Radar Tecnológico

Una guía con opiniones sobre las tecnologías de vanguardia

/thoughtworks

Volumen 28

Sobre el Radar

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

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

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

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

Un vistazo al Radar

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

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

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

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

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

Resistir: Proceder con precaución.

Nuevo **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.

Equipo de traducción: Alejandra Quintana, Alejandro Garcia Luque, Alejandro Hidalgo, Alex Javier Ulloa, Amanda Hernando Bernabe, Ana Corral, Ana Vivanco, Anastasia Potapova Filina, Andrea Obando, Angie Bracho, Araceli Correa, Bárbara Deluchi, Camila Vigneaux, Carolina Melgarejo, Catalina Margozzini, Catalina Solís, Caterina Ojeda, Cesar Abreu, Claudia Lertora, Cristian Montero, Cristóbal Bórquez, Dani Gomez, Daniel Santibañez, Diana Pila, Diana Suasnavas, Eduard Maura i Puig, Enrique Aracil Mas, Erika Vacacela, Esteban Moreno Arias, Eumir Ollarvez, Felipe Jorquera, Fernando Parra, Fran Ruiz-Tagle, Francisca Castro Barriga, Gabriel Bravo, Gabriel Frías Rivas, Gabriel Villacis, Gabriela Marques, Gisela Yumi, Gustavo Chiriboga, Irene Torres, Isabel Hong, Jacob Ibáñez Sánchez, Jesús Cardenal, Jhosep Marin, Joan Galvan, Jorge Agudo Praena, Jorge Palacios, Jorgina Arres, José Albarado , Jose María Alonso Montero, Juan Romero Gómez, Katherine Ayala, Laura Mirás, Lorena Campos, Lucía Parga Basanta, Luis Maracara, Manuel Chamber, Manuel Medina, María Córdova, Maria Fernanda Yepez, Maria Montoza Gonzalez, Milber Champutiz, Nahuel Delgado, Naileth Rivero, Nati Rivera, Nicol Rafalowski, Óscar Rojas, Paula Nieto, Pedro Carbonell, Sebastian Roman, Tatiana, Elizabeth Andrade Ávila, Virginia de la Fuente, Yago Pereiro y Yeraldine Mendoza
Equipo de Edición: Juan Infante Zumer, Fausto de la Torre, Maya Ormaza y Fernando Tamayo
Equipo de Marketing: Elizabeth Parra, María José Lalama, Magdalena Grondona y Daniel Negrete

Contribuyentes

El Technology Advisory Board (TAB) es un grupo de 21 personas tecnólogas senior de Thoughtworks. El TAB se reúne dos veces al año en persona y virtualmente cada dos semanas. Su función principal es ser un grupo de asesoramiento para el CTO de Thoughtworks, Rebecca Parsons.

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

Rebecca Parsons (CTO)

Martin Fowler (Chief Scientist)

Bharani Subramaniam

Birgitta Böckeler

Brandon Byars

Camilla Falconi Crispim

Erik Dörnenburg

Fausto de la Torre

Hao Xu

Ian Cartwright

James Lewis

Marisa Hoenig

Maya Ormaza

Mike Mason

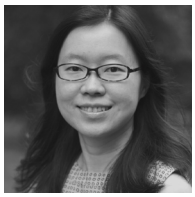
Neal Ford

Pawan Shah

Scott Shaw

Selvakumar Natesan

Shangqi Liu



Sofia Tania

Vanya Seth

Temas

El meteórico ascenso de la IA práctica

No, este texto no fue escrito por ChatGPT. La inteligencia artificial ha estado burbujeando silenciosamente en áreas especializadas durante décadas, y herramientas como GitHub Copilot han existido (y se han adoptado gradualmente) durante algunos años. Sin embargo, en los últimos meses, herramientas como ChatGPT han reorientado completamente a todos hacia lo que es posible y han hecho que las herramientas estén ampliamente disponibles. Varios puntos en esta edición de Radar abordan usos prácticos de la IA para proyectos más allá de sugerir código que requiere ajustes: Desarrollo basado en pruebas asistidas por IA, usando IA para ayudar a construir modelos de análisis, y muchos más. De manera similar a cómo las hojas de cálculo permitieron a los contadores dejar de usar máquinas sumadoras para recalcular hojas de cálculo complejas a mano, la próxima generación de IA asumirá tareas para aliviar a los trabajadores de tecnología, incluidas personas desarrolladoras, al reemplazar tareas tediosas que requieren conocimiento (pero no sabiduría).

Sin embargo, advertimos contra usos excesivos o inapropiados. En este momento, los modelos de IA son capaces de generar un buen primer borrador. Pero el contenido generado siempre necesita ser monitoreado por un humano que pueda validarlo, moderarlo y usarlo de manera responsable. Si se ignoran estas precauciones, los resultados pueden generar riesgos para la reputación y la seguridad de las organizaciones y los usuarios. Incluso algunas demostraciones de productos advierten a los usuarios: “El contenido generado por IA puede contener errores. Asegúrese de que sea preciso y apropiado antes de usarlo”.

Accesibilidad accesible

La accesibilidad ha sido un elemento importante que las organizaciones han considerado por muchos años. Recientemente, hemos destacado las experiencias de nuestros equipos con herramientas y técnicas en constante crecimiento que añaden una mejor accesibilidad al desarrollo, y en varias regiones nuestros equipos destacan el conocimiento de estas técnicas a través de campañas de concientización. Hemos mostrado blips relacionados con la accesibilidad sobre el desarrollo de pipelines de integración continua, manuales de diseño, pruebas de accesibilidad guiadas por IA, linting y pruebas unitarias. Aumentar la conciencia alrededor de este tema tan importante es bienvenida; técnicas que dan acceso a funcionalidades a más personas de mejor manera sólo puede ser algo bueno.

Arenas movedizas de funciones Lambda

Las funciones serverless — AWS Lambdas — aparecen cada vez más como herramientas de equipos de arquitectura y desarrollo, y se utilizan para una amplia variedad de tareas útiles que aprovechan los beneficios de la infraestructura alojada en la nube. Sin embargo, como muchas cosas útiles, las soluciones a veces comienzan siendo lo suficientemente simples pero luego, a partir de un éxito gradual, siguen evolucionando hasta que superan las limitaciones inherentes al paradigma y se hunden en la arena por su propio peso. Si bien vemos muchas aplicaciones exitosas de soluciones estilo serverless, también escuchamos muchas historias de cautela desde nuestros proyectos, como el antipatrón Lambda pinball. Por ejemplo, herramientas que facilitan compartir código entre Lambdas u orquestar iteraciones complejas puede resolver simples problemas comunes pero luego corren el riesgo de recrear, con nuevas piezas, algunos antipatrones de arquitectura terribles. Si necesitas una herramienta para administrar el uso compartido de código y la implementación independiente en un grupo de funciones serverless, quizás sea hora de reconsiderar la idoneidad del enfoque. Como todas las soluciones tecnológicas, "serverless" tiene aplicaciones adecuadas, pero muchas de sus características incluyen compromisos que se profundizan a medida que la solución evoluciona.

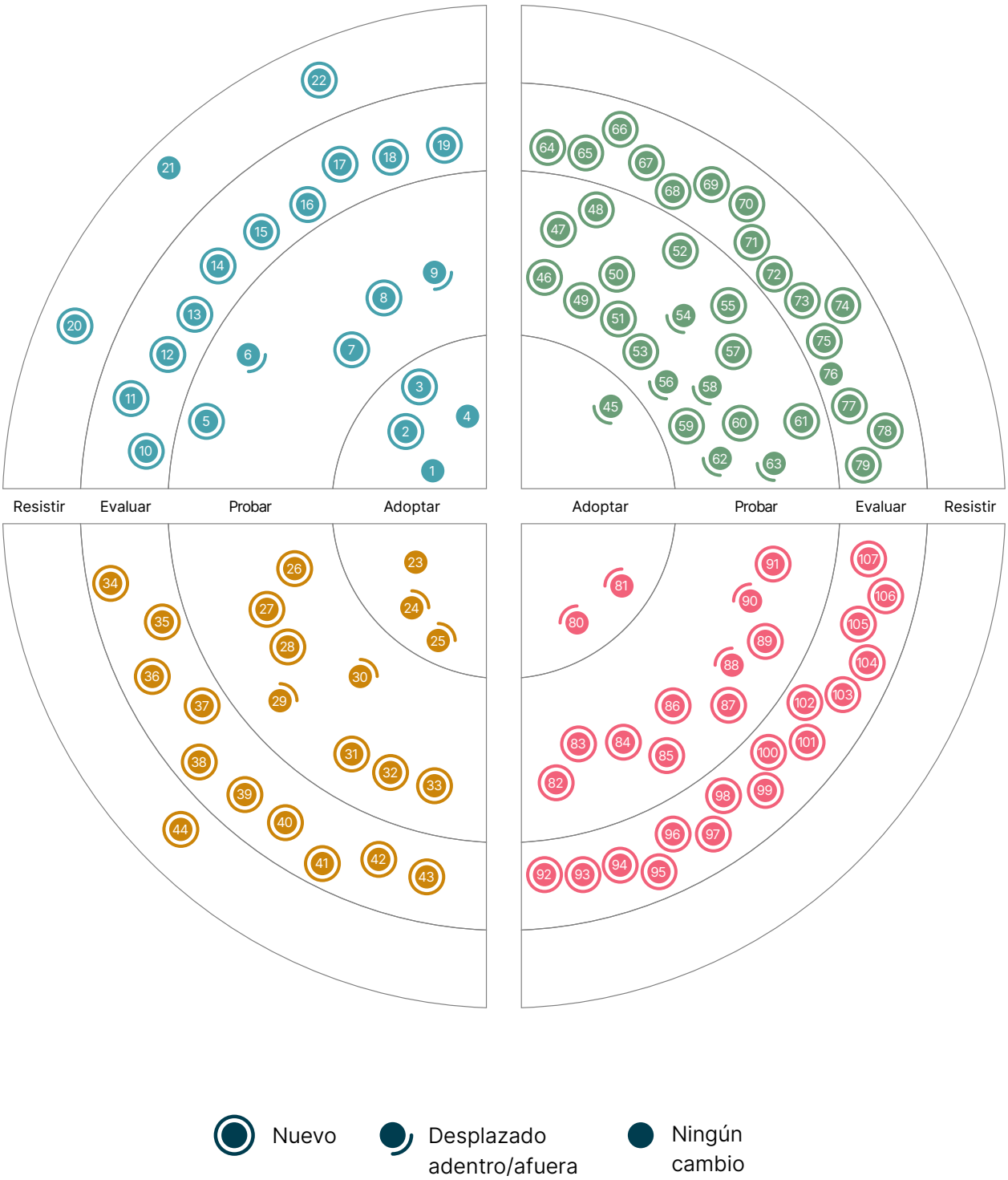
El rigor de la ingeniería se une con la analítica y la IA

Hace tiempo que consideramos que "integrar calidad en el proceso" es un aspecto esencial en el desarrollo de una analítica y modelos de machine learning confiables. Las transformaciones basadas en pruebas, las pruebas de sanidad de datos y la verificación de modelos de datos, fortalecen las pipelines de datos que impulsan los sistemas analíticos. La validación de modelos y el control de calidad son imprescindibles a la hora de afrontar los sesgos y garantizar sistemas de ML éticos con resultados justos. Si se integran estas prácticas, las empresas estarán mejor posicionadas para beneficiarse de la IA y del ML, al mismo tiempo que crearán soluciones responsables basadas en datos que sirven para un grupo diverso de usuarios. El ecosistema de herramientas sobre este tema ha seguido creciendo y madurando: por ejemplo, Soda Core, una herramienta de calidad de datos, permite la validación de datos según llegan al sistema y comprueba automáticamente que no haya anomalías. Deepchecks da pie a la intersección de la integración continua con la validación de modelos, un paso importante para incorporar buenas prácticas de ingeniería en entornos de analítica. Giskard aporta control de calidad para modelos de IA, lo cual ayuda a los diseñadores a detectar sesgos y otras facetas negativas de los modelos, alineándose así con nuestra recomendación de considerar los aspectos éticos con cuidado al desarrollar soluciones con IA. Estas herramientas son una prueba más de la popularidad de la analítica y el ML, así como de su integración con buenas prácticas de ingeniería.

¿Declarar o programar?

Una discusión aparentemente perpetua que ocurre en cada reunión de Radar ganó especial importancia esta vez — para una tarea dada, debería escribir una especificación declarativa utilizando JSON, YAML o algo específico de dominio como HCL, O debería escribir código en un lenguaje de programación de propósito general? Por ejemplo, discutimos las diferencias entre Terraform Cloud Operator versus Crossplan, ya sea para usar o no AWS CDK y utilizar Dagger para programar un pipeline de despliegue entre otros casos. Las especificaciones declarativas, aunque a menudo son más fáciles de leer y escribir, ofrecen abstracciones limitadas que conducen a un código repetitivo. Los lenguajes de programación adecuados pueden usar abstracciones para evitar la duplicación, pero estas abstracciones pueden hacer que el código sea considerablemente más difícil de seguir, especialmente cuando las abstracciones se ejecutan después de años de cambios. En nuestra experiencia, no hay una respuesta universal a la pregunta planteada anteriormente. Los equipos deben considerar ambos enfoques, y cuando una solución resulta difícil de implementar limpiamente en un tipo de lenguaje, deben reevaluar el otro tipo. Incluso puede tener sentido dividir los problemas e implementarlos en diferentes lenguajes.

