuario evolucionan constantemente, el diseño es un área en constante cambio. Esto significa que seguiremos requiriendo aportaciones para las decisiones de diseño. Tomando la idea de documentar decisiones de arquitectura mediante ADRs, comenzamos a adoptar un formato similar — el registro de decisiones del sistema de diseño — para documentar estas decisiones junto a su correspondiente razonamiento, resultados de la investigación y resultados de los experimentos. Comunicar las decisiones del sistema de diseño parecería ser una necesidad emergente en los equipos de producto. Hacerlo de esta sencilla manera también lo recomienda zeroheight. Esta técnica nos ha ayudado a reducir la curva inicial de aprendizaje en los equipos, así como avanzar en conversaciones y alinear el trabajo de equipos que comparten el mismo sistema de diseño.

18. GitOps

Evaluar

GitOps Es una técnica para el despliegue de aplicaciones a través del patrón control loop Un operador mantiene sincronizado el despliegue de la aplicación con la configuración, generalmente un repositorio de Git. La última vez que escribimos acerca de GitOps quedaba pendiente la definición del término, o nombre, por parte de la comunidad. En ese momento nos preocupaban las interpretaciones comunes de la técnica que se incluían acercamientos tipo “rama por ambiente” para la configuración, que podrían causar snowflakes as code. Además, las conversaciones en respecto a GitOps como alternativa a la entrega continua eran confusas. Desde entonces, los cuatro principios de GitOps han aclarado el alcance y la naturaleza de la técnica. Cuando se va más allá del alboroto y confusión, GitOps es una técnica útil que aprovecha la funcionalidad de un cúmulo, o clúster, de Kubernetes y crea oportunidades para separar las preocupaciones entre configurar una aplicación y la implementación del proceso de despliegue. Algunos de nuestros equipos han implementado GitOps como parte de su proceso de entrega continua con experiencias positivas, y es por esto que recomendamos su evaluación

19. Agentes autónomos impulsados por LLM

Evaluar

A medida que continúa el desarrollo de grandes modelos de lenguaje, existe un gran interés en crear agentes de IA autónomos. AutoGPT, GPT-Engineer and BabyAGI son ejemplos de agentes autónomos impulsados por LLM que desarrollan un LLM subyacente para comprender el objetivo que se les ha asignado y trabajar para lograrlo. El agente recuerda hasta dónde ha progresado, utiliza el LLM para razonar sobre qué hacer a continuación, toma acciones y comprende cuándo se ha cumplido el objetivo. Esto a menudo se conoce como razonamiento en cadena de pensamientos y, de hecho, puede funcionar. Uno de nuestros equipos implementó un chatbot de atención al cliente como agente autónomo. Si el chatbot no puede lograr el objetivo del cliente, reconoce su propia limitación y, en su lugar, redirige al cliente hacia un humano. Este enfoque definitivamente se encuentra en una etapa temprana de su ciclo de desarrollo: los agentes autónomos a menudo sufren una alta tasa de fallas e inciden en costosas tarifas de servicios de IA, y al menos una startup de IA se ha alejado del enfoque basado en agentes.

20. Orquestación de plataforma

Evaluar

Con la adopción generalizada de la ingeniería de plataformas, estamos viendo una nueva generación de herramientas que van más allá del modelo tradicional de plataforma-como-servicio (PaaS) y ofrecen contratos publicados entre desarrolladores y equipos de plataforma. El contrato podría implicar el aprovisionamiento de entornos en la nube, bases de datos, monitoreo, autenticación y más en un entorno diferente. Estas herramientas hacen cumplir los estándares organizacionales al tiempo que otorgan a los desarrolladores acceso de autoservicio a las variaciones a través de la configuración. Ejemplos de estos orquestación de plataformas sistemas incluyen Kratix y Humanitec Platform Orchestrator. Recomendamos que los equipos de plataforma evalúen estas herramientas como alternativa a la creación de su propia colección única de scripts, herramientas nativas e infraestructura como código. También hemos notado una similitud con los conceptos del Modelo de Aplicación Abierta (OAM) y su orquestador de referencia KubeVela, aunque OAM afirma ser más centralizado en la aplicación que en la carga de trabajo.

21. LLMs autohospedados

Evaluar

Los modelos grandes de lenguaje (LLMs, por sus siglas en inglés) generalmente requieren una infraestructura significativa de GPU para funcionar, pero ha habido un fuerte impulso para que funcionen en un hardware más modesto. La cuantización de un modelo grande puede reducir los requisitos de memoria, permitiendo que un modelo de alta fidelidad se ejecute en un hardware menos costoso o incluso en una CPU. Esfuerzos como llama.cpp hacen posible ejecutar LLMs en hardware que incluye Raspberry Pis, ordenadores portátiles y servidores de uso general.

Muchas organizaciones están desplegando LLMs auto-hospedados. Esto se debe a menudo a preocupaciones de seguridad o privacidad, o a veces, a la necesidad de ejecutar modelos en dispositivos periféricos. Ejemplos de código abierto incluyen GPT-J, GPT-JT y Llama. Este enfoque ofrece un mejor control del modelo al ajustarlo finamente para un caso de uso específico, una mayor seguridad y privacidad, así como acceso sin conexión. Aunque hemos ayudado a algunos de nuestros clientes a auto-hospedar LLMs de código abierto para completar código, te recomendamos que evalúes cuidadosamente las capacidades organizativas y el costo de operación de tales LLMs antes de tomar la decisión de auto-hospedarlos.

22. Ignorando las listas top 10 de OWASP

Resistir

El Top 10 de OWASP ha sido durante mucho tiempo una referencia de los riesgos de seguridad más críticos para las aplicaciones web. A pesar de ser ampliamente conocido hemos escrito anteriormente sobre su poca utilización en el proceso de desarrollo de software y advertido acerca de ignorar el TOP 10 de OWASP.

Algo que es menos conocido, es que OWASP también publica listas top 10 para otras categorías. La lista Top 10 de OWASP para LLMs, cuya primera versión estable fue liberada a principios de agosto, destaca riesgos como la inyección de comandos, el manejo de salidas inseguras y la contaminación de datos de entrenamiento entre otros, que personas y equipos que construyen este tipo de aplicaciones LLM deberían tener muy en cuenta. OWASP también ha lanzado recientemente la segunda versión de su lista Top 10 de OWASP para APIs. Dada la gran cobertura de información que estas listas dan para aplicaciones web, APIs, LLMs y más), además de su calidad y relevancia para el panorama de seguridad en constante cambio, extendemos nuestra recomendación para advertir a los equipos sobre no ignorar las listas Top 10 de OWASP.

23. Componentes web para aplicaciones web renderizadas en servidor

Resistir

Desde que los mencionamos en el 2014, los componentes web se han hecho populares y en general, nuestra experiencia ha sido positiva. Del mismo modo, hemos apoyado el renderizado de HTML en el servidor advirtiendo sobre el uso de SPA por defecto e incluyendo frameworks como Next.js y HTMX además de frameworks tradicionales del lado del servidor. Sin embargo, aunque es posible combinar ambos, también puede resultar muy problemático; por eso sugerimos evitar los componentes web para aplicaciones web renderizadas en servidor. Siendo una tecnología de navegador, no es trivial usar componentes en el servidor. Los frameworks han surgido para facilitar este proceso. A veces incluso usan un motor de navegador, pero la complejidad se mantiene. Peor aún que los problemas en la experiencia del desarrollador, es la experiencia del usuario: El rendimiento de la carga de página se ve impactado cuando los componentes web personalizados tienen que ser cargados e hidratados en el navegador, y aún con el uso de pre-renderización y retoques cuidadosos del componente, unflash de contenido sin estilos o algún movimiento inesperado en el diseño es inevitable. La decisión de privarse de componentes web puede tener consecuencias de gran impacto, como la que experimentó uno de nuestros equipos, cuando tuvieron que cambiar el diseño del sistema, basado en componentes web Stencil.

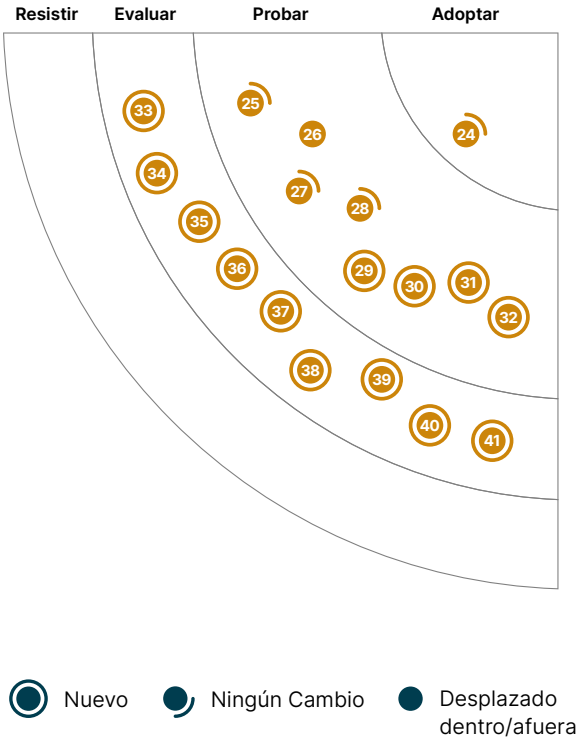
Plataformas

Adoptar
24. Colima

Probar
25. CloudEvents
26. DataOps.live
27. Google Cloud Vertex AI
28. Immuta
29. Lokalise
30. Orca
31. Trino
32. Wiz

Evaluar
33. ActivityPub
34. Azure Container Apps
35. Servicio OpenAI de Azure
36. ChatGLM
37. Chroma
38. Kraftful
39. pgvector
40. Pinecone
41. wazero

Resistir
—



24. Colima

Adoptar

Colima es ahora nuestra alternativa por defecto a Docker Desktop en macOS. Continuamos usándola en varios proyectos para provisionar el tiempo de ejecución (runtime) de los contenedores docker en una Lima VM, para configurar el CLI de docker en macOS y para gestionar el reenvío de puertos y el montaje de volúmenes. Colima puede configurarse para ejecutar containerd como su tiempo de ejecución, que también es el tiempo de ejecución en la mayoría de los servicios gestionados de Kubernetes mejorando la importante paridad entre dev-prod.

25. CloudEvents

Probar

Los eventos son mecanismos comunes en arquitecturas orientadas a eventos o en aplicaciones serverless. Sin embargo, los productores o proveedores cloud tienden a admitirlos de forma diferente, lo que dificulta la interoperabilidad entre plataformas e infraestructuras. CloudEvents es una especificación para usar formatos comunes en la descripción de los datos de eventos asegurando, de esta manera, la interoperabilidad entre servicios, plataformas y sistemas. Ofrece SDKs (kits de desarrollo) en varios lenguajes para integrar la especificación dentro de tu aplicación o herramientas de desarrollo. Nuestros equipos no solo lo usan para soluciones multiplataforma cloud, sino también para la especificación de eventos de dominio, entre otros escenarios. CloudEvents está mantenido por la Cloud Native Computing Foundation (CNCF) y ahora funciona como un proyecto incubadora, ganando cada vez más interés por parte de la industria.

26. DataOps.live

Probar

DataOps.live es una plataforma de datos que automatiza entornos en Snowflake. Inspirada en prácticas de DevOps DataOps.live te permite tratar a la plataforma de datos como a cualquier otra plataforma web al adoptar integración y entrega continua (CI/CD, por sus siglas en inglés), comprobaciones automáticas, observabilidad y administración de código. Nuestros equipos lo están usando para administrar el ciclo de vida de productos de datos, que incluye desarrollo, ramificación y despliegue tanto de código como de datos. Con su administración de entornos, automatizada, resulta muy fácil construir entornos basados en ramas de funcionalidad, modificarlos y destruirlos automáticamente. También vale la pena mencionar su prestación de especificación declarativa (SOLE) que hace posible tener una experiencia de desarrollo simplificada. Esto le permite a los equipos reducir el tiempo que toma construir productos de datos de meses a días. Nuestros equipos han estado utilizando DataOps.live exitosamente en producción, y es por eso que recomendamos esta plataforma a la hora de trabajar con Snowflake.

27. Google Cloud Vertex AI

Probar

Han ocurrido desarrollos significativos en el mundo de la IA desde la primera vez que publicamos al respecto Google Cloud Vertex AI. Desde mayo de 2023, Google ha presentado distintos servicios y funcionalidades que han enriquecido este ámbito. Estos incluyen, Model Garden, un repositorio con más de 100 modelos previamente entrenados; Generative AI Studio, una consola destinada a explorar y prototipar rápidamente modelos de IA generativa; y Extensiones Vertex AI las cuales proveen herramientas para desarrolladores totalmente manejadas para conectar modelos IA y data en tiempo real o acciones vía APIs. La plataforma ha evolucionado para ofrecer modelos de IA generativa y soporte de integración, y estamos emocionados de utilizarlo.

28. Immuta

Probar

Desde la última vez que escribimos acerca de Immuta, nuestros equipos han ganado significativa experiencia con esta plataforma de seguridad de datos. Su capacidad para definir políticas de datos y suscripción como código, control de versiones y el despliegue de estas políticas automáticamente en entornos superiores son características que están entre sus puntos más destacados. Su soporte ABAC permite asociar etiquetas a los orígenes de datos; si la misma etiqueta está asociada con el usuario, el acceso es concedido. Haciendo uso de Immuta y su integración con Snowflake hemos sido capaces de automatizar la concesión de acceso a un producto o conjunto de datos como autoservicio. Cuando un "usuario" solicita acceso a un producto o conjunto de datos, una vez que es aprobado, la etiqueta de producto de datos es asociada con el "usuario" como un atributo. Dado que el atributo en el "usuario" coincide con la etiqueta en el origen de datos, el acceso es concedido automáticamente por la política de Suscripción Global. de Immuta. También cabe mencionar las políticas de enmascarado de datos de Immuta que conservan la privacidad de la información al enmascar y restringir la Información de Identificación Personal (PII por sus siglas en inglés) a un usuario específico. El acceso adicional a información sensible se puede definir a un nivel mucho más granular usando políticas de seguridad a nivel de fila que aseguran que los usuarios sólo tendrán acceso a información específica para la que están autorizados a ver. Estamos contentos con Immuta, y es por eso que le estamos moviendo a Probar — ya que provee una buena experiencia al desarrollador y facilita el manejo de políticas de datos en organizaciones de gran tamaño.

29. Lokalise

Probar

Lokalise es una plataforma de localización totalmente automatizada que permite traducciones específicas según el contexto. Nuestros equipos utilizan la API de Lokalise en sus pipelines de ETL o en sus flujos de trabajo de desarrollo para traducir información localizable. Lokalise soporta múltiples formatos de archivo para las strings localizables. Un aspecto destacado es la capacidad de subir un fichero completo, donde cada par clave-valor se trata y se traduce como un registro separado. De forma transparente, aprovechamos la integración de Lokalise con Google MT para gestionar las traducciones. La interfaz web de usuario de Lokalise proporciona facilidad de acceso para revisiones manuales de las traducciones, acortarlas y reescribirlas según se considere adecuado. En el pasado, hemos destacado herramientas similares como Phrase. Nuestros equipos han tenido una buena experiencia con Lokalise y recomendamos que evalúes la plataforma para flujos de trabajo colaborativos de traducción.

30. Orca

Probar

Orca es una plataforma de seguridad de nube propietaria que identifica, prioriza y corrige riesgos de seguridad, así como problemas de cumplimientos regulatorios. Es compatible con los principales proveedores de nube y con configuraciones híbridas. Orca tiene un sinfín de consultas/reglas de seguridad para monitorear continuamente cargas de trabajo con configuraciones incorrectas, vulnerabilidades y problemas de cumplimiento regulatorio. Soporta también máquinas virtuales cloud, funciones serverless, contenedores y aplicaciones en Kubernetes para las cargas de trabajo desplegadas. Estas reglas de seguridad predeterminadas son constantemente actualizadas para estar al día con los estándares regulatorios y vectores de amenazas en constante evolución. Como Orca no tiene agente, ofrece una buena experiencia de desarrollo y es fácil de configurar. Otra característica notable es que facilita seguridad continua (shift-left security). Nuestros equipos usan Orca CLI para examinar vulnerabilidades y configuraciones incorrectas en imágenes de contenedores y plantillas de Infraestructura (IaC) como hooks en git previo a realizar commits, o como parte de flujos de trabajo de CI/CD. Asimismo, monitorea y examina continuamente los registros de contenedores (por ejemplo, AWS ECR) para detectar vulnerabilidades en imágenes base o dependencias débiles del sistema operativo en imágenes ya publicadas. Acorde a la experiencia de nuestros equipos, Orca ofrece una vista unificada de la postura de seguridad en el camino hacia producción, y por este motivo lo ubicamos en Probar.

31. Trino

Probar

Trino, anteriormente conocido como PrestoSQL, es un motor de consulta SQL distribuido de código abierto diseñado para consultas analíticas interactivas sobre big data. Está optimizado para funcionar tanto en on-premise como en la nube. Soporta la consulta de los datos donde se encuentren, incluidos Hive, Cassandra, bases de datos relacionales e incluso almacenes de datos propietarios. Para mecanismos de autenticación admite contraseñas, LDAP y OAuth. Para autorización y control de acceso, Trino proporciona la capacidad de otorgar acceso a nivel de catálogo, esquema y tabla. Nuestros equipos utilizan grupos de recursos segmentados en función a patrones de consumo, como visualización, informes o casos de uso de Machine Learning, para administrar y limitar el uso de recursos. La monitorización basada en JMX proporciona un rico conjunto de métricas para habilitar la atribución de costos a nivel de consulta o de usuario. Nuestros equipos utilizan Trino como puerta de entrada para el acceso a datos de varias fuentes. Cuando se trata de consultar datos a gran escala, Trino es una apuesta segura para nuestros equipos. Dato curioso, en el pasado presentamos Presto, es el proyecto de Facebook del que procede Trino, apareció por primera vez en Radar en noviembre de 2015.

32. Wiz

Probar

Wiz es otro de los contendientes en el panorama de las plataformas de seguridad en la nube que permite a sus usuarios prevenir, detectar y responder a riesgos de seguridad y amenazas en una sola plataforma. Wiz puede detectar y alertar sobre malas configuraciones, vulnerabilidades y secretos filtrados sea en artefactos que aún tienen que ser desplegados en producción (imágenes de contenedores, código de infraestructura) así como en cargas de trabajo en vivo (contenedores,

máquinas virtuales y servicios en la nube). También contextualiza los resultados al entorno de nube específico del cliente para permitir que los equipos de respuesta comprendan mejor el problema y prioricen las mitigaciones. Nuestros equipos han tenido una buena experiencia con Wiz. Encuentran que Wiz está evolucionando rápidamente y añadiendo nuevas funcionalidades, y aprecian que los habilita para detectar riesgos y amenazas antes que otras herramientas similares ya que busca constantemente cambios.

33. ActivityPub

Evaluar

Con la actual agitación en el espacio de plataformas de microblogging, el protocolo ActivityPub está ganando prominencia. ActivityPub es un protocolo abierto para compartir información como artículos (posts), publicaciones y eventos. Puede utilizarse para implementar una plataforma de redes sociales, pero su principal ventaja radica en que ofrece interoperabilidad entre diferentes plataformas de redes sociales. Esperamos que ActivityPub desempeñe un papel significativo en este espacio, pero lo mencionamos aquí porque nos intrigan las posibilidades más allá de los casos de uso obvios en las redes sociales. Un ejemplo es el soporte de ActivityPub para merge requests, recientemente propuesto por GitLab.

34. Azure Container Apps

Evaluar

Azure Container Apps es un Namespace de Kubernetes gestionado como un servicio que agiliza el despliegue de cargas de trabajo en contenedores mediante la eliminación de un mantenimiento intrincado de los clusters de kubernetes y sus componentes de infraestructura internos, así mismo minimizando las cargas operacionales y administrativas. Sin embargo, es esencial tener cuidado al considerar esta opción; que está actualmente en su fase de desarrollo y ha mostrado inconsistencias en la representación de sus capacidades en el portal de Azure y ha encontrado problemas de integración, en especial con el proveedor estándar de terraform para Azure con el retraso de la replicación de las funcionalidades actuales de la herramienta. Dicho esto, recomendamos analizar cuidadosamente esta herramienta.

35. Servicio OpenAI de Azure

Evaluar

Con el enorme interés en la inteligencia artificial generativa, han surgido muchas soluciones para acceder a los modelos principales. Si estás considerando usar o ya estás usando Azure, entonces merece la pena que evalúes Azure OpenAI Service. Proporciona acceso a los modelos GPT-4 y GPT35-Turbo y Embeddings de OpenAI a través de una API REST, un SDK para Python y una interfaz web. Los modelos pueden ser adaptados a tareas específicas como la generación de contenidos, realización de resúmenes, búsquedas semánticas, procesamiento de lenguaje natural o traducción de código. El ajuste fino del modelo es también posible a través del aprendizaje con pocas muestras (few-shot learning) y el ajuste de sus hiperparámetros. En comparación con la propia API de OpenAI, el servicio de OpenAI de Azure se beneficia de las características de seguridad y cumplimiento de nivel empresarial de Azure; también está disponible en más regiones, aunque la disponibilidad está limitada para cada una de las regiones geográficas más grandes y en el momento en que escribimos estas líneas, India no está incluida.

36. ChatGLM

Evaluar

Hay muchos modelos de lenguajes grandes (LLM) emergentes en el mundo de habla inglesa. Aunque estos modelos suelen estar previamente entrenados en varios idiomas, es posible que su rendimiento en otros idiomas no sea tan bueno como en el inglés. ChatGLM, desarrollado por la Universidad Tsinghua, es un modelo de lenguaje bilingüe abierto optimizado para la conversación en chino basado en la arquitectura del Modelo de lenguaje general Dado que el chino puede ser más complejo que el inglés con su diferente segmentación de palabras y gramática, es importante tener un LLM optimizado para el chino. Nuestro equipo descubrió que ChatGLM superó a otros LLM en precisión y robustez, cuando creamos una aplicación china de detección de emociones para un centro de llamadas. Teniendo en cuenta que muchos LLM no están disponibles en China debido a licencias o restricciones regionales, ChatGLM se convirtió en una de las pocas opciones de código abierto.