El Radar



El Radar

Técnicas

Adoptar

1. Aplicando gestión de producto a plataformas internas
2. Infraestructura CI/CD como servicio
3. Eliminación de dependencias
4. Ejecutar el costo como función de aptitud arquitectónica

Probar

5. Anotaciones sobre accesibilidad en diseños
6. Plataformas limitadas de código bajo
7. Demo de frontend para productos solo API
8. Arquitectura Lakehouse
9. Credenciales verificables

Evaluar

10. Diseño de pruebas de componentes conscientes de la accesibilidad
11. Desarrollo test-first asistido por IA
12. LLMs específicos de dominio
13. Pruebas de accesibilidad guiadas inteligentes
14. Logseq como base de conocimiento del equipo
15. Prompt engineering
16. Análisis de alcance al probar infraestructura
17. Autohospedaje de modelos de lenguaje grandes
18. Dar seguimiento a la salud por sobre la deuda
19. Seguridad de confianza cero para CI/CD

Resistir

20. Gestión despreocupada de webhooks
21. Lambda pinball
22. Planificar para la plena utilización

Plataformas

Adoptar

23. Contentful
24. GitHub Actions
25. K3s

Probar

26. Apache Hudi
27. Arm en la nube
28. Ax
29. DuckDB
30. Feature Store
31. RudderStack
32. Strapi
33. TypeDB

Evaluar

34. Autoware
35. Cozo
36. Dapr
37. Immuta
38. Matter
39. Modal
40. Neon
41. OpenLineage
42. Claves de Acceso
43. Spin

Resistir

44. Denodo como una herramienta principal de transformación de datos

El Radar

Herramientas

Adoptar

45. DVC

Probar

46. Akeyless
47. Apicurio Registry
48. Catálogo de eventos
49. FOSSA
50. Gitleaks
51. Helmfile
52. IBM Equal Access Accessibility Checker
53. Ktlint
54. Kubeflow
55. Mend SCA
56. Mozilla SOPS
57. Ruff
58. Soda Core
59. Steampipe
60. Terraform Cloud Operator
61. TruffleHog
62. Typesense
63. Vite

Evaluar

64. axe Linter
65. ChatGPT
66. DataFusion
67. Deepchecks
68. Herramientas de traducción de tokens de diseños
69. Devbox
70. Evidently
71. Giskard
72. GitHub Copilot
73. iamlive
74. Kepler
75. Agente de Secretos Externos de Kubernetes
76. Kubeshark
77. Obsidian
78. Ory Kratos
79. Ejecutor auto alojado para GitHub de Philips

Resistir

—

Lenguajes y Frameworks

Adoptar

80. Gradle Kotlin DSL
81. PyTorch

Probar

82. dbt-unit-testing
83. Jetpack CameraViewfinder
84. Jetpack DataStore
85. Mikro ORM
86. Preferencia de idioma por aplicación
87. Quarto
88. River
89. Stencil
90. Synthetic Data Vault
91. Vitest

Evaluar

92. .NET 7 Native AOT
93. .NET MAUI
94. dbt-expectations
95. Directus
96. Ferrocene
97. Flutter para sistemas embebidos
98. Fugue
99. Galacean Engine
100. LangChain
101. mljar-supervised
102. nanoGPT
103. pandera
104. Qwik
105. SolidJS
106. Turborepo
107. API para dispositivos WebXR

Resistir

—

Técnicas

Adoptar

1. Aplicando gestión de producto a plataformas internas
2. Infraestructura CI/CD como servicio
3. Eliminación de dependencias
4. Ejecutar el costo como función de aptitud arquitectónica

Probar

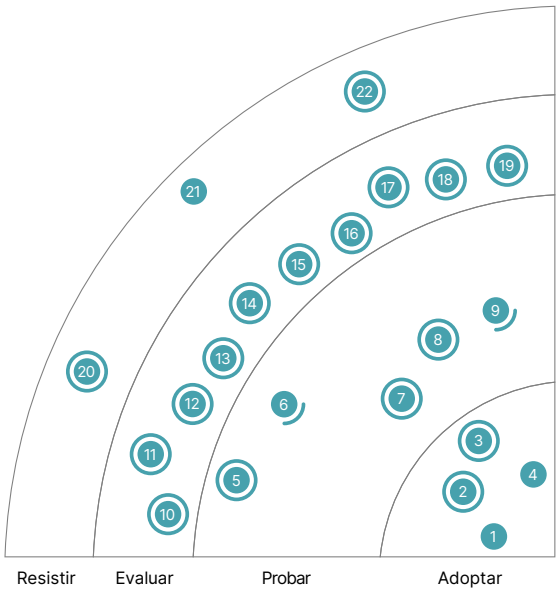
5. Anotaciones sobre accesibilidad en diseños
6. Plataformas limitadas de código bajo
7. Demo de frontend para productos solo API
8. Arquitectura Lakehouse
9. Credenciales verificables

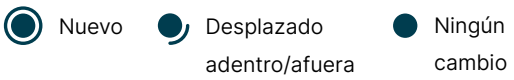
Evaluar

10. Diseño de pruebas de componentes conscientes de la accesibilidad
11. Desarrollo test-first asistido por IA
12. LLMs específicos de dominio
13. Pruebas de accesibilidad guiadas inteligentes
14. Logseq como base de conocimiento del equipo
15. Prompt engineering
16. Análisis de alcance al probar infraestructura
17. Autohospedaje de modelos de lenguaje grandes
18. Dar seguimiento a la salud por sobre la deuda
19. Seguridad de confianza cero para CI/CD

Resistir

20. Gestión despreocupada de webhooks
21. Lambda pinball
22. Planificar para la plena utilización





1. Aplicando gestión de producto a plataformas internas

Adoptar

Seguimos recibiendo buenos comentarios desde los equipos que se encuentran aplicando gestión de producto a plataformas internas. Sin embargo, hay que recordar una característica clave: no se trata sólo de la estructura del equipo o de cambiar el nombre de los equipos de plataforma existentes, sino también de aplicar prácticas de trabajo centradas en el producto dentro del equipo. En concreto, hemos recibido comentarios de que los equipos se enfrentan a dificultades con esta técnica a menos que tengan una mentalidad centrada en el producto. Es probable que esto implique roles adicionales, como un gerente de producto, además de cambios en otras áreas, como la recopilación de requisitos y la medición del éxito. Trabajar de esta manera significa un ejercicio de empatía con los usuarios internos (los equipos de desarrollo) y colaborar con ellos en el diseño. Los gerentes de producto de plataforma crean hojas de ruta y garantizan que la plataforma aporte valor a la empresa y mejore la experiencia de los desarrolladores. Seguimos considerando que esta técnica es clave para crear plataformas internas que permitan desplegar nuevas soluciones digitales con rapidez y eficacia.

2. Infraestructura CI/CD como servicio

Adoptar

Las opciones para la infraestructura CI/CD como servicio han aumentado y madurado a tal nivel que los casos en los que vale la pena gestionar toda una infraestructura CI propia son cada vez menos. Usar servicios gestionados como GitHub Actions, Azure DevOps o Gitlab CI/CD tienen en común todas las ventajas (y trade-offs) con los servicios de gestión en la nube. No se tiene que gastar tiempo, esfuerzo y costos de hardware en mantenimiento y operaciones de esta infraestructura usualmente compleja. Los equipos pueden aprovechar la elasticidad y el autoservicio, mientras que el aprovisionamiento de los agentes adecuados u obtener un nuevo plugin o feature suelen ser un cuello de botella en las empresas que alojan por sí mismos el CI. Incluso los casos de uso que requieren ejecutar la compilación y verificación en su propio hardware pueden ahora ser mayormente cubiertos con ejecutores auto alojados (hemos escrito sobre algunos para GitHub Actions, actions-runner-controller y el ejecutor de GitHub auto alojado Philips). Cabe considerar, sin embargo, que no tendrá seguridad inmediata sólo porque está usando un servicio gestionado; mientras los servicios maduros proveen todas las características de seguridad necesarias, aún necesitará usarlas para implementar políticas de confianza cero para su infraestructura CI/CD.

3. Eliminación de dependencias

Adoptar

Los starter kits y las plantillas son muy utilizados en los proyectos de software para acelerar la configuración inicial, pero pueden introducir muchas dependencias innecesarias para un proyecto en particular. Es importante practicar la eliminación de dependencias , es decir, revisar periódicamente estas dependencias y eliminar las que no se utilicen. Esto ayuda a reducir los tiempos de compilación e implementación y disminuye la superficie de ataque del proyecto al eliminar posibles vulnerabilidades. Aunque no se trata de una técnica nueva, dada la creciente frecuencia de los ataques a las cadenas de suministro de software, abogamos por que se le preste una atención renovada.

4. Ejecutar el costo como función de aptitud arquitectónica

Adoptar

Estimar, rastrear y predecir automáticamente el costo de ejecución de la infraestructura en la nube es crucial para las organizaciones de hoy. Los modelos de precios inteligentes de los proveedores de la nube, combinados con la proliferación de la parametrización de precios y la naturaleza dinámica

de la arquitectura actual, pueden generar costos de ejecución sorprendentemente elevados. Si bien esta técnica ha estado en adopción desde 2019, queremos resaltar la importancia de considerar el costo de ejecución como una función de adecuación de la arquitectura especialmente hoy, debido a la adopción acelerada de la nube y la creciente atención a las prácticas FinOps. Muchas plataformas comerciales ofrecen herramientas que pueden consolidar y clarificar los costes de la nube a los líderes empresariales. Algunos de ellos están diseñados para mostrar los costos de ejecución de la nube a organizaciones financieras o a las unidades de negocio que los han originado.

Sin embargo, las decisiones de consumo de la nube generalmente se toman a nivel de ingeniería, donde se diseñan los sistemas. Es importante que los ingenieros que toman decisiones de diseño tengan alguna forma de predecir el impacto en el costo de sus decisiones arquitectónicas. Algunos equipos automatizan esta predicción al principio del ciclo de vida del desarrollo. Herramientas como Infracost ayudan a los equipos a predecir el impacto de los costos al pensar en posibles cambios en la infraestructura como código. Este cálculo se puede automatizar e integrar con el pipeline de CD. Tenga en cuenta que el costo se verá afectado por las decisiones arquitectónicas combinadas con la demanda de uso real; para hacer esto correctamente, necesita buenas proyecciones del nivel de demanda de uso esperado. Un feedback temprano y frecuente del costo de ejecución puede evitar que se dispare. Cuando el costo previsto se desvía de lo esperado o aceptable, el equipo puede analizar si es hora de evolucionar la arquitectura.

5. Anotaciones sobre accesibilidad en diseños

Probarl

Cuanto antes se considere la accesibilidad en el desarrollo de software, más fácil y barato es asegurar que lo que se construye funcione para la mayor cantidad de personas posible. Las herramientas que permiten comunicar anotaciones sobre accesibilidad en diseños ayudan a los equipos a tener en cuenta elementos importantes como la estructura del documento, el HTML semántico y los textos alternativos desde el principio. Esto les permite garantizar que las interfaces de usuario cumplan con los estándares globales de accesibilidad, así como abordar fallos comunes que son muy fáciles de evitar. Figma ofrece una gran variedad de complementos para anotaciones de accesibilidad: The A11y Annotation Kit, Accessibility Annotation Library de Twitter y el paquete de herramientas de Axe Axe for Designers.

6. Plataformas limitadas de código bajo

Probarl

Siempre hemos defendido escribir menos código. La simplicidad es uno de los valores fundamentales que subyacen a nuestros Sensible Defaults para el desarrollo de software. Por ejemplo, tratamos de no anticiparnos a las necesidades y sólo introducimos código que satisfaga los requisitos comerciales inmediatos y nada más. Una forma de lograr esto es crear plataformas de ingeniería que lo hagan posible a nivel organizacional.

Este es también el objetivo declarado de muchas plataformas de código reducido (o low-code, de sus siglas en inglés) que están ganando popularidad en este momento. Plataformas como Mendix o Microsoft Power Apps pueden exponer procesos comerciales comunes para reutilizar y simplificar los problemas de obtener una nueva funcionalidad directamente desplegada en mano para los usuarios. Estas plataformas han logrado grandes avances en los últimos años con buena testeabilidad y soporte para buenas prácticas de ingeniería. Son particularmente útiles para tareas simples o aplicaciones orientadas a eventos. Sin embargo, pedirles que se adapten a una gama casi infinita de requisitos comerciales genera complejidad. Aunque los desarrolladores pueden estar escribiendo menos (o cero) código, también deben convertirse en expertos en una plataforma comercial que lo abarque todo. Aconsejamos a las empresas que consideren si necesitan toda la funcionalidad que brindan

estos productos o si es mejor buscar plataformas limitadas de código reducido , ya sea desarrollando su propia plataforma como un producto interno o restringiendo cuidadosamente el uso de productos comerciales de código reducido a aquellas tareas simples en las que destacan.

7. Demo de frontend para productos solo API

Probar!

Uno de los grandes desafíos en el desarrollo de APIs es capturar y comunicar su valor de negocio. Las API son por naturaleza artefactos técnicos y mientras que los equipos de desarrollo pueden entender fácilmente los contratos JSON, las especificaciones OpenAPI (Swagger) y demos en Postman (Postman), los stakeholders del negocio, por el contrario, tienden a responder mejor a demos con las que pueden interactuar. El valor del producto se articula de forma más clara cuando se puede ver y tocar, por lo que en algunas ocasiones encontramos más valioso invertir en una demo de frontend para productos solo API. Cuando se crea una interfaz de usuario personalizada en conjunto a un producto de API, los stakeholders pueden ver las analogías con los formularios o informes que podrían resultarles más familiares. A medida que la riqueza y la interacción de la demo se desarrolla, esto les permite tomar decisiones mejor informadas sobre la dirección que debe tomar sobre el producto API. Trabajar en una interfaz de usuario tiene un beneficio adicional y es poder aumentar la empatía del equipo de desarrollo hacia los usuario de negocio. Esta técnica no es nueva, ya que la hemos estado aplicando exitosamente cuando ha sido necesario, siempre y cuando los productos de tipo API estén presentes. Sin embargo, debido a que esta técnica no es ampliamente conocida, pensamos que vale la pena darle atención.

8. Arquitectura Lakehouse

Probar!

Arquitectura Lakehouse es un estilo arquitectónico que combina la escalabilidad de los data lakes con la confiabilidad y el rendimiento de los almacenes de datos. Permite a las organizaciones almacenar y analizar grandes volúmenes de datos diversos en una sola plataforma en lugar de tenerlos en niveles separados de data lakes y almacenes de datos, utilizando las mismas técnicas y herramientas familiares basadas en SQL. Si bien el término a menudo se asocia con proveedores como Databricks, existen otras alternativas como Delta Lake, Apache Iceberg y Apache Hudi que vale la pena considerar. La arquitectura Lakehouse puede complementar las implementaciones de data mesh. Los equipos autónomos de productos de datos pueden optar por aprovechar un Lakehouse dentro de sus propios productos de datos.

9. Credenciales verificables

Probar!

Cuando las mencionamos por primera vez hace tres años, las credenciales verificables (VC, por sus siglas en inglés), lo describimos como un estándar interesante con algunas prometedoras aplicaciones potenciales, aunque era muy poco conocido y entendido fuera de la comunidad de entusiastas. Esto se ponía en evidencia en el caso de las instituciones que otorgan credenciales (entes o agencias gubernamentales, por ejemplo), las cuales serían las responsables de la implementación de los estándares. Tres años y una pandemia después, la demanda de credenciales electrónicas que sean criptográficamente seguras, respeten la privacidad de los usuarios y se puedan verificar automáticamente por máquinas, ha crecido; y como resultado de ello los gobiernos han comenzado a aprovechar el potencial de las VC. El estándar W3C sitúa al titular de las credencial en el centro, en una experiencia similar a cuando se usan credenciales físicas: los usuarios pueden almacenar sus credenciales verificables en sus carteras digitales, y mostrarlas a cualquiera en cualquier momento sin la necesidad de permisos del ente emisor de la credencial. Este enfoque descentralizado permite a los

usuarios gestionar mejor y revelar selectivamente su propia información, lo que mejora en gran medida la protección de la privacidad de los datos.

Durante los últimos seis meses, varios de nuestros equipos han participado en proyectos que involucran las tecnologías de las credenciales verificables. No es de sorprender que los escenarios varíen según el país o la dependencia gubernamental. Nuestros equipos han explorado diferentes combinaciones de identificadores descentralizados, credenciales verificables y presentaciones verificables en múltiples proyectos. Este campo está aún en desarrollo, y ahora que tenemos más experiencia, queremos hacerle seguimiento en el Radar.

10. Diseño de pruebas de componentes conscientes de la accesibilidad

Evaluar

Uno de los muchos lugares en el proceso de entrega de software para considerar los requisitos de accesibilidad desde el principio, es al probar componentes web. Plugins de frameworks para pruebas como chai-a11y-axe proveen aserciones en sus API para verificar los conceptos básicos. Pero además de usar lo que ofrecen los frameworks de pruebas, el diseño de pruebas de componentes conscientes de la accesibilidad ayuda aún más a proporcionar todos los elementos semánticos que necesitan los lectores de pantalla y otras tecnologías de asistencia.

Primeramente, en lugar de usar ID de prueba o clases para encontrar y seleccionar los elementos que desea validar, use un principio de identificacion de elementos por roles ARIA u otros atributos semanticos que son usados por las tecnologias de asistencia. Algunas librerias de pruebas, como Testing Library, incluso recomiendan esto en su documentación. En segundo lugar, no solo pruebe las interacciones con click; también considere a los usuarios que no pueden usar un ratón, o ver la pantalla, y considere agregar pruebas adicionales para el teclado y otras interacciones.