37. Chroma

Evaluar

Chroma Es un almacén de vectores de código abierto y una base de datos integrada, útil para mejorar aplicaciones impulsadas por modelos de lenguaje grandes (LLM) al facilitar el almacenamiento y la utilización del conocimiento del dominio en los LLM, que generalmente carecen de memoria interna. Particularmente en aplicaciones de texto a texto, Chroma puede automatizar el proceso complejo de generar word embeddings y analizar similitudes entre palabras y query embeddings, simplificando considerablemente las operaciones. También te brinda la opción de almacenar incrustaciones personalizadas, fomentando una combinación de automatización y personalización. A la luz de sus capacidades para mejorar la funcionalidad de las aplicaciones impulsadas por LLM, recomendamos a los equipos que evalúen Chroma, aprovechando su potencial para refinar la forma en que se integra el conocimiento del dominio en dichas aplicaciones.

38. Kraftful

Evaluar

Las plataformas de investigación UX como Dovetail ofrecen a las organizaciones una herramienta para entender y mejorar su experiencia de usuario. Con ella, los negocios son capaces de ganar un mejor conocimiento sobre las necesidades, preferencias y comportamientos de sus consumidores de forma fácil y rápida, recogiendo y analizando datos de feedback, encuestas, entrevistas, etc de clientes. El análisis de sentimiento, la segmentación de consumidores, la búsqueda de mercado o los análisis de datos y generación de conocimiento son tareas valiosas en el desarrollo de producto — estas encajan con lo que los LLMs saben hacer bien, por tanto, vemos un gran potencial como elemento disruptor en el campo del desarrollo de producto

Kraftful — auto-descrito como un co-piloto para creadores de producto — ha adquirido ventaja. Es sólo una versión beta y se accede a la lista de herramientas a través de email. Hemos jugado un poco con Kraftful y hemos visto grandes resultados. Puedes conectar más de 30 referencias de feedback de usuario en la plataforma y analiza los datos e identifica solicitudes de funcionalidades, quejas comunes, qué es lo que les gusta a los usuarios sobre el producto e incluso el nombre de los competidores. Para recolectar más detalles, puedes hacer preguntas como lo harías a ChatGPT o a Google Bard — el beneficio aquí es la optimización de tus datos. Una vez se prioriza lo que se va a usar del feedback recibido, Kratful genera historias de usuario basadas en toda la información subyacente — incluyendo criterios de aceptación — convirtiéndolo en un gran asistente incluso para gerentes de producto y analistas de negocio experimentados.

39. pgvector

Evaluar

Con el auge de las aplicaciones impulsadas por la Gen AI, observamos un patrón de almacenamiento y búsqueda eficiente de similitud de vectores incrustados o embeddings. pgvector es una extensión de búsqueda de similitud vectorial de código abierto para PostgreSQL. Nos gusta especialmente porque permite buscar embeddings en PostgreSQL sin necesidad de trasladar los datos a otro sistema de almacenamiento solo para buscar similitudes. Aunque existen varios motores de búsqueda vectorial, especializados, queremos que evalúes pgvector.

40. Pinecone

Evaluar

Pinecone es una base de datos vector, completamente gestionada, developer-friendly y cloud-native con una API sencilla y sin problemas de infraestructura. Pinecone funciona con queries filtradas con baja latencia y con una escala de billones de vectores. Nuestros equipos han encontrado bases de datos propietarias y Pinecone en concreto, muy útiles y fáciles de utilizar en casos de uso cómo almacenar los conocimientos básicos de un equipo o los contenidos de un portal de ayuda (help desk) mejor que afinar LLMs complejas.

41. wazero

Evaluar

wazero es un entorno de ejecución de WebAssembly ((WASM) libre de dependencias, escrito en Go. Aunque el propio entorno es agnóstico a lenguaje, wazero puede ser especialmente interesante para desarrolladores de Go ya que ofrece una forma conveniente de extender programas escritos en este lenguaje con módulos WASM escritos en cualquiera de los lenguajes soportados. Al no depender de CGO, tus aplicaciones Go pueden ser compiladas para otras plataformas fácilmente. Aunque existen muchas opciones de entornos de ejecución para WASM, creemos que merece la pena tener en cuenta wazero.

Herramientas

Adoptar

42. dbt
43. Mermaid
44. Ruff
45. Snyk

Probar

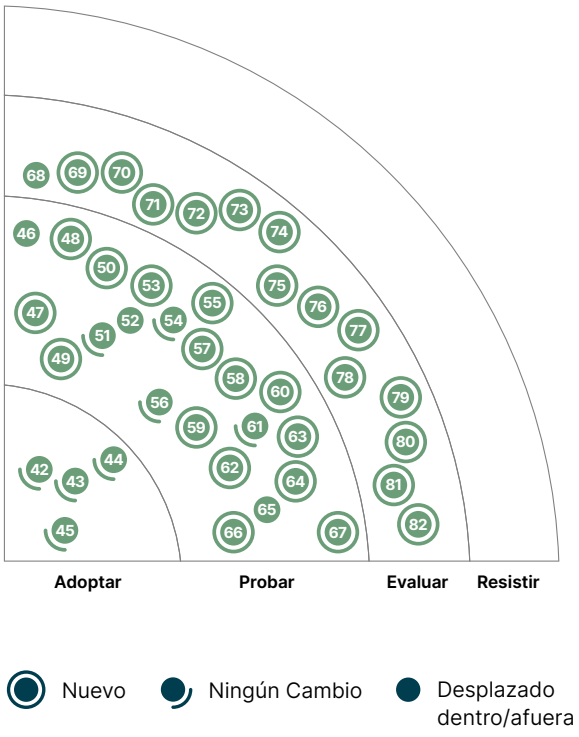
46. AWS Control Tower
47. Bloc
48. cdk-nag
49. Checkov
50. Chromatic
51. Cilium
52. Cloud Carbon Footprint
53. Pruebas de estructura de contenedor
54. Devbox
55. DX DevEx 360
56. GitHub Copilot
57. Insomnia
58. Plugin de cliente HTTP para IntelliJ
59. KEDA
60. Kubeconform
61. mob
62. MobSF
63. Mocks Server
64. Defensa en la ejecución de Prisma
65. Terratest
66. Thanos
67. Yalc

Evaluar

68. ChatGPT
69. Codeium
70. GitHub merge queue
71. Google Bard
72. Google Cloud Workstations
73. Gradio
74. KWOK
75. Llama 2
76. Maestro
77. LLM de código abierto para codificación
78. OpenCost
79. OpenRewrite
80. OrbStack
81. Pixie
82. Tabnine

Resistir

—



42. dbt

Adoptar

dbt continúa siendo nuestra herramienta preferida para transformación de datos en flujos de trabajo ELT. Nos gusta que se presta al rigor de la ingeniería y habilita prácticas como modularidad, testabilidad y reusabilidad de transformaciones SQL. dbt está disponible en ambos formatos, como código libre y como software como servicio, y tiene un ecosistema sano, incluyendo un repositorio con paquetes para pruebas unitarias, calidad de datos y observabilidad de datos, por nombrar unos cuantos. Algunos paquetes que merece la pena mencionar son dbt-expectations y dbt-unit-testing que facilitan las verificaciones de la calidad de los datos y las pruebas unitarias de las transformaciones, respectivamente. dbt se integra bien con una variedad de almacenes de datos en la nube, lake houses y bases de datos como Snowflake, BigQuery, Redshift, Databricks y Postgres. Cuando trabajamos con datos estructurados en los que es posible establecer transformaciones en SQL, nuestros equipos prefieren dbt — que es el motivo por el que lo movemos a Adoptar.

43. Mermaid

Adoptar

Mermaid te permite generar diagramas a partir de un lenguaje de marcado similar a Markdown. Desde la última vez que lo presentamos en el Radar, Mermaid ha agregado soporte para muchos más diagramas e integraciones con repositorios de código fuente, entornos de desarrollo integrado (IDE por sus siglas en inglés) y herramientas de gestión del conocimiento. Es importante destacar que está integrado de forma nativa en repositorios populares como GitHub y GitLab, permitiendo la inserción y actualizaciones fáciles de los diagramas de Mermaid en medio de documentación en Markdown. Muchos de nuestros equipos se decantan hacia Mermaid como su herramienta de referencia para diagramas como código debido a su facilidad de uso, multitud de integraciones y amplia variedad de tipos de diagramas soportados y que continúan en aumento.

44. Ruff

Adoptar

Ruff es un linter relativamente nuevo para Python. Cuando se trata de linters, para nosotros la pregunta no es cuándo usar un linter sino cual linter usar. Ruff se destaca por la mejor experiencia de uso y su velocidad. Tiene más de 500 reglas integradas y reemplaza fácilmente a Flake8, incluyendo muchos de los plugins de linter. Las afirmaciones del equipo de Ruff sobre su desempeño están confirmadas por nuestra experiencia; realmente es al menos un orden de magnitud más rápido que otros linters, lo cual es un gran beneficio, porque ayuda a reducir los tiempos de compilación en grandes bases de código. Por estas razones, Ruff se ha convertido en nuestra opción predeterminada como linter de Python.

45. Snyk

Adoptar

Snyk ofrece tanto pruebas estáticas de seguridad de aplicaciones (SAST) como análisis de componentes de software (SCA) para ayudarte a encontrar, resolver y monitorizar problemas de seguridad durante todo el ciclo de vida del desarrollo de software. Su amplia gama de funcionalidades está diseñada para acelerar el ciclo de feedback, favoreciendo el enfoque de desplazar las pruebas hacia la izquierda (shift-left) en lugar del antipatrón sándwich de seguridad Snyk destaca por ser una de las mejores plataformas de seguridad disponibles al día de hoy, siendo capaz de identificar un número importante de problemas de seguridad, gracias a un equipo dedicado a la investigación y actualización de su base de datos de vulnerabilidades. Aun así, hay puntos de mejora en la plataforma, por ejemplo el panel actualmente no proporciona una manera fácil de filtrar información específica

para generar accionables; dependiendo del ecosistema del lenguaje usado, las integraciones basadas en SCA pueden producir falsos positivos en comparación con las integraciones basadas en pipelines, ya que Snyk tiene que adivinar cuáles fueron las dependencias resueltas; la resolución automatizada no es exitosa de forma consistente; y se requiere invertir tiempo en la integración para alcanzar una protección adecuada o definir una SBOM. en entornos altamente regulados. A pesar de estas deficiencias, muchos de nuestros clientes empresariales han adoptado Snyk; nosotros también lo estamos usando para nuestro uso en TI.

46. AWS Control Tower

Probar

La gestión de cuentas multi-equipo es un desafío en AWS, especialmente en la configuración y gobierno. AWS Control Tower AWS Control Tower Account Factory for Terraform (AFT). AFT te permite aprovisionar personalizaciones para mandar webhooks o tomar acciones específicas que permitan la integración de otras herramientas para iniciar tareas como parte del proceso de creación de cuenta. Uno de los casos de uso aprovechados por nuestro equipo fue gestionar un conjunto de elementos desde el primer momento para cuentas con configuraciones de un solo uso para establecer una base y crear acceso para roles para GitHub Actions. Esto dió como resultado darles a los desarrolladores una cuenta con seguridad basada en una VPC totalmente integrada, lista para recibir carga de trabajo a través de GitHub Actions. Nuestro equipo ha conseguido excelentes resultados al utilizar AWS Control Tower para gestionar cuentas, como un control de acceso único para varios equipos y al aprovechar AFT en sus cargas de trabajo.

47. Bloc

Probar

Bloc es una librería de gestión de estados reactiva para Flutter. Entre todas las opciones de gestión de estados disponibles para Flutter, destacamos Bloc porque nuestros equipos han tenido una buena experiencia con ella a la hora de construir aplicaciones móviles complejas. Estructurar el código siguiendo el patrón BLoC tuvo como resultado una separación clara entre la lógica de negocio y la capa de presentación dado que los widget UI se comunican con dicha lógica vía streams y event sinks. Bloc también ofrece un buen soporte mediante plugins para los IDEs IntelliJ y VSCode.

48. cdk-nag

Probar

cdk-nag identifica y reporta problemas de seguridad y cumplimiento regulatorio en aplicaciones AWS CDK o plantillas de CloudFormation. Viene con varios de los llamados paquetes de reglas: un paquete general de AWS que incluye comprobaciones de lo que AWS considera buenas prácticas, además de paquetes para el cumplimiento de HIPAA, NIST y PCI. Se pueden añadir reglas adicionales según sea necesario. Las reglas pueden resultar en errores o advertencias, y ambas son incluidas en reportes generados por la herramienta. Cuando se presentan errores, el comando cdk deploy no hará despliegues. Si la causa del error no puede ser arreglada a tiempo, aún se puede desplegar con el error presente pero suprimido. Obviamente, esto solo debe hacerse en casos excepcionales.

49. Checkov

Probar

Checkov es un scanner estático de seguridad para infraestructura como código (IaC). Soporta múltiples lenguajes de infraestructura, entre ellos Kubernetes manifests, Helm charts, CloudFormation templates y Terraform. Fácil de desplegar en pipelines de CI/CD, protege de posibles brechas de seguridad en diversas configuraciones de infraestructura cloud. Aprovechando un conjunto de reglas

predeterminadas, identifica escenarios de seguridad comunes con consejos de solución detallados disponibles en su sitio web. Checkov admite reglas personalizadas, y utiliza YAML para definir directivas simples o Python para directivas complejas. Nuestros equipos han utilizado Checkov con éxito para mejorar la seguridad durante las implementaciones de infraestructura, teniendo en cuenta las advertencias tempranas que proporciona sobre posibles problemas antes del despliegue.

50. Chromatic

Probar

Chromatic es una herramienta visual de pruebas de regresión para interfaces de usuario de aplicaciones web. Funciona tomando pantallazos de los componentes de la interfaz y comparándolos contra pantallazos anteriores, cuando estos cambian. Es un servicio alojado que se integra con servicios populares de hosting de código en la nube. Construido sobre Storybook, realiza pruebas de regresión visual a nivel de componentes. Puede renderizar los componentes en distintos anchos de ventanas del navegador para pruebas de responsividad e integrarse con flujos de trabajo de CI, generando un conjunto de cambios de la interfaz por cada commit, lo que facilita su revisión. En el ámbito visual, nuestros equipos encuentran que Chromatic es mejor que otras herramientas. La capacidad de destacar visualmente los cambios la hace particularmente útil.

51. Cilium

Probar

eBPF es famoso por su transparencia, alto rendimiento y baja sobrecarga. Por esta razón la comunidad de nube nativa ha estado explorando su uso para malla de servicios sin sidecar. Cilium es un proyecto de código abierto que provee interconexión de redes, seguridad y observabilidad para entornos de nube nativa como clusters de Kubernetes y otras plataformas de orquestación de contenedores. Proporciona una red simple de capa 3 para enrutamiento o superposición y admite el protocolo L7. Desacoplando la seguridad del direccionamiento, Cilium puede jugar un rol significativo como una nueva capa de protección. Hemos visto la adopción de Cilium en algunos proveedores de computación en la nube y también ha sido utilizado en proyectos en Thoughtworks. La comunidad todavía está debatiendo si eBPF puede reemplazar el uso del patrón sidecar, pero parece haber consenso en que algunas características de las mallas de servicios no pueden o no deben ser ejecutadas en el kernel. Además, aplicar Cilium también requiere experiencia previa relacionada con eBPF. Basándonos en los resultados positivos en nuestro proyecto, te recomendamos utilizar esta tecnología.

52. Cloud Carbon Footprint

Probar

Cloud Carbon Footprint (CCF) es una herramienta de código abierto que estima las emisiones de carbono para las cargas de trabajo en la nube en los principales proveedores de servicios de nube. Consulta las API de la nube para obtener datos sobre el uso de recursos y utiliza múltiples fuentes para rastrear las emisiones de carbono. Siguiendo una metodología publicada, CCF los combina en estimaciones de emisiones y proporciona una visualización de los datos a lo largo del tiempo. Los proveedores de nube han comenzado a agregar ofertas similares a sus plataformas, pero las organizaciones aún están implementando CCF porque reúne las características siguientes: es de código abierto, está diseñado para ampliarse, funciona en múltiples nubes y tiene una metodología publicada transparente. Adicionalmente, incluye estimaciones de emisiones de alcance 2 y alcance 3 — para el uso de electricidad y la producción de hardware, respectivamente. En nuestros experimentos, las estimaciones entre diferentes herramientas han variado, lo cual no es una gran sorpresa dado que todas las herramientas en este espacio hacen estimaciones y multiplican números

estimados. Sin embargo, decidirse por una herramienta, tomar una línea de base y mejorar a partir de esa línea de base es el escenario de uso clave que hemos encontrado, y herramientas como Kepler pueden reducir la necesidad de estimaciones en el futuro. CCF también ofrece recomendaciones de optimización obtenidas de GCP y AWS, que no solo ayudan a reducir su huella de carbono en la nube, sino que también pueden convertirse en parte de una estrategia más amplia de optimización de costos de la nube. Thoughtworks es un contribuyente significativo a CCF.

53. Pruebas de estructura de contenedor

Probar

Pruebas de estructura de contenedor (CST) es una herramienta desarrollada por Google para probar la estructura de una imagen de contenedor. Se puede utilizar CST para verificar la existencia o ausencia de un determinado archivo en el sistema de archivos de la imagen, para verificar el contenido de un archivo, para verificar la salida o los errores dentro de un comando específico emitido en el contenedor y para verificar los metadatos de imagen del contenedor (por ejemplo, etiquetas, punto de entrada y comando) que ayuda a garantizar el cumplimiento de CIS Docker Benchmark. Hemos tenido buenas experiencias con CST y te recomendamos que puedas probarlo. Además de prevenir vulnerabilidades — verificar si el contenedor está exponiendo puertos innecesarios — también lo usamos para validar que cada contenedor Docker cumpla con todos los requisitos necesarios para su despliegue y ejecución de una aplicación en la plataforma empresarial. Uno de estos requisitos era tener instalado un agente de observabilidad en la imagen. Es importante tener en cuenta que Google no admite oficialmente CST, lo que podría afectar el mantenimiento.

54. Devbox

Probar

Devbox es una herramienta basada en terminal que provee una interfaz accesible para crear entornos de desarrollo reproducibles por proyecto, aprovechando el administrador de paquetes Nix sin usar máquinas virtuales o contenedores. Nuestros equipos la utilizan para eliminar la discordancia en la versión y configuración de las herramientas de CLI y scripts personalizados en sus entornos de desarrollo por proyecto, además de la estandarización que proveen los administradores de paquetes de cada lenguaje. Descubrieron que agiliza notablemente su flujo de trabajo de onboarding ya que una vez que ha sido configurado para una base de código, solo toma un comando del CLI (devbox shell) para levantar una nueva máquina. Devbox soporta hooks de terminal, scripts personalizados y generación de devcontainer.json para la integración con VSCode.

55. DX DevEx 360

Probar

DX DevEx 360 es una plataforma de encuestas que ayuda a buscar los principales indicadores de la productividad de las desarrolladoras, enfocándose en las fricciones de su trabajo diario, como procesos de revisión de código, calidad de código, concentración profunda y más. La plataforma está diseñada por Nicole Forsgren y Margaret-Anne Storey, que lideraron trabajos anteriores en DORA y SPACE, entre otras expertas.

Nuestros equipos de ingeniería de plataforma han usado DX DevEx 360 satisfactoriamente para comprender el sentimiento de las desarrolladoras e identificar puntos de fricción para hacerlos constar en la hoja de ruta de la plataforma. Comparado con usar herramientas similares, con DX DevEx 360 recibimos una respuesta de 90% o más, a menudo detallada con comentarios de desarrolladoras sobre problemas e ideas de mejora. También apreciamos que la herramienta da transparencia de los resultados a las ingenieras de la compañía en vez de solo mostrarlo a las personas encargadas del

equipo y permite desglosar equipo por equipo, permitiendo obtener una mejora continua teniendo en cuenta el contexto de cada equipo.

56. GitHub Copilot

Probar

GitHub Copilot es utilizado por muchos de nuestros equipos para ayudarles a escribir código más rápido. En general, a la mayoría de nuestros desarrolladores les parece muy útil y se molestarían si se lo quitáramos. Hemos estado recopilando y compartiendo muchas de nuestras experiencias con Copilot a través de una serie sobre la IA generativa y una guía sobre cómo comenzar con Copilot. Tenga en cuenta que GitHub Copilot se puede usar con cualquier base de código, no solo con bases de código alojadas en GitHub.

También estamos emocionados de que la función de chat de Copilot de la hoja de ruta de Copilot X se ha vuelto más ampliamente disponible desde la última vez que la presentamos en el Radar. Es una poderosa adición a la funcionalidad de asistencia en línea de Copilot. La disponibilidad de una interfaz de chat dentro del entorno de desarrollo integrado (IDE) mejora la facilidad de descubrimiento de información comúnmente buscada y la integración con el editor abierto facilita la exploración de errores o solicitar ayuda al chat con tareas relacionadas con el código actualmente en foco.

57. Insomnia

Probar

Desde que Postman anunció en mayo de 2023 que comenzaría a retirar el modo Scratch Pad con sus capacidades offline, los equipos que necesitan mantener la información de sus workspaces de APIs fuera de servidores de terceros han tenido que buscar alternativas. Insomnia es una de estas alternativas: es una aplicación de escritorio gratuita y de código abierto diseñada para probar, desarrollar y depurar APIs. Aunque Insomnia permite sincronización online, también permite mantener la información de los workspaces de APIs offline. Nuestros equipos no han encontrado problemas al migrar desde Postman para las pruebas manuales de APIs ya que las funcionalidades son similares se pueden importar las colecciones de Postman. Pese a las positivas experiencias de nuestros equipos con Insomnia, seguimos vigilando otras alternativas en desarrollo — desde herramientas de interfaz gráfica de usuario como Insomnia que son alternativas de reemplazo inmediato, a herramientas de línea de comandos como HTTPie, o extensiones para entornos de desarrollo integrados como IntelliJ HTTP client plugin.

58. Plugin de cliente HTTP para IntelliJ

Probar

El Plugin de cliente HTTP para IntelliJ permite a desarrolladores crear, editar y ejecutar peticiones HTTP desde el propio editor de código, simplificando el proceso de desarrollo cuando se están construyendo y consumiendo APIs. Este plugin es cada vez más popular entre nuestros equipos, quienes consideran sus funcionalidades prácticas y fáciles de usar. Las funciones más destacadas incluyen soporte para el uso de archivos privados que protegen claves sensibles al excluirlas de Git automáticamente, control de versiones y la capacidad de usar variables, lo que mejora notablemente la experiencia de desarrollo. Recomendamos probar esta herramienta por la capacidad que tiene de optimizar los flujos de trabajo de los desarrolladores y su capacidad para reforzar las medidas de seguridad.