11. Desarrollo test-first asistido por IA

Evaluar

Como muchas otras veces en la industria del software, hemos estado explorando la rápida evolución de las herramientas de IA que pueden asistirnos al escribir código. Vemos muchas personas alimentando ChatGPT con una implementación, y luego le piden que les genere los test asociados a esa implementación. Sin embargo, como somos grandes partidarios del TDD, y casi nunca nos gusta alimentar un modelo externo con nuestro código de implementación potencialmente sensible, uno de nuestros experimentos en este campo es una técnica que llamamos desarrollo test-first asistido por IA. En este enfoque, le pedimos a ChatGPT que genere los tests por nosotros, y a partir de ahí un desarrollador implementa esa funcionalidad. Más específicamente, primero describimos el stack tecnológico y los patrones de diseño que estamos usando a través de un fragmento disponible que será reusable en múltiples casos de uso. Después describimos la funcionalidad específica que queremos implementar, incluyendo los criterios de aceptación. Basado en todo esto, le pedimos a ChatGPT que genere un plan de implementación para esta funcionalidad usando nuestro estilo de arquitectura y tech stack. Una vez que hemos revisado el plan de implementación, le pedimos que genere los tests para nuestros criterios de aceptación.

Este planteamiento nos ha funcionado extremadamente bien: requirió que el equipo tuviera preparada una descripción muy concisa sobre su estilo de arquitectura y ayudó a desarrolladores junior y nuevos miembros de equipos a desarrollar funcionalidades alineadas con el estilo ya existente en el equipo. El mayor inconveniente de este planteamiento es que aunque no estemos ofreciendo al modelo nuestro código fuente, si que le estamos alimentando potencialmente con información sensible como nuestro stack tecnológico o las descripciones de nuestras funcionalidades. Los equipos deberán asegurarse que trabajan con sus consejeros legales para evitar cualquier tipo de incumplimiento de la

propiedad intelectual, al menos hasta que aparezca disponible una versión “para empresas” de estas herramientas de IA.

12. LLMs específicos de dominio

Evaluar

Anteriormente en el Radar Tecnológico hemos presentado large language models (LLM por sus siglas en inglés) como BERT y ERNIE sin embargo, los LLMs específicos de dominio son una tendencia emergente. Ajustar un LLM de propósito general con datos específicos de un dominio puede adaptarlo para diversas tareas, incluyendo búsqueda de información, incremento del servicio de atención al cliente y creación de contenido. Esta práctica ha mostrado resultados prometedores en industrias como la jurídica y financiera, como ha demostrado OpenNyAI en el análisis de documentos jurídicos. Con cada vez más organizaciones experimentando con LLMs y nuevos modelos como GPT4 siendo lanzados, podemos esperar más casos de uso específicos de dominio en el futuro cercano.

Sin embargo, hay ciertos desafíos y dificultades que considerar. Primero, los LLMs pueden con certeza estar equivocados, por lo que es esencial incluir en el proceso mecanismos que aseguren la exactitud de los resultados. Segundo, los LLMs de terceros pueden retener y compartir tus datos, presentando un riesgo para la propiedad y confidencialidad de la información. Las organizaciones deberían revisar cuidadosamente los términos de uso y confiabilidad de los proveedores o considerar entrenar y ejecutar LLMs sobre una infraestructura que ellos mismos controlen. Como con cualquier tecnología nueva, las empresas deben ir con cuidado, entendiendo las implicaciones y riesgos asociados con la adopción de los LLM.

13.Pruebas de accesibilidad guiadas inteligentes

Evaluar

Puede ser un poco desalentador hacer que una aplicación web cumpla con las tecnologías de asistencia cuando tú no las usas, y no estás familiarizado con directivas como las Guías de Accesibilidad para Contenido Web (WCAG, por sus siglas en inglés). Las Pruebas de Accesibilidad guiadas inteligentes son una categoría de herramientas que ayudan a comprobar si has hecho lo correcto sin necesidad de ser un experto en accesibilidad. Estas herramientas son extensiones del navegador que escanean tu sitio web, resumen cómo lo interpretaría la tecnología de asistencia y luego te hacen una serie de preguntas para confirmar si la estructura y los elementos que creó son los previstos. Usamos axe DevTools, Accessibility Insights for Web o el ARC Toolkit en algunos de nuestros proyectos.

14. Logseq como base de conocimiento del equipo

Evaluar

La gestión del conocimiento del equipo es un concepto familiar, con equipos que utilizan herramientas como wikis para almacenar información e incorporar nuevos miembros. Algunos de nuestros equipos ahora prefieren usar Logseq como base de conocimiento del equipo. Un sistema de gestión del conocimiento de código abierto, Logseq funciona con una base de datos orientada a grafos, ayuda a los usuarios a organizar conceptos, notas e ideas y se puede adaptar para uso en equipo con almacenamiento basado en Git. Logseq permite que los equipos construyan una base de conocimientos democrática y accesible, brindando a cada miembro una experiencia de aprendizaje personalizada y facilitando una incorporación eficiente al equipo. Sin embargo, como ocurre con cualquier herramienta de gestión del conocimiento, los equipos deberán aplicar un buen mantenimiento y gestión de su base de conocimiento para evitar la sobrecarga de información o la desorganización.

Si bien una funcionalidad similar está disponible en herramientas como Obsidian, la diferencia clave radica en el enfoque de Logseq en su consumo, con enlaces a párrafos que permiten que los miembros del equipo encuentren rápidamente el contexto relevante sin tener que leer un artículo completo.

15. Prompt engineering

Evaluar

Prompt engineering se refiere al proceso de diseñar y refinar prompts para modelos generativos de IA para obtener respuestas de alta calidad del modelo. Esto implica elaborar cuidadosamente prompts que sean específicos, claros y relevantes para la tarea o aplicación deseada a fin de obtener resultados útiles del modelo. Prompt engineering tiene como objetivo mejorar las capacidades del modelo de lenguaje de gran tamaño o LLM (Large Language Model) en tareas como responder preguntas, razonamiento aritmético o en contextos específicos de algún dominio. Para la creación de software, puedes usar prompt engineering para obtener un LLM para escribir una historia, una API o un conjunto de pruebas basado en una breve conversación con un stakeholder o incluso algunas notas. El desarrollo de técnicas de prompting efectivas se está convirtiendo en una habilidad valiosa para trabajar con sistemas de IA. Existe un debate sobre si prompt engineering es un arte o una ciencia, y se deben considerar los posibles riesgos de seguridad, como los “ataques de prompt injection”.

16. Análisis de alcance al probar infraestructura

Evaluar

Al desplegar la infraestructura como código, hemos observado que se puede perder mucho tiempo diagnosticando y reparando problemas de producción derivadas de la incapacidad de los sistemas para comunicarse entre sí. Dado que la topología de red entre ellos puede ser más compleja, es posible que no se pueda atravesar toda la ruta aunque los puertos y endpoints individuales se hayan configurado correctamente. Las prácticas de pruebas de infraestructura usualmente incluyen la verificación de que los puertos correctos están abiertos o cerrados, o que se pueda acceder a un punto final (endpoint), pero sólo recientemente hemos empezado a hacer análisis de alcance al probar la infraestructura. El análisis generalmente implica más que simples determinaciones de sí y no. Por ejemplo, una herramienta puede recorrer e informar sobre múltiples rutas a través de pasarelas de tránsito. Esta técnica es compatible con herramientas de los principales proveedores de nube. Azure cuenta con un servicio llamado Network Watcher el cual se puede programar en pruebas automatizadas y GCP admite Pruebas de conectividad (Connectivity Tests). Ahora, en AWS, se puede probar el alcance a través de cuentas dentro de la misma organización.

17. Autohospedaje de modelos de lenguaje grandes

Evaluar

Los Modelos de Lenguaje Grandes (LLMs por sus siglas en inglés) suelen necesitar una infraestructura de GPU considerable para funcionar. Empezamos a ver conversores, como llama.cpp, tque permiten ejecutar estos modelos en una variedad de hardware como Raspberry Pis, ordenadores portátiles y servidores básicos. Por lo tanto, podríamos decir que el autohospedaje de Modelos de Lenguaje Grandes (LLMs) ya es una realidad. Actualmente hay algunos modelos de código abierto que pueden autohospedarse, como GPT-J, GPT-JT and LLaMA. Este método tiene varias ventajas: un mejor control de ajuste para casos de uso específicos, seguridad y privacidad mejorada, así como acceso sin conexión. Sin embargo, recomendamos evaluar con prudencia la capacidad dentro de la organización y el coste de ejecutar tales modelos antes de decidir autohospedar.

18. Dar seguimiento a la salud por sobre la deuda

Evaluar

El seguimiento de la deuda técnica es un tópico perenne en las organizaciones de entrega de software. ¿Qué es deuda técnica y qué no lo es? ¿Cómo se prioriza? Y, lo que es más importante, ¿Cómo expresas el valor de pagarla a los stakeholders internos? Siguiendo la manera de razonar del Manifiesto Ágil — "mientras haya valor en el elemento de la derecha, valoramos más el elemento de la izquierda" — nos gusta la idea de dar seguimiento a la salud por sobre la deuda. La gente de REA en Australia comparte un buen ejemplo de cómo puede hacerse este seguimiento de la salud a través de un seguimiento de las calificaciones del sistema en las categorías de desarrollo, operaciones y arquitectura.

Centrarse en la salud en lugar de la deuda es un enfoque más constructivo. Conecta a un equipo con el valor final de la reducción de la deuda y les ayuda a priorizar. Lo ideal es que cada parte de la deuda técnica abordada pueda relacionarse con una de las expectativas acordadas. Los equipos deben tratar la calificación de la salud igual que otros objetivos de nivel de servicio (SLOs) y priorizar las mejoras siempre que salgan de la "zona verde" para una categoría dada.

19. Seguridad de confianza cero para CI/CD

Evaluar

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 engloba un conjunto de técnicas: si está disponible, autentifica tus pipelines con el proveedor de cloud mediante mecanismos de identidad federada como OIDC, en vez de darles acceso directo a tus secretos. Implementa el principio de menos privilegios minimizando el acceso de usuarios individuales o cuentas de ejecución, en lugar de usar "cuentas de usuario dios" con acceso ilimitado. Utiliza tus cuentas de ejecutores de forma efímera en vez de reutilizarlos, para reducir el riesgo de exposición de los secretos de ejecuciones pasadas o ejecutar tus trabajos en cuentas comprometidas. Mantén actualizado el software de tus agentes y ejecutores. Monitorea la integridad, confidencialidad y disponibilidad de tus sistemas de CI/CD de la misma forma que monitoreas tu software en 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.

20. Gestión despreocupada de webhooks

Resistir

A medida que el trabajo en remoto continúa creciendo, también lo hace la adopción de plataformas de colaboración por chat y ChatOps. Estas plataformas a menudo ofrecen webhooks como una forma simple de automatizar el envío de mensajes y notificaciones, pero estamos notando una tendencia preocupante: la gestión despreocupada de los webhooks — donde se están tratando como configuración en vez de un secreto o credencial. Esto puede conllevar ataques de 'phishing' y espacios internos comprometidos.

Los webhooks son credenciales que ofrecen un acceso privilegiado a un espacio interno que puede contener API keys que pueden ser fácilmente extraídas y utilizadas directamente. El hecho de no tratarlas como secretos abre la posibilidad de que los ataques de phishing tengan éxito. Los webhooks en los repositorios Git pueden ser fácilmente extraídos y usados para enviar payloads fraudulentos, las cuales el usuario puede no tener ninguna forma de autenticar. Para mitigar esta amenaza, los equipos que se encargan de 'webhooks' necesitan cambiar su cultura y tratar a los webhooks como credenciales sensibles. Los desarrolladores de software que construyen integraciones con plataformas de 'ChatOps' también deben ser conscientes del riesgo y asegurarse de que los webhooks se gestionen con las medidas de seguridad apropiadas.

21. Lambda pinball

Resistir

Si bien las arquitecturas serverless pueden ser extremadamente útiles para resolver algunos problemas, tienen un cierto nivel de complejidad, especialmente cuando involucran una ejecución y flujo de datos no triviales a través de múltiples Lambdas interdependientes – esto a veces puede resultar en una arquitectura Lambda pinball Nuestros equipos han observado que mantener y probar las arquitecturas de Lambda pinball puede ser un gran desafío: entender la infraestructura, el despliegue, el diagnóstico y la depuración puede volverse difícil. A nivel de código, el mapeo simple entre los conceptos de dominio y las múltiples Lambdas involucradas es prácticamente imposible, lo que hace que cualquier cambio y añadidura sea un desafío. Si bien creemos que la tecnología serverless es la opción adecuada para algunos problemas y dominios, no es una "solución milagrosa" para todos los problemas, por lo que debe tratar de evitar el Lambda pinball. Un patrón que puede ayudar es establecer una distinción entre interfaces públicas y publicadas y aplicar límites de dominio con interfaces publicadas entre ellas.

22. Planning for full utilization

Resistir

Aunque la práctica de crear un exceso de capacidad en el proceso de entrega es bien conocida en la comunidad de gestión de productos, todavía vemos demasiados equipos que planifican la plena utilización de los miembros del equipo. Reservar algo de capacidad durante la planificación del sprint generalmente conduce a una mejor previsibilidad y mejor calidad; promueve la resiliencia del equipo ante eventos inesperados como enfermedades, problemas de producción, solicitudes inesperadas de producto y deuda técnica, al tiempo que permite actividades productivas como la creación de equipos y la ideación que pueden conducir a la innovación del producto. Trabajar con una utilización inferior a la plena significa que los equipos pueden estar más atentos a la solidez del software resultante y prestar más atención a las señales de observabilidad adecuadas. Nuestra experiencia es que un equipo plenamente utilizado también provoca un colapso en el throughput, al igual que una autopista totalmente utilizada crea un tráfico lento y desmoralizador. Por ejemplo, cuando uno de nuestros equipos tuvo problemas de soporte impredecibles, vieron un aumento del 25% en el throughput y una disminución del 50% en la volatilidad del tiempo de ciclo mediante la planificación de la velocidad de las funcionalidades basada las capacidades de sólo dos -de las tres- parejas de desarrolladores.

Plataformas

Adoptar

23. Contentful
24. GitHub Actions
25. K3s

Probar

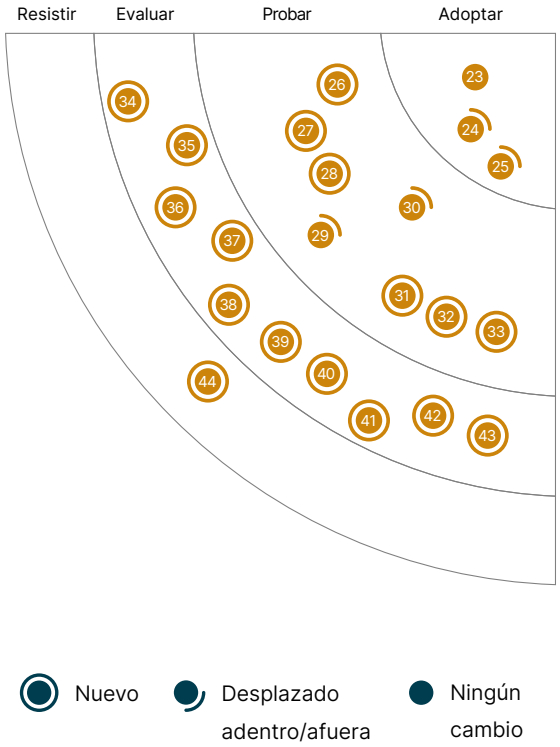
26. Apache Hudi
27. Arm en la nube
28. Ax
29. DuckDB
30. Feature Store
31. RudderStack
32. Strapi
33. TypeDB

Evaluar

34. Autoware
35. Cozo
36. Dapr
37. Immuta
38. Matter
39. Modal
40. Neon
41. OpenLineage
42. Claves de Acceso
43. Spin

Resistir

44. Denodo como una herramienta principal de transformación de datos



23. Contentful

Adoptar

Los sistemas de gestión de contenidos headless se han vuelto un componente habitual de las plataformas digitales. Contentful aún es nuestra alternativa por defecto en este espacio, pero nuevos sistemas como Strapi también nos han impresionado. Particularmente, nos gusta el enfoque API-first de Contentful y la implementación de CMS como código. IEs compatible con primitivas potentes de modelado de contenidos como código y scripts de evolución de modelo de contenidos, lo cual permite tratarlo como otros esquemas de almacenamiento de datos y habilita las prácticas de diseño evolutivo de bases de datos que deben aplicarse en el desarrollo de CMS. Recientemente, Contentful ha publicado un framework de aplicación para escribir aplicaciones que faciliten la adaptación de Contentful a procesos empresariales individuales y la integración con otros servicios. Las aplicaciones se pueden compilar por y para una organización concreta, pero también está surgiendo un sitio de mercado para apps.

24. GitHub Actions

Adoptar

GitHub Actions se ha convertido en el punto de partida por defecto de muchos equipos que necesitan poner en marcha rápidamente CI o CD en un ambiente totalmente nuevo. Entre otras cosas, puede soportar flujos más complejos y llamar a otras acciones en acciones compuestas. A pesar de que el ecosistema GitHub Marketplace sigue creciendo, nosotros aún recomendamos tomar precauciones al otorgar acceso a sus pipelines en ejecución por parte de GitHub Actions creadas por terceros. Recomendamos seguir los consejos de GitHub sobre security hardening para evitar compartir secretos de manera insegura. Sin embargo, la conveniencia de crear un build workflow directamente en GitHub que está junto a su código fuente combinado con la habilidad de correr las GitHub Actions localmente, usando una herramienta de código abierto como act, ies una opción convincente que ha simplificado la configuración y onboarding de nuestros equipos.

25. K3s

Adoptar

K3s sigue siendo nuestra distribución de Kubernetes por defecto para edge computing y entornos con recursos limitados. K3s es un Kubernetes ligero totalmente compatible y con una reducida sobrecarga operacional. Utiliza sqlite3 por defecto como almacenamiento en lugar de etcd. Su consumo de memoria es reducido ya que ejecuta todos los componentes importantes en un único proceso. Hemos usado K3s en entornos como sistemas de control industrial y terminales de punto de venta y estamos muy satisfechos con nuestra decisión. Como el runtime de K3s, containerd, ahora soporta wasm, K3s puede ejecutar y manipular directamente cargas de trabajo en WebAssembly reduciendo todavía más la sobrecarga en tiempo de ejecución.

26. Apache Hudi

Probar

Apache Hudi es una plataforma data lake de código abierto que provee garantías transaccionales ACID al data lake. Nuestros equipos han tenido una muy buena experiencia al utilizar Hudi en un escenario de alto volumen y de alta capacidad mediante inserciones y upserts en tiempo real. En particular, nos gusta la flexibilidad que Hudi ofrece para personalizar el algoritmo de compactación, que ayuda a abordar problemas con “archivos pequeños”. Apache Hudi se encuentra dentro de la misma categoría que Delta Lake y Apache Iceberg. Todos ellos soportan características similares, pero cada uno difiere en sus implementaciones subyacentes y listas detalladas de características.

27. Arm en la nube

Probar

Las instancias de cómputo en la nube se han vuelto cada vez más populares en los últimos años debido a su costo y eficiencia energética en comparación con las tradicionales basadas en instancias x86. Muchos proveedores ofrecen ahora instancias basadas en Arm, incluyendo a AWS, Azure y GCP. El beneficio a nivel de costo de usar Arm en la nube puede ser particularmente ventajoso para negocios que manejan grandes cantidades o necesitan escalar. Basado en nuestra experiencia, recomendamos las instancias basadas en Arm para todos los tipos de carga a menos que haya dependencias específicas de la arquitectura. Las herramientas para dar soporte a múltiples arquitecturas, como las imágenes multi-arquitectura de Docker, también simplifican los flujos de trabajo de creación e implementación.

28. Ax

Probar

Ante el desafío de explorar grandes espacios de configuración, donde puede llevar un tiempo considerable evaluar una configuración dada, los equipos pueden recurrir a la experimentación adaptativa, un proceso iterativo guiado por máquina, para encontrar soluciones óptimas de una manera eficiente en recursos. Ax es una plataforma para gestionar y automatizar experimentos adaptativos, incluyendo experimentos de ML, A/B testing y simulaciones. Actualmente, soporta dos estrategias de optimización: optimización Bayesiana usando BoTorch, la cual se basa en PyTorch, y contextual bandits. Cuando Facebook liberó Ax y BoTorch, describió casos de uso como el incrementar la eficiencia de la infraestructura de back-end, ajustando los modelos de clasificación y optimizando la búsqueda de hiperparámetros para una plataforma de ML. Hemos tenido buenas experiencias con Ax para una variedad de casos de uso y, si bien existen herramientas para el ajuste de hiperparámetros, no conocemos una plataforma que brinde funcionalidad en un ámbito similar al de Ax.

29. DuckDB

Probar

DuckDB es una base de datos integrada basada en columnas para cargas de trabajo analíticas y de ciencia de datos. Los analistas de datos suelen cargar los datos localmente en herramientas como pandas o data.table para analizar patrones rápidamente y formular hipótesis antes de escalar la solución en el servidor. Sin embargo, ahora estamos usando DuckDB para tales casos de uso, porque desbloquea el potencial para hacer un análisis más grande que la memoria. DuckDB admite uniones de rango, ejecución vectorizada y control de concurrencia multiversión (MVCC) para grandes transacciones, y nuestros equipos están muy contentos con eso.

30. Feature Store

Probar

Cualquier sistema de software debe representar adecuadamente el ámbito dado en el que se emplea y siempre debe tener en cuenta objetivos y metas clave. Los proyectos de machine learning (ML) no son diferentes. Feature engineering es un aspecto crucial de la ingeniería y diseño de sistemas de software de ML. Feature Store es un concepto arquitectónico relacionado que facilita la identificación, el descubrimiento y la supervisión de las características (features) relevantes para un determinado dominio o problema comercial. La implementación de este concepto implica una combinación de diseño arquitectónico, ingeniería de datos y gestión de infraestructuras para crear un sistema de ML escalable, eficiente y fiable. Desde el punto de vista de las herramientas, se pueden encontrar plataformas de código abierto y completamente gestionadas, pero son sólo una parte de

este concepto. En el diseño integral de los sistemas de ML, la implementación de un feature store permite las siguientes capacidades: la habilidad de (1) definir las features adecuadas; (2) mejorar la reutilización y hacer que las features estén disponibles de forma coherente, independientemente del tipo de modelo, lo que también incluye la configuración de las pipelines de feature engineering que curan los datos como se describe en el feature store; (3) permitir el descubrimiento de features y (4) permitir su consumo. Nuestros equipos aprovechan los feature stores en producción para ver sus frutos en los sistemas de ML de extremo a extremo.

31. RudderStack

Probar

RudderStack es una plataforma de gestión de datos de cliente (CDP en inglés) que facilita el almacenamiento de datos en un data warehouse o data lake. Este enfoque, cada vez más conocido como Headless CDP, separa las funciones de la plataforma de datos de cliente de su interfaz de usuario, enfatiza la capacidad de configuración a través de APIs y el data warehouse/data lake como almacenamiento primario. Como es de esperarse de un producto en esta categoría, RudderStack cuenta con un amplio repositorio de integraciones con productos de terceros (tanto de fuente como de destino de datos) y la capacidad de ingesta de eventos personalizados. RudderStack tiene una versión comercial, y también una versión de código abierto auto alojada con funcionalidad limitada.

32. Strapi

Probar

Strapi es un sistema de gestión de contenidos (CMS) de código abierto, basado en NodeJS y headless, similar a Contentful. Existe desde hace tiempo y lo hemos utilizado con éxito en algunos proyectos. Strapi proporciona APIs REST y GraphQL, dispone de una documentación completa, cuenta con una API de modelo de datos fácil de usar y permite personalizar tanto la interfaz de usuario como la lógica.

33. TypeDB

Probar

TypeDB Es una base de datos de grafos de conocimiento, diseñada para trabajar con relaciones de datos intrincadas que facilitan la consulta y el análisis de grandes conjuntos de datos. El lenguaje de consulta TypeQL de TypeDB tiene una sintaxis similar a SQL que facilita la curva de aprendizaje para la definición de esquemas, consulta y exploración. TypeDB incluye una variedad de herramientas las cuales facilitan el trabajo con las bases de datos, incluyendo una interfaz de línea de comandos y una interfaz gráfica de usuario, TypeDB Studio, que provee algunas características para trabajar con TypeDB, como la gestión de esquemas, la consulta de datos, la visualización de relaciones o incluso la colaboración con otros. Existe una gran cantidad de documentation disponible y una comunidad activa de apoyo. Nuestros equipos la utilizaron para construir grafos de conocimiento de conceptos taxonómicos en diferentes bases de datos, y aprovecharon sus potentes capacidades de interferencia añadiendo nuevas reglas de interferencia para aumentar la eficiencia y reducir la carga de trabajo. Gracias a su intuitiva experiencia de desarrollo y su comunidad de apoyo, TypeDB es un buen candidato a considerar para cualquier equipo que busque crear soluciones de datos que dependan de relaciones de datos complejas, como los datos de lenguaje natural, motores de recomendación y grafos de conocimiento.

34. Autoware

Evaluar

Autoware es un software de código abierto de conducción autónoma construido sobre ROS (Robot Operating System) que puede ser usado para desarrollar y desplegar sistemas de conducción asistida (ADAS, por sus siglas en inglés) para una amplia gama de vehículos como coches y camiones. Provee de un set de herramientas y algoritmos para varios aspectos de la conducción autónoma, como percepción, toma de decisiones y control. También tiene un módulo de planificación y control que genera una trayectoria para el vehículo basada en su entorno y objetivos. Fomenta la innovación abierta en tecnologías de conducción autónoma. Estamos construyendo prototipos usando Autoware para validar nuevas ideas de productos y lo encontramos útil.

35. Cozo

Evaluar

Cozo es una base de datos relacional integrable que utiliza Datalog para realizar consultas. Estamos intrigados su compatibilidad con las consultas time-travel y el modelado de datos gráficos en esquemas relacionales. Nos gusta bastante que delegue el almacenamiento de datos en motores populares ya existentes, como SQLite, RocksDB, Sled y TiKV. Aunque Cozo se encuentra aún en sus primeras fases de desarrollo, creemos que merece la pena evaluarlo.

36. Dapr

Evaluar

Dapr, abreviatura de Distributed Application Runtime, ayuda a los desarrolladores a crear microservicios resilientes,con o sin estado que se ejecutan en la nube. Algunas personas pueden confundirlo con una service mesh, porque utiliza una arquitectura sidecar que se ejecuta como un proceso separado junto con la aplicación. Dapr está más orientado a las aplicaciones y se enfoca en encapsular la tolerancia a fallos y la conectividad requeridas para construir aplicaciones distribuidas. Por ejemplo, Dapr proporciona varios bloques de construcción, desde la invocación del servicio y la publicación/suscripción de mensajes hasta el bloqueo distribuido, los cuales son patrones comunes en comunicación distribuida. Uno de nuestros equipos evaluó Dapr en un proyecto reciente; dada su experiencia positiva, esperan llevarla a otros proyectos en el futuro.

37. Immuta

Evaluar

Immuta es una plataforma de seguridad de datos que permite proteger el acceso a tus datos, descubre automáticamente datos sensibles y audita cómo los mismos son usados en una organización. En el pasado, hemos mencionado la importancia de la automatización, prácticas de ingeniería y tratamiento de las políticas de seguridad como código cuando pensamos en cuestiones de seguridad. La seguridad de datos es igual. Nuestros equipos han estado explorando Immuta para administrar las políticas de datos como código y permitir un control de acceso granular, el cual va más allá de lo que el control de acceso basado en roles (RBAC por sus siglas en Inglés) puede ofrecer. Políticas manejadas a través de control de versiones pueden probarse antes de formar parte de un pipeline de Entrega e Integración Continua. En un ecosistema de datos descentralizados, como el que nos permite data mesh, tener roles específicos de dominios pueden llevar a la proliferación de roles o grupos en el sistema de identidad. La capacidad del control de accesos basado en atributos de Immuta (ABAC por sus siglas en Inglés) reduce la concesión de accesos a una ecuación matemática que asocia un “atributo” del usuario con una “etiqueta” en la fuente de datos. Esta plataforma aún es nueva, pero vale la pena destacar para las necesidades de seguridad de datos.

38. Matter

Evaluar

Matter es un estándar abierto para tecnología de hogar inteligente, lanzado por Amazon, Apple, Google, Comcast y Zigbee Alliance (ahora Connectivity Standards Alliance, o CSA). Éste permite que dispositivos funcionen con cualquier ecosistema certificado por Matter, lo que reduce la fragmentación y promueve la interoperabilidad entre dispositivos y plataformas IoT de distintos proveedores. Su enfoque en la estandarización a nivel de aplicación, compatibilidad con Wi-Fi y Thread como soluciones de conectividad y el respaldo de las principales empresas tecnológicas lo distinguen de otros protocolos como Zigbee. Aunque la cantidad de dispositivos habilitados para Matter aún es relativamente baja, su creciente importancia en el área de IoT hace que valga la pena evaluar este estándar para aquellos que buscan construir soluciones de IoT y hogar inteligente.

39. Modal

Evaluar

Modal es una plataforma como servicio (PaaS) que ofrece realizar cálculos bajo demanda, sin la necesidad de tener tu propia infraestructura. Modal te permite desplegar modelos de machine learning, trabajos de cálculo en paralelo de forma masiva, colas de tareas y aplicaciones web. Modal proporciona una abstracción en forma de contenedores que hace que el cambio de local a despliegue en nube sea instantáneo, con recarga en caliente tanto en local como en la nube. Incluso realiza el borrado de despliegues automáticamente, evitando la necesidad de una limpieza manual, aunque también puede hacerlos persistentes.

Modal está escrito por el mismo equipo que desarrolló el primer sistema de recomendaciones para Spotify. Se ocupa de la stack AI/ML completa y puede proveer recursos GPU bajo demanda, lo cual es útil si necesitas hacer uso computacional intensivo. Tanto si trabajas desde tu portátil como desde la nube, Modal simplemente funciona, proporcionando una forma fácil y eficiente de ejecutar y desplegar tus proyectos.