59. KEDA

Probar

KEDA, cuyo nombre se puede traducir al español como “escalador automático basado en eventos de Kubernetes”, hace exactamente lo que su nombre sugiere: permite el escalado de un clúster de Kubernetes en función de la cantidad de eventos que deben ser procesados. En nuestra experiencia, es preferible usar indicadores adelantados como el tamaño de la cola, en lugar de indicadores retrasados como el uso de CPU. KEDA soporta diferentes fuentes de eventos e incorpora un catálogo de más de 50 escaladores para varias plataformas en la nube, bases de datos, sistemas de mensajería, sistemas de telemetría, integración/despliegue continuo y más. Nuestros equipos reportan que la facilidad con la que KEDA se integra les ha permitido mantener funcionalidad en micro servicios en Kubernetes para los que en otros casos habrían considerado migrar parte del código de gestión de eventos a funciones serverless.

60. Kubeconform

Probar

Kubeconform es una herramienta simplificada para validar manifiestos de Kubernetes y definiciones de recursos personalizados (CRD en inglés). Fácilmente desplegable en pipelines de CI/CD o máquinas locales, fomenta la confianza al validar los recursos antes del despliegue, mitigando así potenciales errores. Dado su historial mejorando el aseguramiento operacional, especialmente en recursos con plantillas compartidas entre equipos, recomendamos probar Kubeconform para reforzar la seguridad y eficiencia de sus procesos de validación de recursos.

61. mob

Probar

mob es una herramienta de línea de comandos que, de manera transparente, facilita el git handover en pairing remoto o programación en grupo. Oculta toda la parafernalia del control de versiones detrás de una interfaz de línea de comandos, lo que simplifica las sesiones de programación en grupo. También proporciona consejos específicos sobre cómo participar de forma remota, por ejemplo, “tomar control de la pantalla” en Zoom en lugar de terminar el compartir pantalla, asegurando que la disposición en el monitor no cambie para los participantes. Varios de nuestros equipos recomiendan encarecidamente mob, y se ha convertido en una parte integral de nuestra cadena de herramientas en pair programming remoto o programación en grupo.

62. MobSF

Probar

MobSF es una herramienta de pruebas automatizadas de seguridad estática y dinámica de código abierto diseñada para detectar vulnerabilidades de seguridad en aplicaciones móviles iOS y Android. Escanea tanto el código fuente como los archivos binarios de la aplicación y proporciona informes detallados sobre las vulnerabilidades encontradas. MobSF se distribuye en forma de imágenes Docker e incorpora una APIs REST de fácil uso. Además, a través de mobsfscan, puede integrarse en flujos de trabajo de CI/CD. Nuestra experiencia utilizando MobSF para la prueba de seguridad en aplicaciones Android ha sido positiva; recomendamos probarlo para sus necesidades de pruebas de seguridad en aplicaciones móviles.

63. Mocks Server

Probar

Mocks Server es una herramienta de simulación de API basada en Node.js valorada por nuestros equipos por su capacidad para replicar complejas respuestas API, encabezados y códigos de estado. La generación de respuesta dinámica admite la simulación de diversos escenarios, lo que permite realizar pruebas rigurosas de las interacciones con las API. Los simulacros (mocks) pueden describirse como YAML o JSON y administrarse a través de CLI, API REST o código JavaScript. Las características de Mocks Server incluyen funciones de coincidencia de solicitudes, proxy y grabación-reproducción, que facilitan la emulación de interacciones API realistas. Nos gusta especialmente la integración con contenedores Docker, lo que hace que sea más fácil el despliegue del servidor de manera consistente entre entornos para que pueda ser versionado y mantenido como otro artefacto del ecosistema. Su enfoque sencillo se alinea con nuestro énfasis en la simplicidad y eficiencia en los procesos de desarrollo. Esperamos utilizar Mocks Server más ampliamente a medida que nuestra estrategia de prueba evolucione junto con nuestras soluciones.

64. Defensa en la ejecución de Prisma

Probar

Prisma runtime defense, que es una parte de la suite de Prisma Cloud, ofrece un nuevo enfoque a la seguridad de contenedores. Emplea un mecanismo para construir un modelo del comportamiento esperado de un contenedor y luego detecta y bloquea las actividades anómalas cuando se encuentra una variación durante el tiempo de ejecución. Monitorea los procesos del contenedor, actividades de la red y sistemas de archivos para encontrar patrones y cambios que indiquen que un ataque podría estar en proceso y realiza bloqueos de acuerdo a las reglas configuradas. Los modelos que aprenden lo que se identifica como comportamientos “normales” son construidos tanto por el análisis estático de imágenes de docker, como por análisis del comportamiento dinámico por un periodo preconfigurado. Nuestros equipos han encontrado prometedores los resultados en su uso.

65. Terratest

Probar

Terratest sigue siendo una interesante alternativa para probar infraestructura. Es una librería de Golang que facilita la escritura de pruebas automatizadas. Usando herramientas de infraestructura como código, por ejemplo Terraform, puedes crear componentes de infraestructura reales (servidores, firewalls, o balanceadores de carga) para desplegar aplicaciones sobre ellos y luego validar el comportamiento esperado con Terratest. Al final de la prueba, Terratest puede retirar las aplicaciones y hacer una limpieza de los recursos. Nuestros equipos reportan que esta forma de probar componentes desplegados de infraestructura fomenta la confianza en la infraestructura como código. Vemos a nuestros equipos escribiendo una variedad de pruebas de seguridad para los componentes de las aplicaciones y su integración. Esto incluye detectar desconfiguraciones, verificar control de accesos (ej. Para verificar si algún rol o permiso IAM está configurado correctamente o para asegurar que sólo los usuarios verificados tengan acceso a recursos específicos) y probar la seguridad en la red para verificar la prevención de tráfico no autorizado a recursos sensibles, por nombrar algunos. Esto permite mantener las pruebas de seguridad a la izquierda y entrega retroalimentación durante el proceso de desarrollo en sí mismo.

66. Thanos

Probar

A pesar de que Prometheus continúa siendo una opción sólida como herramienta autogestionada de observabilidad, muchos equipos que manejan sistemas distribuidos nativos en cloud se encuentran

con sus limitaciones por soportar un único nodo a medida que crecen sus métricas en cardinalidad y volumen total, así como cuando empiezan a necesitar una configuración con alta disponibilidad. Thanos complementa a Prometheus al añadir funcionalidades que le vuelven apto para monitorización a gran escala, largo plazo y alta disponibilidad. Hace esto por ejemplo, al introducir componentes que leen datos desde instancias de Prometheus y los guardarán en almacenes de objetos, manejan retención y compactación en el almacenamiento de objetos y federan las búsquedas a través de múltiples instancias de Prometheus. Nuestros equipos han notado que la migración desde Prometheus es descomplicada, debido a que Thanos mantiene compatibilidad con la API de consultas de Prometheus. Esto significa que pueden seguir utilizando los tableros existentes, herramientas de alertas y otras herramientas que se integran con la API de consultas de Prometheus. A pesar de que nuestros equipos han tenido éxito con Thanos, también recomendamos tener en consideración Cortex, como otra opción para extender las funcionalidades de Prometheus.

67. Yalc

Probar

Yalc es un sencillo repositorio local de paquetes JavaScript y una herramienta para publicar y utilizar paquetes en un entorno de desarrollo local. Es una alternativa más fiable que el comando npm link que tiene algunas limitaciones. Yalc es útil cuando se trabaja con múltiples paquetes, especialmente cuando algunos utilizan yarn y otros npm. También es útil para probar los paquetes localmente antes de publicarlos en un registro remoto. Según nuestra experiencia, Yalc es valioso en una configuración multi-paquete y acelera flujos de trabajo en el front-end y de otras aplicaciones basadas en JavaScript.

68. ChatGPT

Evaluar

ChatGPT sigue captando atención. Imaginativos casos de uso y enfoques innovadores de prompting están expandiendo su utilidad con el paso del tiempo. GPT4, el modelo de lenguaje grande (LLM) que impulsa ChatGPT, ahora también tiene la capacidad de integrarse con herramientas externas como repositorios de gestión de conocimientos, entornos de codificación en sandbox o búsquedas web. La reciente introducción de ChatGPT Enterprise puede ayudar a disminuir la preocupación sobre temas de propiedad intelectual, a la vez que proporciona características empresariales como el seguimiento de uso y una mejor gestión de usuarios mediante SSO.

Aunque la capacidad de ChatGPT para escribir código ha sido muy elogiada, creemos que las organizaciones deberían considerar usarlo en todo el ciclo de vida del software para mejorar la eficiencia y reducir errores. Por ejemplo, ChatGPT podría proporcionar nuevas perspectivas o dar sugerencias adicionales para tareas tan diversas como el análisis de requisitos, diseño arquitectónico o ingeniería inversa de sistemas legacy. Creemos no obstante que ChatGPT es más adecuado para ser usado como entrada a procesos — como ayuda para un primer borrador de historias o una plantilla para tareas de codificación — en lugar de una herramienta que produzca resultados totalmente elaborados. Dicho esto, y con sus capacidades mejorando cada semana, algunas tareas de programación ahora pueden ser completamente posibles mediante preguntas cuidadosamente formuladas, lo cual es un arte en sí mismo.

69. Codeium

Evaluar

En el ámbito de asistentes de código potenciados por IA, Codeium es uno de los productos más prometedores. De manera similar a Tabnine, Codeium intenta abordar algunas de las más grandes preocupaciones que las compañías tienen con respecto a los asistentes de código: Reducen un poco

la angustia acerca del uso de licencias de código abierto, ya que no entrenan a sus modelos con código de repositorios con licencia no permisiva. También ofrecen auto hospedado de la herramienta de manera tal que no tengas que enviar fragmentos de código a un servicio de terceros. Codeium se destaca por su amplio soporte en entornos de desarrollo integrado (IDE en inglés) y servicios para notebooks, y aunque no ha estado en el mercado tanto tiempo como GitHub Copilot o Tabnine, nuestras primeras impresiones del producto han sido positivas.

70. GitHub merge queue

Evaluar

Hace tiempo que somos defensores de ramas de desarrollo de corta duración que son integradas frecuentemente en la rama principal del código, preparada para ser desplegada en cualquier momento. Esta práctica de desarrollo basada en trunk-based development va muy de la mano con integración continua y, cuando las condiciones lo permiten, se obtienen ciclos de feedback más rápidos y un flujo de desarrollo más eficiente. Sin embargo, no todo el mundo está a favor de este enfoque, y muchas veces tenemos que adaptar nuestro estilo a las prácticas de nuestros clientes. Esto incluye crear ramas de larga duración junto con pull requests que se deben revisar y aprobar manualmente antes de ser fusionados en la rama principal. En estas situaciones, utilizamos la nueva funcionalidad GitHub merge queue que nos permite encolar pull requests e integrarlos en una rama especial en el orden en que se recibieron. También tenemos la opción de automatizar nuestros propios merge checks para evitar commits incompatibles. Básicamente simula trunk-based development (aunque las PR no se hayan fusionado aún en la rama principal del código) y permite al equipo probar sus cambios junto con su contexto sin tener que esperar a que se apruebe ese pull request. Con GitHub merge queue, obtienes los beneficios del trunk-based development incluso cuando no puedes practicarlo por completo.