40. Neon

Evaluar

Neon Es una alternativa de código abierto a AWS Aurora PostgreSQL. Las bases de datos analíticas nativas de la nube han adoptado la técnica de separar el almacenamiento de los nodos de procesamiento para escalar elásticamente bajo demanda. Sin embargo, es difícil hacer lo mismo en una base de datos transaccional. Neon lo consigue con su nuevo motor de almacenamiento multiusuario para PostgreSQL. Con cambios mínimos en el código principal de PostgreSQL, Neon aprovecha AWS S3 para almacenamiento de datos a largo plazo y escala elásticamente el procesamiento hacia arriba o hacia abajo (incluyendo el escalado a cero). Esta arquitectura tiene varios beneficios — incluyendo clones baratos y rápidos, copia-en-escritura y ramificación. Nos entusiasma ver nuevas innovaciones sobre PostgreSQL. Nuestros equipos están evaluando Neon, y te recomendamos que tú también lo hagas.

41. OpenLineage

Evaluar

OpenLineage es un estándar abierto para recopilar metadatos de linaje para pipelines de datos, diseñado para instrumentar jobs mientras se ejecutan. Define un modelo genérico de ejecución, jobs y entidades de conjuntos de datos utilizando convenciones de nomenclatura consistentes. El modelo de linaje principal se puede extender definiendo facetas específicas que enriquecen esas entidades. OpenLineage resuelve el problema de interoperabilidad entre productores y consumidores de información de linaje, quienes de otra manera requerirían saber cómo comunicarse entre sí de diferentes maneras. Aunque existe un riesgo al ser otro "estándar en el medio", ser un proyecto respaldado por Linux Foundation AI & Data Foundation aumenta su posibilidad de adopción masiva. OpenLineage actualmente soporta colecciones de datos en múltiples plataformas, como Spark, Airflow y dbt, aunque los usuarios necesitan configurar sus listeners. El soporte para consumidores de datos de OpenLineage es más limitado en estos momentos.

42. Claves de Acceso

Evaluar

El "fin de las contraseñas" podría estar cerca, finalmente. Impulsadas por la alianza FIDO y respaldadas por Apple, Google y Microsoft, passkeys se acerca al uso generalizado. Al configurar un nuevo inicio de sesión con Passkeys, se generan un par de claves: el sitio web recibe la clave pública y el usuario conserva la clave privada. El manejo del inicio de sesión utiliza criptografía asimétrica. El usuario demuestra que está en posesión de la clave privada, pero, a diferencia de las contraseñas, ésta nunca se envía al sitio web. En los dispositivos de los usuarios, el acceso a Passkeys está protegido mediante datos biométricos o un PIN.

Las Passkeys se pueden almacenar y sincronizar dentro de los ecosistemas de las Grandes Tecnológicas, utilizando el iCloud Keychain de Apple, el Administrador de contraseñas de Google o Windows Hello. En la mayoría de los casos, esto solo funciona con versiones recientes del sistema operativo y del navegador. En particular, el almacenamiento de passkeys en Windows Hello no es compatible con Windows 10. Afortunadamente, el Client to Authenticator Protocol (CTAP) hace posible que las Passkeys sean guardadas en un dispositivo diferente al que crea la clave o al que la necesita para iniciar sesión. Por ejemplo, un usuario crea una passkey para un sitio web en Windows 10 y la almacena en un iPhone escaneando un código QR. Debido a que la clave se sincroniza a través de iCloud, el usuario puede iniciar sesión en el sitio web desde, por ejemplo, su MacBook. Las Passkeys también se pueden almacenar en candados de hardware, y el soporte para aplicaciones nativas ha llegado a iOS y Android.

A pesar de algunos problemas de usabilidad, por ejemplo, Bluetooth debe funcionar porque la proximidad del dispositivo se verifica cuando se escanea un código QR, vale la pena considerar passkeys. Sugerimos experimentar con ellas en passkeys.io para tener una idea de su uso..

43. Spin

Evaluar

Spin es una plataforma de código abierto para la construcción y ejecución de microservicios en WebAssembly (WASM). En anteriores ediciones del Radar, hemos hablado sobre WebAssembly en el contexto de los navegadores, pero ahora estamos presenciando su entrada por el lado del servidor debido a sus capacidades para sandboxing granular, interoperabilidad entre lenguajes y recargas en caliente. Con Spin CLI, puedes crear y distribuir rápidamente microservicios de WebAssembly en Rust, TypeScript, Python y TinyGo. Estamos emocionadas con Spin, y recomendamos que lo evalúes cuidadosamente a medida que sale del early preview.

44. Denodo como una herramienta principal de transformación de datos

Resistir

Denodo es una herramienta de virtualización de datos que apunta a hacer más fácil el proceso de exponer y asegurar datos transformados de un modo más amigable para el consumidor (desde múltiples fuentes de datos sobrepuestos y por una variedad de interfaces) desde una sola plataforma. La transformación de datos desde Denodo puede definirse al crear bases de datos virtuales y vistas usando un lenguaje semejante a SQL llamado VQL, que se ejecuta cuando el usuario hace consultas hacia la base de datos virtual. Internamente, Denodo puede delegar las consultas de las bases de datos virtuales en una o múltiples bases de datos superpuestas.

Pese a que Denodo hace fácil el proceso de comenzar a exponer datos de una forma amigable para el consumidor, su desempeño se degrada a medida de que las capas de visualizaciones y bases de datos virtuales se construyen una sobre otra y las consultas con múltiples uniones apuntan a múltiples bases de datos inferiores. Estos problemas se pueden resolver, pero requieren un conocimiento bastante profundo del comportamiento del producto y de las opciones de configuración de su rendimiento. Por estos inconvenientes y debido al limitado soporte que tiene para pruebas unitarias, recomendamos no utilizar Denodo como principal herramienta de transformación de datos y utilizar herramientas como Spark SQL (con dbt) para procesos de transformación de datos en su lugar.

Herramientas

Adoptar

45. DVC

Probar

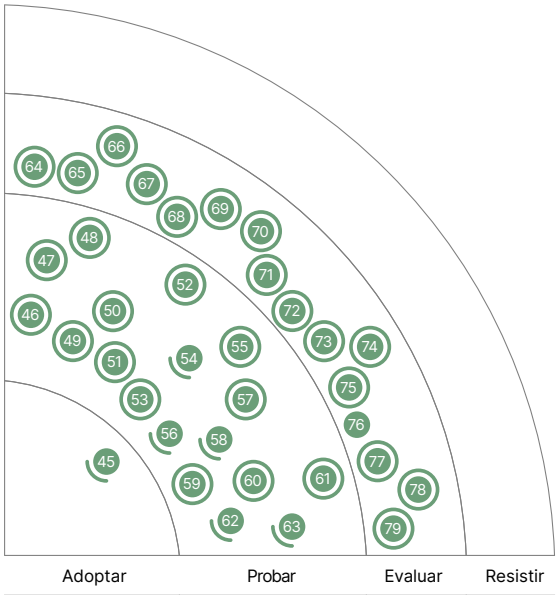
46. Akeyless
47. Apicurio Registry
48. Catálogo de eventos
49. FOSSA
50. Gitleaks
51. Helmfile
52. IBM Equal Access Accessibility Checker
53. Ktlint
54. Kubeflow
55. Mend SCA
56. Mozilla SOPS
57. Ruff
58. Soda Core
59. Steampipe
60. Terraform Cloud Operator
61. TruffleHog
62. Typesense
63. Vite

Evaluar

64. axe Linter
65. ChatGPT
66. DataFusion
67. Deepchecks
68. Herramientas de traducción de tokens de diseños
69. Devbox
70. Evidently
71. Giskard
72. GitHub Copilot
73. iamlive
74. Kepler
75. Agente de Secretos Externos de Kubernetes
76. Kubeshark
77. Obsidian
78. Ory Kratos
79. Ejecutor auto alojado para GitHub de Philips

Resistir

—



Nuevo

Desplazado adentro/afuera

Ningún cambio

45. DVC

Adoptar

DVC sigue siendo nuestra herramienta preferida para gestionar experimentos en proyectos de ciencia de datos. El hecho de que esté basado en Git lo convierte en un dominio conocido para que los desarrolladores lleven prácticas de ingeniería al ecosistema de ciencia de datos. La visión dogmática que tiene DVC de lo que es un un punto de control de un modelo encapsula cuidadosamente un conjunto de datos de entrenamiento, un conjunto de datos de prueba, hiperparámetros del modelo y el código. Convertir la reproducibilidad en una preocupación de primera clase, permite al equipo viajar en el tiempo a través de varias versiones del modelo. Nuestros equipos han utilizado con éxito DVC en producción para permitir la continuous delivery for ML (CD4ML); ya que se puede conectar con cualquier tipo de almacenamiento (incluyendo AWS S3, Google Cloud Storage, MinIO y Google Drive). Sin embargo, con conjuntos de datos cada vez más grandes, capturar instantáneas basadas en el sistema de archivos puede volverse particularmente costoso. Cuando los datos subyacentes cambian rápidamente, DVC al control de un buen almacenamiento versionado, permite realizar un seguimiento de las desviaciones del modelo durante un período de tiempo. Nuestros equipos han utilizado eficazmente DVC además de formatos de almacenamiento de datos como Delta Lake que optimiza el control de versiones (COW). La mayoría de nuestros equipos de ciencia de datos configuran DVC como una tarea del día cero mientras inician un proyecto; por este motivo, nos complace moverlo a la sección de Adoptar.

46. Akeyless

Probar

A medida que más organizaciones adoptan la computación en la nube, muchas están comenzando a integrar múltiples proveedores de nube simultáneamente para maximizar la flexibilidad y minimizar las dependencias. Sin embargo, el manejo de llaves y control de accesos a través de múltiples proveedores de la nube puede significar un desafío importante, lo que conduce a una mayor complejidad y riesgos de seguridad. Akeyless Es una plataforma centralizada, basada en la nube que provee la gestión unificada de secretos con una serie de ventajas para manejar secretos y datos sensibles. Se integra de forma fluida con diferentes proveedores, simplificando el manejo de secretos y control de accesos para monitorear y controlar quién tiene acceso a datos sensibles; con encriptación, control de accesos, autenticación multifactor y otros mecanismos de seguridad que aseguran que solo los usuarios autorizados sean capaces de acceder a datos sensibles. Adicionalmente, Akeyless provee una interfaz intuitiva para administración y monitoreo, proporcionando una experiencia de desarrollo y administración menos compleja y más escalable.

47. Apicurio Registry

Probar

Dentro de cualquier organización, los productores y consumidores de API necesitan estar sincronizados con los esquemas que serán usados para la comunicación entre ellos. Especialmente, a medida que en la organización crece el número de API y sus respectivos productores y consumidores; lo que pudo empezar simplemente compartiendo esquemas entre equipos, comenzará a presentar desafíos de escalamiento. Al encontrarse con este problema, algunos de nuestros equipos han recurrido a Apicurio Registry, un registro centralizado de código abierto para diversos tipos de esquemas y artefactos de API, incluyendo especificaciones de OpenAPI y esquemas de Protobuf y Avro. Apicurio Registry permite a los usuarios interactuar con éste a través de una interfaz de usuario,

así como una API REST y un plugin de Maven. También ofrece la opción de hacer cumplir restricciones de evolución de esquemas, tales como la retrocompatibilidad. Además, cuando se trata de trabajar con clientes de Kafka Apicurio Registry es compatible con el Confluent Schema Registry. Si bien la documentación del Confluent Schema Registry ha resultado más útil para nuestros equipos, el Apicurio Registry suple las necesidades de una fuente única de verdad para distintos esquemas.

48. Catálogo de eventos

Probar

--- Actualmente, las empresas suelen utilizar la transmisión de eventos como fuente de verdad y como un mecanismo de intercambio de información en las arquitecturas de microservices Esto crea la necesidad de estandarizar los tipos de eventos y compartir esos estándares en toda la empresa. Es habitual que se despliegue un registro de esquemas de eventos, pero la oferta existente tiende a estar especializada en un único servicio como Apache Kafka o Azure Event Hub. También suelen quedarse cortos a la hora de ofrecer una documentación rica sobre los tipos de eventos que vayan más allá de simples definiciones de esquema. EventCatalog es un proyecto de código abierto que ofrece algo que habitualmente vemos a las compañías construir ellas mismas: un repositorio plenamente accesible de documentación de eventos y esquemas. Éstos describen el rol que el evento juega en el negocio, a dónde pertenecen dentro de un modelo de dominio de negocio y qué servicios lo suscriben y publican. Si estás buscando una nueva manera de publicar documentación de eventos en tu organización, esta herramienta podría salvarte de la molestia de construirla tú mismo.

49. FOSSA

Probar

FOSSA es una herramienta de código abierto que ayuda a los desarrolladores y equipos a determinar en qué componentes de código abierto se basa su código y bajo qué licencias se publican estos componentes. Esta información es esencial para garantizar el cumplimiento de varias licencias de código abierto y mantener la Software Bill of Materials. FOSSA se integra con herramientas de gestión de dependencias de varios stacks tecnológicos para identificar qué componentes de código abierto se utilizan en un proyecto. También destaca cualquier problema de licencia basado en las políticas de la organización y genera informes de los mismos. Algunas características clave de FOSSA incluyen su capacidad para integrarse con flujos de desarrollo, como CI, y para realizar un seguimiento del cumplimiento en tiempo real. Muchos de nuestros clientes y equipos han encontrado en FOSSA una herramienta valiosa y eficaz.

50. Gitleaks

Probar

Gitleaks es una herramienta de línea de comandos SAST (static application security testing) de código abierto para detectar y prevenir secretos incrustados en el código (hardcoded) como pueden ser contraseñas, claves API o tokens en repositorios Git Se puede usar como hook de pre-commit en Git o en un pipeline de CI/CD. Nuestros equipos encontraron que Gitleaks es más sensible que otras herramientas de escaneo de secretos. Gitleaks utiliza expresiones regulares y codificación de cadenas de entropía para detectar secretos. Según nuestra experiencia, la flexibilidad para proporcionar expresiones regulares personalizadas junto con la codificación de entropía, permitió a los equipos categorizar mejor los secretos en función de sus necesidades. Por ejemplo, en lugar de categorizar todas las claves de API como “clave de API genérica”, permitió su categorización específica como “clave de proveedor de nube”.

51. Helmfile

Probar

Helmfile Es una herramienta de línea de comandos de código abierto y una especificación declarativa para administrar e instalar una colección de gráficos Helm Puede usarse para facilitar el control de versiones de los archivos de valores de Helm, los gráficos utilizados y otras personalizaciones. Permite flujos de trabajo de CI/CD con gráficos de Helm y ayuda a crear entornos reproducibles. Usamos Helmfile para administrar despliegues complejos con varias docenas de gráficos de Helm y descubrimos que simplifica el flujo de despliegue.

52. IBM Equal Access Accessibility Checker

Probar

Los defectos son más baratos de reparar cuando se detectan a tiempo. Es por eso que siempre intentamos entregar el feedback lo más rápido posible a los equipos de desarrollo en forma de análisis estático, pruebas unitarias o pruebas end-to-end que se ejecutan en el entorno local. La accesibilidad no es una excepción a esto y es por eso que hemos presentado herramientas como Lighthouse, axe-core y axe Linter en el pasado. Cuando se trata de probar automáticamente páginas web que ya están implementadas en producción, uno de nuestros equipos optó por utilizar IBM Equal Access Accessibility Checker en una comparación directa. Aunque todavía estamos en el proceso de evaluar los resultados, podemos decir que ofrece una forma eficiente de probar las páginas una vez desplegadas. Hacemos hincapié en que esto debe usarse para mejorar, no reemplazar, las primeras pruebas automatizadas por parte del equipo desarrollador. La herramienta se distribuye bajo una licencia Creative Commons y es de uso gratuito bajo esas restricciones.