71. Google Bard

Evaluar

Google Bard es un chatbot de IA generativa desarrollado por Google AI. Similar a ChatGPT, es conversacional y capaz de comunicarse y generar texto similar a como lo haría un humano en respuesta a una amplia variedad de directrices y preguntas. Bard funciona con el Modelo de Lenguaje Pathways (PaLM 2), de Google, que es un amplio modelo de lenguaje (LLM) entrenado en un conjunto de datos masivo de texto y de código. Bard es capaz de generar texto, traducir idiomas, escribir diferentes tipos de contenido creativo y responder preguntas de manera informativa.

Bard también puede ser utilizado como herramienta de guía para el desarrollo de software. A veces, a nuestros desarrolladores les ha resultado útil obtener algunas sugerencias de código o buenas prácticas y configuraciones de infraestructura para diferentes escenarios. También experimentamos con Bard para las traducciones de idiomas del Radar y obtuvimos resultados decentes para crear el primer borrador del texto. Aunque la herramienta aún está en desarrollo, por lo que puede ser un poco más lenta aún en comparación con ChatGPT, animamos a los desarrolladores a explorar la herramienta y evaluar sus potenciales beneficios.

72. Google Cloud Workstations

Evaluar

Google Cloud Workstations es la oferta de entorno de desarrollo en la nube (CDE por sus siglas en inglés) de GCP. Ofrece entornos de desarrollo en contenedores totalmente gestionados a los que se puede acceder a través de SSH, HTTPS, VS Code y Jetbrains, entre otros, brindando a los desarrolladores la ilusión de conectarse a un entorno local. Google Cloud Workstations permite a los administradores hacer que los entornos de desarrollo en contenedores formen parte de una

red privada y que sean accesibles de forma pública o privada. Esta capacidad de modificar las configuraciones de red, junto con el soporte para crear entornos con imágenes personalizadas o predefinidas, hace que, en nuestra opinión, valga la pena evaluar Google Cloud Workstations para las organizaciones que buscan una solución CDE segura dentro de su propio entorno de GCP. Si está considerando Google Cloud Workstations, le recomendamos que pruebe su configuración de red antes de implementarla ampliamente, ya que la alta latencia puede convertirse en una fricción real para la experiencia del desarrollador de estos contenedores.

73. Gradio

Evaluar

Gradio es una biblioteca de código abierto de Python que permite la creación rápida y sencilla de interfaces interactivas basadas en web para modelos de aprendizaje automático. Una interfaz gráfica de usuario sobre modelos de aprendizaje automático permite a audiencias no técnicas una mejor comprensión de las entradas, restricciones y resultados. Gradio admite muchos tipos de entradas y salidas — desde texto e imágenes hasta voz — y se ha convertido en una herramienta de referencia para la creación rápida de prototipos y la evaluación de modelos. Gradio te permite alojar fácilmente tus demos en Hugging Face o ejecutarlas localmente y permitir que otros accedan a tu demo de forma remota mediante una URLXXXXX.gradio.app. Por ejemplo, el famoso experimento DALL-E mini utiliza Gradio y está alojado en Hugging Face Spaces. Nuestros equipos han estado utilizando con éxito esta biblioteca para experimentación y prototipado, por eso la incluimos en Evaluar.

74. KWOK

Evaluar

KWOK (Kubernetes WithOut Kubelet) es una herramienta para simular el ciclo de vida de nodos y cápsulas (pods) falsos para probar el plano de control (control plane) de un clúster de Kubernetes Es difícil realizar una prueba de estrés en controladores y operadores personalizados de Kubernetes sin tener un clúster considerablemente grande. Sin embargo, con KWOK puedes establecer de forma sencilla un clúster con miles de nodos en una portátil sin que este consuma altos recursos de procesador o de memoria. Esta simulación permite distintas configuraciones de tipos de nodos y de cápsulas para probar una variedad de escenarios y casos extremos. Si necesitas un clúster real para probar tus operadores y Custom Resource Definitions (CRDs), recomendamos kind o k3s; pero si solo necesitas simular un alto número de nodos falsos, entonces te animamos a considerar KWOK.

75. Llama 2

Evaluar

Llama 2, de Meta, es un potente modelo de lenguaje gratuito tanto para investigación como para uso comercial. Está disponible como modelo pre entrenado en bruto y, perfeccionado, como Llama-2-chat para conversación y Code Llama para completar código. Dado que está disponible en varios tamaños — 7B, 13B, y 70B — Llama 2 es una buena opción para un LLM auto-gestionado, si quiere controlar sus datos.

Meta describe Llama 2 como open source, una afirmación que ha suscitado algunas críticas. La licencia y la política de uso aceptable de Meta imponen restricciones al uso comercial para algunos usuarios y también restringen el uso del modelo y el software para determinados fines. Los datos de entrenamiento de Llama 2 no son abiertos, lo que puede dificultar la comprensión y modificación del modelo. Dicho esto, se agradece la disponibilidad de un modelo potente y capaz, al menos de forma semiabierta.

76. Maestro

Evaluar

Maestro es una nueva herramienta multi-plataforma de automatización de test UI con tolerancia integrada a la inestabilidad y variancia en tiempos de carga en aplicación debido a la conexión y a factores externos. Con una sintaxis YAML declarativa, hace más fácil escribir y mantener los test automatizados para aplicaciones móviles. Soporta aplicaciones navitas iOS y Android, aplicaciones React Native y Flutter así como una variedad de herramientas para automatizar interacciones complejas de UI mobile, como tapping, scrolling o swipping. Maestro se distribuye como un único binario para un fácil uso, corre en modo interpretado y facilita autorizar tests nuevos gracias a herramientas como el modo continuo. A Maestro todavía le faltan herramientas específicas como soporte para dispositivos iOS, pero la herramienta está evolucionando rápidamente.

77. LLM de código abierto para codificación

Evaluar

GitHub Copilot es una herramienta muy útil para ayudar con la codificación durante el desarrollo de software. Debajo del capó, los LLMs pueden potenciar experiencias perfectas para los desarrolladores a través de asistencia de código en línea, refinamiento de código, soporte conversacional en el IDE y mucho más. La mayoría de estos modelos son propietarios y sólo se pueden utilizar mediante servicios de suscripción. La buena noticia es que puedes utilizar varios LLM de código abierto para codificar. Si se encuentra en un entorno donde necesita crear su propio servicio de asistencia a la codificación (como en una industria altamente regulada), considere modelos como StarCoder y WizardCoder. StarCoder se entrena con un gran conjunto de datos mantenido por BigCode, y Wizardcoder es un modelo StarCoder refinado con Evol-Instruct.

Hemos usado StarCoder en nuestros experimentos y lo encontramos útil para generar elementos estructurados de ingeniería de software como código, YAML, SQL y JSON. Según nuestros experimentos, encontramos que ambos modelos son receptivos al aprendizaje en contexto usando ejemplos de few-shot en el mensaje. No obstante, para tareas específicas (como la generación de SQL para una base de datos específica como Postgres), los modelos necesitaban ajustes. Recientemente, Meta dio a conocer su Code Llama, una versión especializada en código de Llama 2. Asegúrese de proceder con precaución al utilizar estos modelos de código abierto. Considere su licencia, la licencia del código y del conjunto de datos utilizados para entrenar el modelo. Evalúe cuidadosamente estos aspectos antes de elegir cualquiera de estos LLM de codificación para su organización.

78. OpenCost

Evaluar

OpenCost es un proyecto de código abierto para monitorizar los costes de infraestructura desglosados con granularidad a nivel de objetos Kubernetes (pods, contenedores, clústers, etc.), incluyendo diversos recursos dentro del clúster (CPU, GPU, RAM, almacenamiento, redes). Obtiene los datos de facturación gracias a las integraciones con múltiples APIs de proveedores cloud pudiendo ser también configurado para clústeres de Kubernetes en local. OpenCost es el motor de asignación de costes originalmente construido y aún utilizado por Kubecost, pero también se puede usar de manera independiente. Los datos de asignación de costes se pueden exportar en archivos CSV o Prometheus para un análisis y visualización adicionales. Nuestros equipos están observando de cerca el desarrollo de herramientas como OpenCost y Kubecost para que los equipos de producto y plataforma tengan acceso a los costes en organizaciones que han adoptado Kubernetes. En estas

etapas iniciales, encuentran que OpenCost aún no funciona bien con ciertas cargas de trabajo como instancias puntuales de corta duración que se utilizan a menudo en pipeline de datos.

79. OpenRewrite

Evaluar

Hemos visto varios casos de uso para herramientas de inteligencia de código: migrar a una nueva versión de API de una librería ampliamente utilizada, comprender el impacto en una empresa de una vulnerabilidad recién descubierta en dicha librería o aplicar actualizaciones a muchos servicios creados a partir de la misma plantilla. Sourcegraph sigue siendo una herramienta popular en este espacio, y OpenRewrite es otra herramienta que queremos destacar. Aunque nuestros equipos la han utilizado sobre todo en Java para problemas limitados, como la actualización de servicios creados a través de un kit de inicio, sigue ampliando su cobertura de lenguajes y casos de uso. Nos gusta que venga con un catálogo de recetas incluido, que describen los cambios que se deben realizar, por ejemplo, para migrar versiones de frameworks comúnmente utilizados. El motor de refactorización, las recetas incluidas y los plugins de herramientas de construcción son software de código abierto, lo que facilita que los equipos usen OpenRewrite solo cuando lo necesiten. Queda por ver cómo el espacio en proceso de maduración de las herramientas de inteligencia de código, que se basan en analizar el código fuente y representarlo con un árbol de sintaxis abstracta (AST, por sus siglas en inglés), se verá afectado por los rápidos desarrollos en el espacio de los LLMs.

80. OrbStack

Evaluar

OrbStack es una forma de ejecutar contenedores de Docker en macOS; a nuestras desarrolladoras les ha parecido más ligera, rápida y fácil de configurar y usar que Docker Desktop y Colima. La herramienta está aún en desarrollo y por ello tiene menos funcionalidades, pero ya muestra un gran potencial con su simplicidad y velocidad. También puedes usar OrbStack para crear y administrar MVs de Linux en macOS.

81. Pixie

Evaluar

Pixie es una herramienta de observabilidad para aplicaciones nativas de Kubernetes Toma un enfoque interesante hacia la observabilidad al aprovechar eBPF para recolectar automáticamente datos de telemetría de múltiples fuentes de datos. Los datos de telemetría recolectados son guardados localmente en cada nodo y procesados de forma centralizada a través de su API de nivel de control. En general, dentro del ecosistema de Kubernetes, consideramos que merece la pena evaluar Pixie como herramienta de observabilidad.

82. Tabnine

Evaluar

Tabnine es un contendiente en el agitado espacio de los asistentes de codificación. Proporciona sugerencias en línea de código y un chat directamente en el IDE. Similar a GitHub Copilot, Tabnine existe desde antes del actual auge de este tipo de herramientas y, por lo tanto, es uno de los productos más maduros de este tipo. A diferencia de Copilot, utiliza un modelo que solo se entrena con código con licencia autorizado y ofrece una versión de hospedaje propio para organizaciones que se preocupan por enviar sus fragmentos de código a un servicio de terceros. Tabnine está disponible como versión gratuita limitada así como en versión de pago que tiene sugerencias más comprensibles y también ofrece un modo con un modelo local (aunque menos potente) que puedes usar sin conexión a Internet.

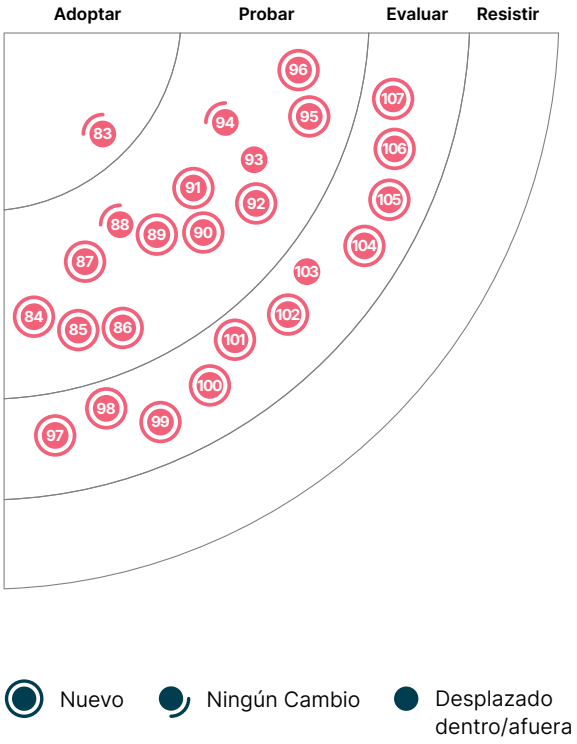
Lenguajes & Frameworks

Adoptar
83. Playwright

Probar
84. API Minimalista .NET
85. Ajv
86. Armeria
87. AWS SAM
88. Dart
89. fast-check
90. Kotlin con Spring
91. Mockery
92. Netflix DGS
93. OpenTelemetry
94. Polars
95. Pushpin
96. Snowpark

Evaluar
97. Perfiles de Referencia
98. GGML
99. GPTCache
100. API Inflexión Gramatical
101. htmx
102. Kotlin Kover
103. LangChain
104. LlamaIndex
105. promptfoo
106. Semantic Kernel
107. Spring Modulith

Resistir
—



Nuevo Ningún Cambio Desplazado dentro/afuera

83. Playwright

Adoptar

Con Playwright puedes escribir pruebas de inicio a fin que se ejecutan en Chrome, Firefox y WebKit. Utilizando el protocolo de herramientas de desarrollo de Chrome (CDP por sus siglas en inglés) Playwright ofrece nuevas funcionalidades y elimina muchos de los problemas vistos con WebDriver. Los navegadores basados en Chromium implementan CDP directamente. Sin embargo, para ser compatible con Firefox y Webkit, el equipo de Playwright tiene que enviar parches a estos navegadores, lo que a veces puede limitar el marco de trabajo.

Entre las muchas funcionalidades de Playwright se incluyen: Esperas automáticas integradas, que resultan en pruebas más confiables y fáciles de entender; contexto del navegador, que te permite probar que la persistencia de la sesión a través de las pestañas funciona correctamente; y la capacidad de simular notificaciones, geolocalización y configuraciones de modo oscuro. Nuestros equipos están impresionados con la estabilidad que proporciona a los paquetes de prueba y les gusta recibir feedback más rápidamente ejecutando las pruebas en paralelo. Otras características que distinguen a Playwright son una mejor compatibilidad con la carga diferida y el rastreo. Aunque Playwright tiene algunas limitaciones — por ejemplo, actualmente la compatibilidad con componentes es experimental — nuestros equipos lo consideran el marco de referencia de pruebas y, en algunos casos, están migrando de Cypress y Puppeteer.

84. API Minimalista .NET

Probar

ASP.NET core MVC ha demostrado ser un enfoque potente y flexible para construir aplicaciones web que alojan APIs. Sin embargo, su flexibilidad conlleva un cierto nivel de complejidad, que incluye boilerplate y convenciones que no siempre son evidentes. El enrutamiento proporcionado por ASP.NET permite alojar múltiples servicios en una única aplicación, pero en el mundo actual de funciones serverless y microservicios que pueden ser desplegados de manera independiente, esta flexibilidad podría ser excesiva. .NET Minimal APIs ofrece un enfoque sencillo al implementar una aplicación web con una single-API en el ecosistema .NET. El framework Minimal API puede implementar un endpoint API con solo unas pocas líneas de código. Minimal API se une a la nueva generación de frameworks de API — incluyendo Micronaut, Quarkus y Helidon — que están optimizados para despliegues ligeros y tiempos de inicio rápidos. Nos interesa la combinación de Minimal APIs y .NET 7 Native AOT para implementar microservicios simples y livianos en funciones serverless.

85. Ajv

Probar

Ajv es una popular biblioteca de JavaScript utilizada para validar un objeto de datos usando una estructura definida con un esquema JSON. Ajv es rápida y flexible para validar tipos de datos complejos. Soporta una amplia gama de funciones de esquema, incluyendo palabras clave y formatos personalizados. Es utilizada por muchas aplicaciones y bibliotecas JavaScript de software abierto. Nuestros equipos han usado Ajv para implementar pruebas de contrato guiadas por el consumidor en flujos de CI y, junto con otras herramientas para generar datos simulados con el esquema JSON, es muy potente. En el mundo TypeScript, Zod es una alternativa popular que tiene una API declarativa para definir el esquema y validar los datos.

86. Armeria

Probar

Armeria es un framework de código abierto para crear microservicios. Nuestros equipos lo han utilizado para crear APIs asincrónas y nos gusta bastante el enfoque para abordar aspectos transversales, como el rastreo distribuído o los rompedores de circuito, a través de service decorators. El framework incluye la reutilización de puertos para el tráfico gRPC y REST, entre otras opciones ingeniosas de diseño. Con Armeria, podemos agregar nuevas características en gRPC de manera incremental sobre una base de código existente en REST o viceversa. En general, consideramos que Armeria es un framework flexible para microservicios con varias integraciones fuera de la caja.

87. AWS SAM

Probar

AWS Serverless Application Model (SAM) es una herramienta de código abierto para crear aplicaciones serverless usando servicios de AWS. Ya en ediciones previas, resaltamos Serverless Framework como una herramienta popular para desplegar servicios serverless en distintos proveedores en la nube, principalmente servicios basados en AWS Lambda. Recientemente, AWS SAM ha ganado popularidad ya que la herramienta ha evolucionado mucho desde sus primeras versiones. Los equipos en Thoughtworks lo encuentran fácil de configurar y también lo usan para escribir tests y depurar servicios basados en el servicio AWS Lambda, ya que permite ejecuciones locales de Lambdas para desarrollo.

88. Dart

Probar

Dart es un lenguaje de programación desarrollado por Google que soporta la creación de aplicaciones multiplataforma, incluyendo navegadores web, WebAssembly, aplicaciones de escritorio y móviles. Su adopción ha sido impulsada por el predominio de Flutter — un popular conjunto de herramientas de Interfaz de Usuario (UI) multi-plataforma impulsado por Dart — en el espacio de frameworks de aplicaciones móviles nativas multiplataforma. En respuesta al feedback de la comunidad, Dart ha evolucionado desde sus versiones iniciales y ha añadido la integración sólida denull safety en la versión tres, además de un sistema robusto de tipado. Por último, el ecosistema de Dart está creciendo rápidamente, con una comunidad vibrante y un amplio rango de librerías y herramientas disponibles, haciéndolo atractivo para desarrolladores.

89. fast-check

Probar

fast-check es una herramienta de pruebas basada en propiedades para JavaScript y TypeScript, capaz de generar automáticamente datos de prueba de forma que se pueda explorar una amplia gama de datos de entrada sin tener que crear pruebas separadas. Esto facilita el descubrimiento de escenarios extremos. Nuestros equipos están obteniendo buenos resultados con el uso de fast-check en las pruebas de back-end gracias a su buena documentación, facilidad de uso y perfecta integración con los frameworks de pruebas existentes, lo que mejora la eficiencia de las pruebas unitarias.

90. Kotlin con Spring

Probar

Hace cinco años movimos Kotlin al anillo de adopción y hoy muchos de nuestros equipos confirman que Kotlin no sólo es su opción por defecto en la JVM, sino que ha desplazado a Java casi por

completo en el software que escriben. Al mismo tiempo, la sed por microservicios parece estar desapareciendo: hemos observado que la gente empieza a explorar arquitecturas con unidades desplegables más grandes, utilizando frameworks como Spring Modulith entre otros. Conocemos buenos frameworks nativos en Kotlin y hemos mencionado algunos de ellos anteriormente; sin embargo, en algunos casos, la madurez y la riqueza de características del framework Spring es una gran ventaja, y hemos estado utilizando Kotlin con Spring con éxito, normalmente sin problemas o con problemas menores.

91. Mockery

Probar

Mockery es una librería madura de Golang que ayuda a generar implementaciones mock de interfaces simulando comportamientos de dependencias externas. Con métodos de tipado seguro para generar las expectativas de llamadas y flexibilidad para mockear valores de retorno, permite que los tests se centren en la lógica de negocio en lugar de preocuparse por la corrección de dependencias externas. Mockery usa generadores Go y simplifica la gestión de mocks en la suite de pruebas.

92. Netflix DGS

Probar

Hemos usado mayormente GraphQL para agregación de recursos de lado del servidor y hemos implementado el lado del servidor usando una variedad de tecnologías. Para servicios escritos con Spring Boot, nuestros equipos han tenido buenas experiencias con Netflix DGS. Está construido encima de graphql-java y provee funcionalidades y abstracciones en el modelo de programación Spring Boot que facilita el implementar endpoints en GraphQL e integrarlos con características de Spring como Spring Security. A pesar de estar escrito en Kotlin, DGS funciona igualmente bien con Java.

93. OpenTelemetry

Probar

OpenTelemetry Hemos estado utilizando OpenTelemetry como solución durante un tiempo y la hemos recomendado en ediciones anteriores. Su capacidad para capturar, instrumentar y gestionar de manera fluida datos de telemetría entre distintos servicios y aplicaciones ha mejorado nuestra tecnología de observabilidad. La flexibilidad y compatibilidad de OpenTelemetry en distintos entornos la ha convertido en un instrumento muy valioso de nuestra caja de herramientas. Ahora sentimos especial curiosidad por el reciente lanzamiento de la especificación del OpenTelemetry Protocol (OTLP) que incluye gRCP y HTTP. Este protocolo estandariza el formato de transmisión de datos de telemetría, que promueve la interoperabilidad y simplifica la integración con otras herramientas de monitorización y análisis. A medida que seguimos explorando el potencial de integración del protocolo, estamos evaluando su impacto a largo plazo en nuestras estrategias de monitoreo y observabilidad y en el panorama general de monitorización.

94. Polars

Probar

Polars es una librería de DataFrame en memoria, implementada en Rust. A diferencia de otras DataFrames (como pandas), Polars es multiproceso, permite la ejecución perezosa y es segura para operaciones paralelas. Los datos en memoria se organizan en formato Apache Arrow para conseguir operaciones analíticas eficientes y para permitir la interoperabilidad con otras herramientas. Si estás familiarizado con pandas, puedes rápidamente empezar con los bindings Python de Polars. Creemos que Polars, con las implementaciones de Rust y los bindings de Python, es una estructura de datos en memoria de alto rendimiento que te ayudará para tus necesidades analíticas. Nuestros equipos siguen teniendo una buena experiencia con Polars, razón por lo cual lo estamos moviendo a Probar.