53. Ktlint

Probar

El ecosistema Kotlin sigue evolucionando y nuestros equipos reportan experiencias positivas con Ktlint, un linter y formateador simple y fácil de configurar para código Kotlin. Nos gusta el formateo de código automatizado y dogmático ya que permite a los desarrolladores centrarse más en lo que hace el código que en cómo se ve; esta herramienta permite a los equipos de desarrollo mantener la consistencia y la legibilidad en sus bases de código de manera eficiente, reduciendo la probabilidad de merges desordenados debido a problemas de formato. Ktlint se puede configurar fácilmente para ejecutarse en hooks de pre-commit, apuntando solo a los archivos con cambios y dando como resultado procesos de integración más rápidos.

54. Kubeflow

Probar

Kubeflow es una plataforma de aprendizaje automático o machine learning (ML) nativa de Kubernetes- que simplifica los ciclos de vida de construcción, capacitación, y despliegue de modelos en diversas infraestructuras. Hemos utilizado ampliamente Pipelines para codificar flujos de ML para varios modelos a través de casos de uso de experimentación, entrenamiento, y servicio. Además de Pipelines, Kubeflow incluye múltiples componentes, entre los que encontramos muy útil el ajuste de hiperparámetros con Katib y multi-tenancy to be quite useful.

55. Mend SCA

Probar

Mend SCA (software composition analysis), anteriormente Whitesource, ayuda a detectar dependencias de software de código abierto identificando si están actualizadas, contienen fallas de seguridad o tienen requisitos de licenciamiento. Nuestros equipos han tenido una buena experiencia con la integración de Mend SCA en los procesos a producción. Desde la integración con el IDE, generando un PR automático basado en un problema identificado hasta la integración en con la CI/CD, esta herramienta ofrece una excelente experiencia para desarrolladores. Otras herramientas SCA populares, como Snyk, son comparables y también vale la pena explorarlas para sus necesidades de seguridad.

56. Mozilla SOPS

Probar

Nuestro consejo en lo que respecta a la gestión de secretos siempre ha sido desvincularlo del código original. Sin embargo, los equipos se enfrentan a menudo a la disyuntiva entre la automatización total (con el enfoque de infraestructura como código) y unos pocos pasos manuales (utilizando herramientas como las cajas fuertes) para gestionar, seleccionear y rotar secretos semilla. Por ejemplo, nuestros equipos utilizan SOPS para gestionar las credenciales semilla para arrancar la infraestructura. En algunas situaciones, sin embargo, es imposible eliminar los secretos de repositorios de código heredado. Para tales necesidades, encontramos que Mozilla SOPS es una buena opción para cifrar secretos en archivos de texto. SOPS se integra con almacenes de claves gestionados en la nube como AWS y GCP Key Management Service (KMS) o Azure Key Vault como fuentes de claves de cifrado. También funciona multiplataforma y admite claves PGP.

57. Ruff

Probar

Ruff es un nuevo linter para Python. Para nosotras, la pregunta no es, si usar un linter o no, sino qué linter usar, y en Python hay distintas opciones. Ruff destaca por dos razones: su experiencia “listo para usar” y su velocidad. Tiene más de 500 reglas integradas y reemplaza fácilmente a Flake8, ya que incluye muchos de sus plug-ins. Las afirmaciones del equipo detrás de Ruff sobre su rendimiento se confirman con nuestra experiencia. Realmente, Ruff es al menos en orden de magnitud, más rápido que otros linters, lo cual es un beneficio enorme al ayudar a reducir el tiempo de compilación en grandes bases de código.

58. Soda Core

Probar

Soda Core Es una herramienta de observabilidad y calidad de datos de código abierto. Nuestros equipos la han utilizado para validar datos a medida que llegan a un sistema, antes y después de las transformaciones, y configurar controles de seguimiento automatizados para detectar anomalías. Estamos contentos con SodaCL, el DSL para escribir verificaciones de datos en Soda Core, ya que ayuda a otros miembros del equipo, más allá de los ingenieros de datos, a escribir controles de calidad. En general, nuestra experiencia utilizando Soda Core para encontrar y resolver problemas de datos a escala ha sido positiva.

59. Steampipe

Probar

Steampipe es una herramienta de código abierto que permite realizar consultas instantáneas mediante SQL a servicios en la nube como AWS, Azure y GCP. Con más de 100 plugins y soporte integrado para la creación de dashboards, Steampipe hace trivial conectar datos de configuración de la nube en vivo con conjuntos de datos internos o externos, así como la creación de dashboards de seguridad o conformidad. Hemos disfrutado trabajando con Steampipe y creando varios de estos dashboards mediante configuraciones en la nube de AWS.

60. Terraform Cloud Operator

Probar

Más y más equipos están usando el patrón de Kubernetes Operators para manejar sus clusters de Kubernetes. Antes recomendábamos Crossplane para esto, y ahora tenemos una herramienta alternativa, Terraform Cloud Operator para Kubernetes. Esta herramienta integra Terraform Cloud y Kubernetes al extender el plano de control de Kubernetes para habilitar el manejo del ciclo de vida de infraestructuras en la nube y en las premisas a través de manifiestos de Kubernetes. Nuestro equipo lo utiliza para proveer recursos de namespaces de Kubernetes y RoleBindings a instancias de bases de datos en la nube y otros recursos SaaS. Nos gusta mucho porque aprovecha el módulo de Terraform, que es una capa de abstracción más familiar para operar sobre recursos en la nube.

61. TruffleHog

Probar

TruffleHog es una herramienta de SAST (pruebas estáticas de seguridad de aplicaciones, por sus siglas en inglés) de código abierto para detectar secretos de diversas fuentes. Si bien repositorios de GitHub y GitLab son el caso de uso más popular, TruffleHog también se puede usar para escanear repositorios en la nube como S3 y GCS, ficheros y directorios locales y logs de CircleCI. Las personas desarrolladoras pueden configurar TruffleHog como un hook pre-commit o escanear el historial de los repositorios existentes en toda una organización de GitHub para detectar secretos. La herramienta permite detectar patrones personalizados de expresiones regulares, lo cual ha resultado ser bastante útil incluso estando en etapa alfa. TruffleHog también tiene una versión empresarial, pero para nuestras personas desarrolladoras la versión de código abierto ha resultado fácil de configurar y suficiente para los casos de uso más comunes. La herramienta tiene una comunidad muy activa que añade nuevas funcionalidades con regularidad.

62. Typesense

Probar

Typesense es un buscador de código abierto, tolerante a errores de tipeo, optimizado para experiencias de búsqueda de baja latencia y alto rendimiento. Si estás construyendo una aplicación de búsqueda en que la latencia es crítica con un tamaño de índice de búsqueda que pueda caber en memoria, Typesense es una alternativa poderosa. Nuestros equipos utilizan Typesense en clústeres multi-nodo de alta disponibilidad para distribuir la carga de trabajo y garantizar que la infraestructura crítica para la búsquedas sea resiliente. Los equipos tuvieron una buena experiencia con Typesense en producción, por lo que lo hemos pasado a Probar.

63. Vite

Probar

Vite, una herramienta de compilación para el frontend, ha seguido madurando y creciendo en popularidad desde que la presentamos en el anillo Evaluar en el Radar anterior. Se está convirtiendo rápidamente en la opción predeterminada entre nuestros equipos al comenzar un nuevo proyecto frontend. Vite proporciona un conjunto de valores predeterminados para construir, empaquetar y administrar dependencias en aplicaciones basadas en módulos ES para el navegador. Dado que aprovecha la velocidad nativa de esbuild y el empaquetador Rollup, Vite mejora significativamente la experiencia de desarrollo frontend. Además, cuando es usado con React, Vite ofrece una atractiva alternativa al incondicional, pero casi extinto Create React App. Vite se basa en módulos ES y, a diferencia de otras herramientas más antiguas, no proporciona reemplazo de código no soportado en navegadores antiguos como lo hacen librerias de shimming y polyfilling, lo que significa que se requiere de una estrategia diferente para navegadores que no soportan módulos ES. En estos casos, algunos de nuestros equipos importan polyfills a nivel de módulo ES para que Vite pueda de esa manera ser consistente en todos los entornos.

64. axe Linter

Evaluar

Cada vez es más fácil para los desarrolladores detectar problemas de accesibilidad en las primeras etapas del proceso de desarrollo. Mientras que herramientas como axe-core escanean el código en busca de problemas de accesibilidad en el pipeline, la extensión de VSCode axe Linter ayuda a encontrarlos incluso antes de eso, mientras se escribe el código. La gran mayoría de los problemas de accesibilidad se clasifican en categorías que podrían evitarse mediante pruebas automatizadas y el uso de linters de retroalimentación en tiempo real como éste.

65. ChatGPT

Evaluar

ChatGPT es una herramienta interesante que tiene el potencial de ser de gran utilidad en varios aspectos del proceso de creación de software. Como un Gran Modelo de Lenguaje (LLM por sus siglas en inglés) que ha "leído" miles de millones de páginas web, ChatGPT propone perspectivas nuevas y ayuda en diferentes tareas, desde generar ideas y requisitos hasta escribir código y pruebas. Su habilidad para trabajar en las múltiples partes del ciclo de desarrollo del software la convierte en una herramienta versátil que puede mejorar la eficiencia y reducir los errores durante el proceso de desarrollo. GPT4, el modelo de lenguaje grande que alimenta ChatGPT, tiene ahora la habilidad de integrarse con herramientas externas como repositorios de gestión del conocimiento, entornos aislados de código o búsquedas web. Por ahora, creemos que ChatGPT se usa mejor como el punto de partida de un proceso, por ejemplo, para ayudar a escribir un primer borrador de una historia o un esqueleto para una tarea de codificación, que para producir resultados completamente "horneados".

Existen preocupaciones alrededor de la propiedad intelectual y la privacidad de los datos con estas herramientas de IA, incluyendo algunas cuestiones legales no resueltas, por lo que recomendamos a las organizaciones que se dejen asesorar por sus equipos legales antes de usarlo. Algunos de nuestros clientes ya han empezado a experimentar con ChatGPT en varias fases del ciclo de vida del software, y recomendamos explorar la herramienta para valorar sus posibles beneficios. Esperamos que, como GitHub Copilot, pronto dispondrá de una oferta "para empresas" que pueda mitigar dichas preocupaciones sobre la propiedad intelectual.

66. DataFusion

Evaluar

DataFusion es parte de la exploración llevada a cabo por parte de la comunidad de datos de Rust sobre el rendimiento, seguridad de la memoria y funciones de concurrencia aplicadas al procesamiento de datos. Comparte similitudes con Polars, como son una familiar API DataFrame en Rust (con enlaces de Python), el uso de Apache Arrow bajo el capó y compatibilidad con SQL. Si bien está diseñado principalmente para la ejecución de un solo proceso, el soporte de procesamiento distribuido está en desarrollo dentro de Ballista. Creemos que las bibliotecas de Rust para el procesamiento de datos son un espacio en evolución que vale la pena seguir y explorar, y DataFusion es parte de él.

67. Deepchecks

Evaluar

A medida que el aprendizaje automático se abre camino en la comunidad, las prácticas están creciendo en torno a modelos de pruebas automatizadas, validación de los datos de entrenamiento y modelos de observabilidad del rendimiento en producción. Cada vez más, estas verificaciones automatizadas están siendo integradas en procesos de entrega continua o se ejecutan contra modelos de producción para medir el rendimiento de dichos modelos y detectar desviaciones. Han surgido algunas herramientas con capacidades parecidas o que se superponen para gestionar distintos pasos de este proceso (Giskard y Evidently ambién se han incluido en este volumen). Deepchecks es otra de estas herramientas que está disponible como biblioteca Python de código abierto y puede ser invocada desde el código del pipeline a través de un amplio conjunto de APIs. Una característica exclusiva de esta herramienta es su capacidad para manejar datos tanto tabulares como de imágenes, con un módulo para datos lingüísticos que actualmente se encuentra en fase alfa. Por el momento, ninguna herramienta puede gestionar por sí sola la variedad de pruebas y controles de seguridad de todo el proceso de ML. Recomendamos evaluar Deepchecks para su nicho de aplicación en particular.

68. Herramientas de traducción de tokens de diseños

Evaluar

Los tokens de diseño son un mecanismo útil para definir elementos estándar en sistemas de diseño. Pero mantener dichos elementos de diseño consistentes a lo largo de diferentes medios como aplicaciones móviles o frameworks web es una tarea que se vuelve cada vez más formidable. Las herramientas de traducción de tokens de diseño simplifican este problema organizando y automatizando la transformación desde la descripción del token (en YAML o JSON) al código que realmente controla su representación en un determinado medio como CSS, componentes de React o HTML. Style Dictionary es un ejemplo open-source ampliamente usado y que se integra bien en pipelines de compilación, pero también hay alternativas comerciales como Specify.

69. Devbox

Evaluar

Devbox proporciona una interfaz accesible para crear entornos de desarrollo reproducibles por proyecto aprovechando el administrador de paquetes Nix. Nuestros equipos lo utilizan para eliminar los conflictos de versión y configuración en sus entornos de desarrollo, y les gusta por su facilidad de uso. Devbox admite shell hooks, scripts personalizados y generación de devcontainer.json para la integración con VSCode.

70. Evidently

Evaluar

Evidently es una herramienta open source basada en Python para ayudar al monitoreo de construcción de modelos de machine learning que garanticen su calidad y su estabilidad en operaciones en producción. Puede ser utilizada en varias etapas del ciclo de vida del modelo: como un panel para revisar el modelo en un notebook, como parte de un pipeline o como un servicio de monitoreo después del despliegue. Con un foco particular en detectar la deriva del modelo, Evidently también ofrece funcionalidades como calidad del modelo, inspección de la calidad del dato y la detección de desvíos del modelo (model drift). Adicionalmente, tiene incorporadas muchas métricas, visualizaciones asociadas y tests que pueden ser fácilmente combinados en un informe, panel o un pipeline orientado a tests.

71. Giskard

Evaluar

Giskard Es una herramienta de código abierto diseñada para ayudar a las organizaciones a crear modelos de IA más sólidos y éticos; al proporcionar funcionalidades para garantizar la calidad haciendo foco en la explicabilidad y la equidad. Facilita la cooperación entre las partes interesadas, tanto técnicas como no técnicas, lo que les permite evaluar modelos de forma colaborativa y establecer criterios de aceptación basados en la prevención de sesgos y otras métricas esenciales de calidad. Giskard garantiza que los resultados del modelo estén mejor alineados con los objetivos comerciales y ayuda a resolver problemas de calidad antes del despliegue en producción.

72. GitHub Copilot

Evaluar

GitHub Copilot es un asistente de inteligencia artificial (IA) para código, fue creado de manera colaborativa entre Microsoft y OpenAI. Utiliza modelos de ML o aprendizaje automático para generar sugerencias basadas en el contexto en el que se encuentra trabajando la persona desarrolladora. Cuenta con una sólida integración con IDEs y utiliza una base de código existente y un contexto de editor para crear sugerencias. A pesar de ser catalogado como “tu pareja de programación de IA”, no podemos decir que hace “pairing”, probablemente lo describiríamos como una especie de Stack Overflow sobrecargado y sensible al contexto. Cuando predice correctamente lo que un desarrollador está tratando de hacer, puede ser una herramienta poderosa para hacer las cosas. Sin embargo, como todas las IA basadas en modelos de lenguaje de gran tamaño (Large Language Models), tiene una tendencia a alucinar con el uso de API posibles pero inexistentes y puede introducir errores a través de algoritmos ligeramente defectuosos. Hemos tenido éxito en la generación de código a nivel de línea, bloque y método, así como en la creación de pruebas o configuraciones de infraestructura. Curiosamente, funciona mejor cuando utilizas buenas prácticas de nombres, por lo que promueve un código más comprensible.

Las capacidades de las herramientas de IA avanzan rápidamente y creemos que es sensato que las organizaciones las prueben. Algunos argumentos de venta a favor de Copilot afirman ganancias en eficiencia muy altas, pero seguimos escépticos: después de todo, escribir código no es lo único a lo que los desarrolladores dedican su tiempo, y es claramente difícil medir la productividad del desarrollador en primer lugar. Dicho esto, Copilot es una herramienta bastante económica; si ofrece alguna ganancia de productividad, probablemente valga la pena. Copilot X -en versión preliminar al momento de escribir este texto-, ofrece funcionalidades e integración adicionales dentro de un flujo de creación de software. Copilot tiene una oferta “para empresas” que provee más claridad sobre los problemas de propiedad intelectual, así como la capacidad de administrar las funciones de las

herramientas de forma centralizada en toda la organización. Creemos que estas características son críticas para su adopción empresarial.

73. iamlive

Evaluar

Crear justo las mínimas politicas AWS IAM viables que necesitamos, de acuerdo con el principio de mínimo privilegio, puede ser un largo camino de ensayo y error. iamlive puede acortar ese camino considerablemente, ya que monitorea las llamadas de AWS CLI hechas desde una máquina y determina las políticas necesarias para ejecutar dichas llamadas. La herramienta genera un documento de políticas con declaraciones, acciones, principios, y recursos que pueden ser usados como un buen punto de partida. Hemos encontrado que es particularmente útil para crear las políticas necesarias en las pipelines CI/CD que aprovisionan la infraestructura, reduciendo las típicas idas y venidas que se dan tras un fallo de Terraform cuando una política de roles IAM es insuficiente.

74. Kepler

Evaluar

Medir el consumo de energía es un paso importante para que los equipos reduzcan la huella de carbono de su software. Cloud Carbon Footprint (CCF) estima la energía en base a los datos de facturación y uso recuperados de las APIs de la nube. Kepler — acrónimo de Kubernetes-based Efficient Power Level Exporter: Exportador de niveles de energía eficiente basado en Kubernetes) — va un paso más allá: usa contadores de software por medio de RAPL, ACPI y nvml to measure power consumption by hardware resources and employs an eBPF-para atribuir el consumo de energía a procesos, containers y pods de Kubernetes. El consumo de energía luego se convierte en estimaciones de energía usando un modelo de ML personalizado y datos provenientes de SPEC Power benchmark. Finalmente, los reportes de consumo de energía a nivel de pods están disponibles como métricas de Prometheus. En los casos en que Kubernetes se ejecuta en máquinas virtuales, por ejemplo, cuando no se usan instancias físicas, Kepler usa cgroups para estimar el consumo de energía. Tenemos gran experiencia con CCF y podemos dar constancia de su utilidad, pero estamos intrigados por el enfoque que le ha dado el proyecto Kepler.

75. Agente de Secretos Externos de Kubernetes

Evaluar

El Kubernetes External Secrets Operator permite a proveedores de secretos externos integrarse con Kubernetes. Lee de la API del proveedor externo e inyecta el resultado en un Secreto de Kubernetes. El agente trabaja con una amplia variedad de herramientas de gestión de secretos, incluyendo algunos que hemos incluido en ediciones previas del Radar. Nuestros equipos han encontrado una gran simplificación en el uso de los secretos cuando trabajan con Kubernetes al permitir el uso de un único almacenamiento para todo el proyecto.

76. Kubeshark

Evaluar

Kubeshark es un visualizador de tráfico de APIs para Kubernetes. Hasta Noviembre del 2022 era conocido como Mizu. A diferencia de otras herramientas, Kubeshark no requiere instrumentación o cambios en el código. Se ejecuta como un DaemonSet para inyectar un container a nivel de nodo en el clúster de Kubernetes y realizar operaciones como tcpdump. La encontramos una herramienta útil para debuggear, ya que puede observar todas las comunicaciones dentro de la API a través de múltiples protocolos (REST, gRPC, Kafka, AMQP y Redis) en tiempo real.

77. Obsidian

Evaluar

La gestión del conocimiento es crítica para las tecnologistas, ya que necesitamos estar aprendiendo y actualizándonos constantemente con lo último en tecnología. Recientemente, algunas herramientas como Obsidian y Logseq han aparecido en la categoría de herramientas para tomar notas que permiten enlazarlas para formar un grafo de conocimiento, al tiempo que las almacenan en archivos markdown planos en un directorio local, permitiendo así que los usuarios sean dueños de su data. Estas herramientas ayudan a los usuarios a organizar y enlazar sus notas de manera flexible y no lineal.

Obsidian tiene un repositorio rico en plugins de la comunidad. Algunos que han captado nuestra atención particularmente, son Canvas, que es parecido a tener una versión local de Miro o Mural, y Dataview, que efectivamente trata tus notas como una base de datos y ofrece un lenguaje de consulta para filtrar, ordenar y extraer datos de ellas.

78. Ory Kratos

Evaluar

Ya hemos evaluado Ory Hydra como una solución OAuth2 “self-hosted”, y los comentarios de los equipos han sido buenos. Esta vez, nos volvemos hacia Ory Kratos, un sistema de gestión de identidades y usuarios basado fundamentalmente en un API que es amigable para el equipo desarrollador y fácilmente personalizable. Proporciona funciones comunes que querríamos tener en un sistema de gestión de identidades, incluyendo autoservicio de login y registro, autenticación multifactor (MFA/2FA), verificación y recuperación de cuenta. Al igual que Hydra, Kratos carece de interfaz de usuario y precisa que los equipos de desarrollo construyan su propia IU, lo que proporciona mayor flexibilidad al equipo. Los equipos de desarrollo también pueden personalizar el esquema de identidad para ajustarlo a diferentes contextos de negocio. Kratos no tiene dependencias externas aparte de la base de datos, y es fácil de desplegar y escalar en diferentes entornos de cloud. Si tienes que construir un sistema de gestión de usuarios, te recomendamos que pruebes Kratos.

79. Ejecutor auto alojado para GitHub de Philips

Evaluar

Aunque los ejecutores de GitHub Actions rcubren una amplia gama de los entornos de ejecución más comunes, a veces se necesita algo más específico para casos particulares, como un entorno de ejecución de un lenguaje menos común o o una configuración de hardware específica. En estas situaciones funciona mejor un ejecutor auto alojado. El ejecutor auto alojado para GitHub de Philips es un módulo de Terraform que permite levantar ejecutores personalizados en instancias spot de AWS EC2. Al auto alojar ejecutores se pierde parte de la gestión del ciclo de vida de GitHub Actions, por lo que el módulo crea una serie de Lambdas para contrarrestarlo. También se encarga de escalar los ejecutores según necesidad, lo cual contribuye a gestionar costos y permite que los ejecutores sean efímeros, una buena práctica que mejora la repetibilidad y la seguridad. Al auto alojar ejecutores hay muchas cosas que se pueden pasar por alto si se construye de cero, por lo que herramientas como ésta pueden facilitar la tarea.

Lenguajes y Frameworks

Adoptar

80. Gradle Kotlin DSL
81. PyTorch

Probar

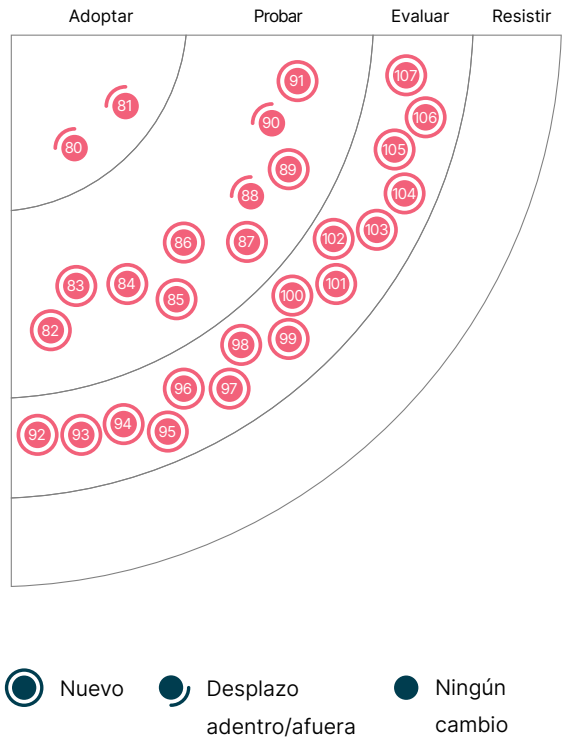
82. dbt-unit-testing
83. Jetpack CameraViewfinder
84. Jetpack DataStore
85. Mikro ORM
86. Preferencia de idioma por aplicación
87. Quarto
88. River
89. Stencil
90. Synthetic Data Vault
91. Vitest

Evaluar

92. .NET 7 Native AOT
93. .NET MAUI
94. dbt-expectations
95. Directus
96. Ferrocene
97. Flutter para sistemas embebidos
98. Fugue
99. Galacean Engine
100. LangChain
101. mljar-supervised
102. nanoGPT
103. pandera
104. Qwik
105. SolidJS
106. Turborepo
107. API para dispositivos WebXR

Resistir

—



80. Gradle Kotlin DSL

Adoptar

Nuestro equipo ve ahora a Gradle Kotlin DSL como tecnología por defecto para iniciar nuevos proyectos usando Gradle, preferentemente sobre Groovy. Equipos que ya estén utilizando Groovy deberían considerar migrar. Kotlin provee mejor soporte para refactorización y edición más simple en IDEs, y nuestros equipos reportan que produce código que es más fácil de leer y mantener. Dado que algunos IDEs ahora soportan la migración, debería ser relativamente rápido experimentar con la sustitución del Groovy existente. En algunas situaciones Kotlin puede ser más lento que Groovy; sin embargo, para muchos proyectos, es poco probable que impacte al equipo.

81. PyTorch

Adoptar

PyTorch sigue siendo nuestra elección de framework para machine learning (ML). La mayoría de nuestros equipos prefieren PyTorch a TensorFlow. PPyTorch expone el funcionamiento interno de ML que TensorFlow oculta, haciéndolo más sencillo de depurar. Con gráficos computacionales dinámicos, la optimización de modelos es mucho más fácil en comparación con cualquier otro framework de ML. La amplia disponibilidad de modelos de última generación State-of-the-Art (SOTA) models por sus siglas en inglés) y la facilidad para implementar documentos de investigación hacen que PyTorch destaque. Cuando se trata de ML gráfico, PyTorch Geometric es un ecosistema más maduro y nuestros equipos han tenido excelentes experiencias con él. PyTorch también ha ido cerrando las brechas en lo que respecta a la implementación y escalado de modelos; nuestros equipos han usado con éxito, por ejemplo, TorchServe para desplegar en producción modelos previamente entrenados. Dado que muchos equipos utilizan PyTorch de forma predeterminada para sus necesidades de deep-learning de extremo a extremo, con gusto recomendamos adoptar PyTorch.

82. dbt-unit-testing

Probar

dbt-unit-testing es un paquete dbt que permite escribir pruebas unitarias para un modelo y su lógica creando mocks de sus dependencias. Esto aporta rigurosidad a la ingeniería para obtener un feedback rápido durante el desarrollo al ecosistema de datos. Nuestros equipos usan este paquete con Snowflake para practicar el desarrollo guiado por pruebas (TDD), aunque solo fue factible para transformaciones simples. La biblioteca ciertamente tiene algunas dificultades en cuanto a la depuración de pruebas fallidas, pero la capacidad de escribir pruebas unitarias a transformadores a medida que desarrollamos el modelo proporcionó una limpia experiencia de desarrollo.

83. Jetpack CameraViewfinder

Probar

Cuando se añaden capacidades de uso de cámara a las aplicaciones Android, el equipo desarrollador debe estar atento a problemas. La recientemente estrenada API de Jetpack CameraViewfinder mejora significativamente la experiencia de desarrollo en este aspecto. Internamente éste utiliza un TextureView (vista de textura) o un SurfaceView (vista de superficie) para mostrar la imagen que entrega la cámara y aplicar transformaciones que muestran el visor de la cámara, arregla aspecto de ratio, escala y rotación de ser necesario. También se proveen fondos optimizados para aparatos que se doblan. Pese a que no es una característica mayor, lo destacamos para asegurarnos de que los equipos estén al tanto de su existencia.

84. Jetpack DataStore

Probar

Jetpack DataStoreEs una nueva solución de almacenamiento de datos para almacenar datos de forma asíncrona, consistente y transaccional. Tiene dos implementaciones: Preferences DataStore para pares clave-valor sin tipado y Proto DataStore para tipos de datos complejos usando Protobufs. Por defecto se utiliza con Kotlin coroutines y Flow pero hay disponible soporte adicional para RXJava 2 y 3. La documentación recomienda que consideres migrar a DataStore si actualmente estás usando SharedPreferences, y estamos de acuerdo con esa recomendación.

85. Mikro ORM

Probar

Mikro ORM es un framework de mapeo objeto-relacional (ORM) que tiene un interesante enfoque centrado en TypeScript-Al utilizar TypeScript en todo el stack, genera una experiencia de desarrollo consistente desde el navegador hasta el backend, lo que facilita a los desarrolladores escribir y mantener el código. En particular, el rendimiento de Mikro ORM es excelente, lo que permite una ejecución rápida de consultas y minimiza la latencia. Si bien Mikro ORM ofrece características atractivas, es esencial tener en cuenta las limitaciones generales asociadas con los mapeadores relacionales de objetos. Los frameworks ORM son a menudo complejos y ofrecen solo una abstracción propensa a filtraciones sobre un almacén de datos relacionales, por lo que usar uno siempre es un balance de compromisos.

86. Preferencia de idioma por aplicación

Probar

Mucha gente habla más de un idioma y usa idiomas distintos en diferentes contextos. Los dispositivos y plataformas que pueden ejecutar aplicaciones piden habitualmente a los usuarios seleccionar un idioma para el sistema y luego hacen que las aplicaciones usen ese idioma. En teléfonos móviles, particularmente, las personas usuarias pueden preferir que algunas aplicaciones usen otro idioma distinto al especificado por el sistema; Apple introdujo una opción de idioma por aplicación en iOS hace algún tiempo. Los equipos de desarrollo de aplicaciones Android, sin embargo, han tenido que implementar soluciones propias dentro de sus aplicaciones si querían proporcionar esta opción - hasta ahora. Android 13 introduce una nueva opción del sistema, preferencia de idioma por aplicación, y una API pública, haciendo fácil para las personas desarrolladoras ofrecer esta funcionalidad. Para mantener la retrocompatibilidad, APIs equivalentes están disponibles en AndroidX mediante AppCompatDelegate. Animamos a los equipos de desarrollo a reemplazar sus soluciones propias y usar esta nueva funcionalidad en sus aplicaciones.

87. Quarto

Probar

Quarto es un sistema de publicación científica y técnica de código abierto. Con él, podemos crear notebooks computacionales que permiten escribir documentos en markdown, incrustar código y emitir el resultado de ese código en el documento final. Se puede usar para crear informes de análisis de datos reproducibles y personalizables, los cuales se pueden compartir fácilmente en una variedad de formatos. Nuestros equipos de ciencia de datos utilizaron Quarto para compartir informes de análisis de datos que contienen visualizaciones (plots) y tablas. Les gustó poder usar R y Python para generar estos informes dinámicos y luego exportarlos como HTML t para compartirlos con las partes interesadas. Si estás buscando compartir tu investigación y análisis dentro o fuera de tu organización, te recomendamos que evalúes Quarto.