95. Pushpin

Probar

Pushpin es un proxy inverso que actúa como intermediario entre los clientes y los servidores backend, administrando las conexiones de larga duración como WebSockets y Server-Sent Events. Proporciona una manera de terminar las conexiones de larga duración de los clientes, permitiendo que otros sistemas se abstengan de esa complejidad. Gestiona eficazmente un gran número de conexiones persistentes y las distribuye automáticamente entre múltiples servidores backend, optimizando el rendimiento y la fiabilidad. Nuestra experiencia usándolo para comunicaciones en tiempo real con dispositivos móviles mediante WebSockets ha sido positiva, y hemos sido capaces de escalar horizontalmente para millones de dispositivos.

96. Snowpark

Probar

Snowpark es una librería para consultar y procesar datos a gran escala en Snowflake. Nuestros equipos lo usan para escribir código manejable para interactuar con datos que residen en Snowflake — semejante a escribir código Spark pero para Snowflake. En última instancia, es un motor que traduce código a SQL que Snowflake comprende. Se pueden construir aplicaciones que procesan datos en Snowflake sin mover datos hacia el sistema donde se ejecuta el código de la aplicación. Una desventaja: el soporte de test unitarios no es óptimo; nuestros equipos lo compensan escribiendo otro tipo de tests.

97. Perfiles de Referencia

Evaluar

Perfiles de Referencia — no confundir con perfiles de referencia de Android — son perfiles de Android Runtime que guían la compilación previa. Se crean una vez por versión en una máquina de desarrollo y se envían con la aplicación, lo que los hace disponibles más rápidamente que depender de Perfiles en la Nube, una tecnología más antigua. El runtime utiliza el Perfil de Referencia en una aplicación o biblioteca para optimizar rutas de código importantes, lo que mejora la experiencia para usuarios nuevos y existentes cuando descargan o actualizan la aplicación. Crear Perfiles de Referencia es relativamente sencillo y, según su documentación, puede llevar a mejoras significativas (de hasta un 30%) en el rendimiento.

98. GGML

Evaluar

GGML es una librería de aprendizaje automático en C que permite la inferencia de CPU. Esta librería define un formato binario para distribuir modelos grandes de lenguaje (LLMs, por sus siglas en inglés). Para hacerlo, usa cuantificación digital, una técnica que permite que los LLMs ejecuten inferencia de CPU efectiva en hardware de consumo. GGML soporta varias estrategias de cuantificación digital (e.g., cuantificación de 4 bits, 5 bits, y 8 bits), cada una de las cuales ofrece diferentes relaciones coste-beneficio entre eficiencia y rendimiento. Una manera rápida de probar, ejecutar y construir aplicaciones con estos modelos de cuantificación, es un binding de Python llamado C Transformers. Se trata de un wrapper de Python sobre GGML que nos abstrae del repetitivo código necesario para ejecutar inferencia al proveer una API de alto nivel. Hemos usado estas librerías para construir pruebas de concepto y experimentos. Si estás valorando usar LLMs auto alojados, evalúe cuidadosamente estas librerías para su organización.

99. GPTCache

Evaluar

GPTCache es una librería de caché semántica para modelos de lenguajes grandes(LLMs). Vemos la necesidad de una capa de caché delante de los LLM por dos razones principales: mejorar el rendimiento general reduciendo llamadas a API externas y reducir el costo de operación almacenando en caché respuestas similares. A diferencia de los enfoques tradicionales cacheo que buscan coincidencias exactas, lo que las soluciones de cacheo basadas en LLM requieren son coincidencias similares o relacionadas para las consultas de entrada. GPTCache aborda esto con la ayuda de algoritmos de embedding para convertir los inputs en embeddings y luego usar un almacén de datos vectorial para la búsqueda de similitudes en dichos embeddings. Un inconveniente de este diseño es que podrías encontrar falsos positivos durante los aciertos de caché o falsos negativos durante los fallos de caché, por lo que te recomendamos que evalúes cuidadosamente GPTCache para tus aplicaciones basadas en LLM.

100. API Inflexión Gramatical

Evaluar

En muchos idiomas, el género es más visible que en inglés y las palabras cambian según el género. Por ejemplo, al dirigirse a las personas usuarias a menudo es necesario cambiar la forma gramatical por género (inflexionar las palabras) y suele ser común utilizar el género masculino de forma predeterminada. Existe evidencia que esta práctica tiene un impacto negativo en el desempeño y actitud de las personas — y por supuesto, es inapropiada. Utilizar alternativas con género neutro a menudo pueden parecer incómodas. Dirigirse al usuario de manera correcta, es entonces la opción preferida, y con la introducción en Android 14 de Grammatical Inflection API, las personas desarrolladoras de Android, tienen una forma más fácil de hacerlo.

101. htmx

Evaluar

htmx es una biblioteca de interfaz de usuario HTML pequeña y ordenada que recientemente se hizo popular aparentemente de la nada. Durante nuestra discusión del Radar, descubrimos que su predecesor intercooler.js existió hace ya diez años. A diferencia de otros marcos de trabajo precompilados y complejos de JavaScript/TypeScript, htmx fomenta el uso directo de atributos HTML para acceder a operaciones como AJAX, transiciones CSS, WebSockets y eventos enviados por el servidor. No hay nada técnicamente sofisticado en htmx, pero su popularidad recuerda la simplicidad del hipertexto en los primeros días de la web. El sitio web del proyecto también incluye algunos

ensayos reveladores (y divertidos) sobre hipermedia y desarrollo web, lo que sugiere que el equipo detrás de htmx ha pensado cuidadosamente sobre su propósito y filosofía.

102. Kotlin Kover

Evaluar

Kotlin Kover es un conjunto de herramientas de cobertura de código diseñado específicamente para Kotlin, que soporta Kotlin JVM, Multiplatform y proyectos Android. La cobertura de código es importante porque identifica fragmentos no testeados de código, lo que refuerza la fiabilidad del software. A medida que Kover va evolucionando, se destaca su capacidad para generar informes completos en HTML y XML, junto con una precisión inigualable diseñada para Kotlin. Para equipos que trabajan únicamente con Kotlin, les recomendamos que evalúen Kover para aprovechar su potencial para mejorar la calidad del código.

103. LangChain

Evaluar

LangChain ies un framework para construir aplicaciones con modelos de lenguaje de gran tamaño (LLMs). Para construir este tipo de productos en forma práctica, se deben combinar con datos específicos de usuario o dominio que no han formado parte de un entrenamiento anterior. LangChain llena este vacío con características como gestión de prompts, encadenamiento, agentes y cargadores de documentos. El beneficio de contar con plantillas prompt y cargadores de documentos es que pueden acelerar el tiempo de puesta en marcha. Aunque es una opción popular implementar aplicaciones Retrieval-Augmented Generation y el patrón ReAct prompting LangChain ha sido criticado por ser difícil de usar y demasiado complicado. Al elegir un stack tecnológico para tu aplicación LLM, podrías buscar alternativas similares como Semantic Kernel en este espacio de desarrollo rápido.

104. LlamaIndex

Evaluar

LlamaIndex es un framework de datos diseñado para facilitar la integración de datos privados o de específicos de dominio con modelos grandes de lenguaje (LLMs). Ofrece herramientas para consumir datos de diversas fuentes — incluyendo APIs, bases de datos y PDFs — y estructura estos datos en un formato que los LLMs pueden consumir fácilmente. Al usar varios tipos demotores, LlamaIndex habilita interacciones en lenguaje natural con estos datos estructurados, haciéndolo accesible para aplicaciones que van desde la búsqueda basada en consultas hasta interfaces conversacionales. De manera similar a LangChain, el objetivo de LlamaIndex es acelerar el desarrollo con LLMs, pero toma un enfoque más parecido a un framework de datos.

105. promptfoo

Evaluar

promptfoo permite la ingeniería de instrucciones. basada en pruebas. Cuando integramos Modelos de Lenguaje de Aprendizaje Profundo (LLMs, por sus siglas en inglés) en aplicaciones, ajustar las instrucciones para producir resultados óptimos y consistentes pueden llevar mucho tiempo. Puedes utilizar promptfoo tanto como una interfaz de línea de comandos (CLI) como una biblioteca para probar sistemáticamente instrucciones mediante casos de prueba predefinidos. El caso de prueba, junto con las afirmaciones, se pueden configurar en un archivo de configuración YAML sencillo. Esta configuración incluye las instrucciones que se están probando, el proveedor del modelo, las afirmaciones y los valores de variables que se sustituirán en las instrucciones. promptfoo admite muchas afirmaciones, como verificar la igualdad, la estructura JSON, la similitud, funciones

personalizadas e incluso usar un LLM para evaluar las salidas del modelo. Si estás buscando automatizar la retroalimentación sobre la calidad de las instrucciones y el modelo, definitivamente deberías considerar promptfoo.

106. Semantic Kernel

Evaluar

Semantic Kernel es la versión de código abierto de uno de los componentes principales en la suite de productos Copilot de Microsoft. Es una biblioteca Python que ayuda a construir aplicaciones sobre un LLM, similar a LangChain. El concepto principal de Semantic Kernel es su planificador, que nos permite construir agentes autónomos basados en LLMs los cuales crean un plan para un usuario, y luego lo ejecutan paso a paso con ayuda de plugins.

107. Spring Modulith

Evaluar

Aunque fuimos promotores iniciales de los microservicios y hemos visto el patrón utilizado con éxito en una gran variedad de sistemas, también hemos observado la aplicación incorrecta y el abuso de los microservicios, a menudo como resultado de la Envidia de microservicios. En lugar de comenzar un nuevo sistema con una colección de procesos desplegados por separado, a menudo es aconsejable empezar con un monolito bien estructurado y descomponerlo en unidades desplegables de manera independiente cuando la aplicación llega al nivel en que los beneficios de los microservicios superen la complejidad adicional inherente en los sistemas distribuidos. Recientemente hemos visto un resurgir del interés en esta aproximación y una definición más detallada de qué, exactamente, constituye un monolito bien estructurado. Spring Modulith es un framework que te ayuda a estructurar tu código de forma que sea más fácil descomponerlo en microservicios cuando llegue el momento adecuado. Proporciona una manera de modularizar tu código de modo que los conceptos lógicos de dominios y contextos acotados (bounded context) se alineen con los conceptos físicos de archivos y estructura de paquetes. Esta alineación facilita la refactorización del monolito cuando sea necesario, así como testear los dominios de manera aislada. Spring Modulith proporciona un mecanismo de eventos dentro del proceso que ayuda a desacoplar aún más los módulos dentro de una única aplicación. Lo mejor de todo es que se integra con ArchUnit y jmolecules para automatizar la verificación de sus reglas de diseño orientadas a dominio.

Manténte al día de todas las noticias y opiniones relacionadas con el Radar

Suscríbete al Radar Tecnológico para recibir por correo electrónico cada dos meses las opiniones tecnológicas de Thoughtworks y futuras publicaciones del Radar Tecnológico.

Suscríbete ahora

Thoughtworks es una consultora global de tecnología que integra estrategia, diseño e ingeniería para impulsar la innovación digital. Somos 11,500+ personas en 51 oficinas en 18 países. En los últimos 30 años, hemos logrado un impacto extraordinario junto con nuestros clientes, ayudándoles a resolver problemas complejos de negocio a través de la tecnología como elemento diferenciador.

/thoughtworks

Strategy. Design. Engineering.