88. River

Probar

En el corazón de muchos enfoques de machine learning se encuentra la creación de modelos partiendo de un conjunto de datos de entrenamiento. Una vez creado el modelo, puede ser usado una y otra vez. A pesar de ello, el mundo continúa girando y a menudo el modelo necesita cambiar a medida que se dispone de nuevos datos. Volver a ejecutar el paso de creación del modelo simplemente puede ser lento y costoso. El aprendizaje incremental aborda este problema, haciendo posible aprender incrementalmente de streams de datos para reaccionar rápido a los cambios. Como beneficio adicional, los requisitos de cómputo y memoria son menores y más predecibles. Nuestra experiencia práctica con River continúa siendo positiva. Vowpal Wabbit, que puede ser una alternativa, tiene una curva de aprendizaje mucho más pronunciada, y el API tipo Scikit ofrecida por River hace que River sea más accesible a los equipos de ciencia de datos.

89. Stencil

Probar

Stencil es una librería que permite a los desarrolladores crear componentes web reutilizables utilizando herramientas bien establecidas como TypeScript, JSX y JSDoc. Según las experiencias de nuestros equipos, Stencil.js es una muy buena opción para crear sistemas de diseño independientes de la plataforma. Para los pocos navegadores que no son compatibles con las funciones de los navegadores modernos, Stencil.js garantiza la compatibilidad mediante el reemplazo de funciones y APIs no compatibles a demanda.

90. Synthetic Data Vault

Trial

Synthetic Data Vault (SDV) es un ecosistema de bibliotecas para generación de datos sintéticos que pueden aprender la distribución de un conjunto de datos para generar datos sintéticos con el mismo formato y las mismas propiedades estadísticas que la fuente. En el pasado, hablamos sobre las desventajas de usar datos reales en ambientes de pruebas. Sin embargo, los matices en la distribución de datos en producción difícilmente pueden ser replicadas manualmente, trayendo como resultado sorpresas y defectos. Hemos tenido buenas experiencias utilizando SDV para generar grandes volúmenes de datos para pruebas de rendimiento. SDV se comporta bien con el modelado de una sola tabla. Sin embargo, los tiempos de generación de datos crecen considerablemente en la medida que el número de tablas con restricciones de claves foráneas aumentan. Aún así, SDV ofrece una gran ventaja para pruebas de rendimiento local. Es una buena herramienta para la generación de datos sintéticos y vale la pena su consideración para distintas necesidades de pruebas.

91. Vitest

Probar

Vitest es un framework de pruebas unitarias para Javascript. Hasta ahora, muchos equipos confiaban en Jest, pero Jest no se lleva bien con Vite, una herramienta moderna para desarrollar front-end. El uso conjunto de Jest y Vite forzaba a los equipos el uso de dos flujos de trabajo — una para compilación y desarrollo y otra para pruebas unitarias — lo que requería una configuración tediosa de los flujos de trabajo con configuraciones duplicadas. Estos problemas se solucionan con Vitest. Está diseñado específicamente para Vite y utiliza Vite como paquete. Como característica adicional, Vitest tiene APIs compatibles con Jest, lo que hace posible usar Vitest como reemplazo directo de Jest en varias configuraciones de compilación. Sin embargo, usar Vite y Vitest juntos brinda una mejor experiencia de desarrollo y, aunque Vitest es rápido, según nuestra experiencia, no es necesariamente más rápido que usar Jest.

92. .NET 7 Native AOT

Evaluar

.NET 7 Native AOT es un gran paso adelante en una larga línea de enfoques para desplegar Aplicaciones .NET de forma nativa. Elimina por completo IL y JIT en tiempo de ejecución. Introducida en .NET 7, esta mejora es particularmente significativa para ejecutar aplicaciones .NET en funciones serverless. Esta nueva opción de despliegue elimina el problema del arranque en frío, que ha sido un problema persistente para .NET en plataformas sin servidor como AWS Lambda y Azure Functions. Con Native AOT, puede generar un binario desplegable más pequeño que los métodos anteriores, lo que resulta en tiempos de arranque en frío más rápidos. AWS ha adoptado oficialmente Native AOT, apoyándolo con sus Amazon Lambda Tools. Esta nueva opción de implementación pone a .NET 7 a la par con TypeScript/JavaScript en términos de tiempos de arranque en frío, lo que la convierte en una opción viable para organizaciones con una infraestructura orientada en gran medida a .NET.

93. .NET MAUI

Evaluar

.NET MAUI es un nuevo framework multiplataforma para crear aplicaciones móviles nativas y de escritorio con C# y XAML. Permite la creación de aplicaciones que pueden ejecutarse en Android, iOS, macOS y Windows a partir de un solo código base. Sin embargo, como nueva tecnología, el ecosistema alrededor de MAUI aún no es tan maduro como el de React Native u otros sistemas multiplataforma, y sólo soporta C#. Adicionalmente, MAUI puede tener que encarar los desafíos que experimentan las organizaciones que han usado Xamarin en el pasado, como escasas herramientas multiplataforma, problemas de integración en dispositivos móviles, disponibilidad de desarrolladoras y un ecosistema inmaduro.

Aunque Microsoft anunció su compromiso con MAUI como un framework de desarrollo de código abierto y orientado principalmente a móviles, su éxito aún está por demostrarse. Si ya estás usando Xamarin, podría considerar MAUI como una mejora potencial. No obstante, si C# o Xamarin aún no forman parte de sus herramientas, sería prudente acercarse a MAUI con cierta cautela hasta que la tecnología sea ampliamente adoptada y probada en el mercado.

94. dbt-expectations

Evaluar

dbt-expectations es un paquete de extensión para dbt inspirado en Great Expectations. La calidad de los datos es un principio importante de la gobernanza de datos, por lo que, cuando se trata de gobernanza de datos automatizada es importante crear controles integrados que marquen anomalías o problemas de calidad en los pipelines de datos. Al igual que las pruebas unitarias se ejecutan en un pipeline de compilación, dbt-expectations realiza aserciones durante la ejecución de un pipeline de datos. En el mundo de dbt, puede ejecutar pruebas de calidad de datos al estilo de Great Expectations en su almacén directamente dentro de dbt. Nuestros equipos han estado explorando esto, y tenía sentido resaltarlo.

95. Directus

Evaluar

Hemos utilizado Directus como un sistema de gestión de contenidos (CMS) headless. Aunque tenemos opciones en lo que se refiere a productos CMS headless, necesitábamos una solución auto alojada con una gestión enriquecida de activos digitales y flujos de autoría de contenidos. En esta evaluación encontramos que Directus se ajusta bien a nuestras necesidades; nos gusta bastante su procesamiento de datos basado en eventos y la automatización a través de flujos.

96. Ferrocene

Evaluar

El lenguaje Rust ha estado ganando popularidad en los últimos años por sus características de seguridad, rendimiento y concurrencia. Sin embargo, han faltado herramientas Rust certificadas para aplicaciones en mercados donde la seguridad es crítica como es el caso de la industria automotriz. Este vacío está siendo cubierto por Ferrocene, una herramienta de compilador de Rust. Ferrocene promete cumplir con el estándar de seguridad funcional ISO26262 para sistemas electrónicos en vehículos de carretera; y los esfuerzos para calificar este lenguaje y sus herramientas para su uso en estos dominios ya está en marcha. Estamos emocionados por su progreso y la disponibilidad de este tipo de herramientas que cumplan con la seguridad acelerará sin duda la adopción de Rust en la industria automotriz.

97. Flutter para sistemas embebidos

Evaluar

Flutter for embedded hace que sea relativamente fácil crear y mantener una UI moderna similar a las aplicaciones móviles, pero para sistemas integrados como "human-machine interface" (HMI) en automóviles, refrigeradores y otros electrodomésticos. Esto es posible con Flutter ahora admite integradores personalizados, lo que permite la portabilidad a diferentes plataformas. Las apps están escritas en el lenguaje de programación Dart utilizando el SDK y ecosistema de Flutter. Hemos estado construyendo prototipos con él — a nuestras personas desarrolladoras les encanta la experiencia de desarrollo y a nuestros clientes les gusta la agilidad, la velocidad y la experiencia de usuario moderna que brinda.

98. Fugue

Evaluar

En ingeniería de datos estamos viendo una selección desconcertante de herramientas y tecnologías. Especialmente para los ingenieros menos experimentados, puede tener sentido trabajar con una capa de abstracción para adentrarse a las herramientas, y así concentrarse en la tarea en cuestión sin tener que aprender varias API específicas de la tecnología y tener la opción de cambiar las tecnologías subyacentes sin demasiado esfuerzo. Fugue es esa capa de abstracción. Proporciona una interfaz unificada para computación distribuida, lo que permite ejecutar código en Python, pandas y SQL en Spark, Dask, Ray y DuckDB con menos sobreescritura. Sin embargo, si tu equipo ya se ha decidido por un conjunto de tecnologías, y si están familiarizados con sus API y están inmersos en ajustar y optimizar sus sistemas de backend, una capa de abstracción de este tipo proporciona menos valor en nuestra experiencia.

99. Galacean Engine

Evaluar

Galacean Engine es un motor interactivo web-first y mobile-first, diseñado para proporcionar una manera fluida de renderizar arquitectura y animación basadas en componentes de manera compatible con dispositivos móviles. Con su enfoque en el rendering ligero y de alto rendimiento, se ha convertido en una opción cada vez más popular para los equipos de desarrollo que crean juegos móviles atractivos. Es un motor basado en TypeScript que, según los desarrolladores, supera a las alternativas.

100. LangChain

Evaluar

LangChain es un framework para construir aplicaciones con grandes modelos lingüísticos (LLMs, por sus siglas en inglés). Estos modelos han desencadenado una carrera para incorporar la IA generativa en varios casos de uso. Sin embargo, utilizar estos LLMs de forma aislada puede no ser suficiente: hay que combinarlos con tus activos diferenciadores para construir un producto impactante. LangChain cubre este nicho con algunas funcionalidades interesantes, incluyendo la gestión de avisos, el encadenamiento, generación de datos aumentada y un rico conjunto de agentes para determinar qué acciones realizar y en qué orden. Esperamos que más herramientas y marcos evolucionen con los LLMs, y recomendamos evaluar LangChain.

101. mljar-supervised

Evaluar

mljar-supervised es un paquete AutoML de Python que ayuda a comprender y explicar datos tabulares. Nuestros equipos de ciencia de datos están entusiasmados con él y lo utilizan para automatizar el análisis exploratorio de datos. Abstrae la forma común de preprocesar los datos, construir los modelos de machine learning (ML) y realizar ajustes de hiperparámetros para encontrar el mejor modelo. La explicabilidad y la transparencia son principios importantes y ahí es donde brilla mljar-supervised. Permite ver exactamente cómo se construye el pipeline de ML con un detallado informe en formato markdown (marcado ligero) para cada modelo de ML. Definitivamente es un paquete AutoML interesante que vale la pena evaluar para sus necesidades de ML.

102. nanoGPT

Evaluar

nanoGPT es un framework para entrenamiento y fine-tuning de transformadores generativos preentrenados de tamaño medio (GPT). El autor, Andrej Karpathy, hace referencia a las publicaciones Attention is All You Need y OpenAI's GPT-3 para construir un GPT desde cero usando PyTorch. Con todo el revuelo en torno a la IA generativa, queremos destacar nanoGPT por su simplicidad y enfoque en articular claramente los componentes básicos de la arquitectura GPT.

103. pandera

Evaluar

En versiones anteriores del radar, presentamos plataformas de prueba y validación de datos como Great Expectations que pueden ser usados para validar supuestos y probar la calidad de los datos entrantes utilizados para entrenamiento o clasificación. A veces, sin embargo, lo que necesitas es una simple librería de código para implementar verificaciones de pruebas y controles de calidad directamente en los pipelines. pandera es una librería en Python para probar y validar datos a través de una amplia gama de tipos de frames como pandas, Dask o PySpark. pandera puede implementar aserciones simples sobre campos o pruebas de hipótesis basadas en modelos estadísticos. La amplia gama de librerías de frames compatibles implica que las pruebas pueden ser escritas una vez y luego aplicada a una variedad de formatos de datos subyacentes. pandera puede usarse también para generar datos sintéticos para probar modelos ML.

104. Qwik

Evaluar

Uno de los retos de crear una experiencia rica e interactiva basada en el navegador es el minimizar el tiempo desde la primera petición hasta una completa interactividad para el usuario. Al iniciar, la aplicación puede requerir descargar grandes cantidades de JavaScript al navegador o ejecutar un proceso largo para restaurar el estado de la aplicación en el servidor. Qwik es un nuevo framework de front-end que serializa el estado de la aplicación para que pueda ser renderizada en el servidor sin necesidad de rehidratar y reproducir la lógica de la aplicación. Esto se logra a través de reanudabilidad (resumability), que involucra el pausar la ejecución en el servidor para reanudarla en el cliente. Al igual que otros frameworks de front-end modernos, tales como Astro o Svelte, Qwik también agiliza los tiempos de carga iniciales de una página al minimizar la cantidad de JavaScript a cargar. En el caso de Qwik, la descarga inicial de la aplicación es principalmente HTML, con la mayoría de JavaScript cargado dinámicamente bajo demanda, si es posible, desde una caché local.

105. SolidJS

Evaluar

SolidJS Es una librería declarativa de JavaScript para crear interfaces de usuario. Este último año hemos visto un incremento en la visibilidad y popularidad de SolidJS entre desarrolladores, particularmente en aquellos interesados en crear interacciones de usuario más ricas. SolidJS compila sus plantillas a nodos reales del DOM (en lugar de usar vDOM) y las actualiza con reacciones muy granulares, lo que reduce las actualizaciones innecesarias en el DOM, dando lugar a un mayor rendimiento y una mejor experiencia de usuario. Cuenta con una API simple y con buen soporte para TypeScript, que puede ayudar a atrapar errores durante el desarrollo. Otro beneficio de SolidJS es el pequeño tamaño del paquete, que es ideal para construir aplicaciones webs rápidas y livianas y beneficia el enfoque mobile-first. SolidJS es un framework relativamente nuevo, por lo que no tiene una comunidad o ecosistema tan grande como otros frameworks. Sin embargo, a juzgar por el creciente número de librerías y herramientas útiles, parece que está creciendo en popularidad. Su sistema de actualización reactivo, su modelado funcional de componentes y su sistema de plantillas hace a SolidJS una atractiva elección a evaluar. Estamos viendo interés por parte de muchos equipos y comunidades.

106. Turborepo

Evaluar

Uno de los temas que constantemente capta nuestra atención en nuestras conversaciones es el de los monorepos. En algunas compañías los han adoptado para toda la organización, mientras que en otras han aplicado el concepto en determinadas aplicaciones concretas como aplicaciones móviles o desarrollos que combinan IU/BFF. Independientemente de si los monorepos son apropiados o cuándo, la industria parece estar reconsiderando herramientas que puedan gestionar de manera efectiva grandes bases de código y construirlas de manera eficiente como unidades desplegables. Turborepo es una herramienta relativamente nueva en esta categoría que ofrece una alternativa a Nx o Lerna para grandes bases de código en JavaScript o TypeScript. Uno de los retos con el manejo de los repositorios grandes es poder ejecutar su construcción de manera suficientemente rápida para no interrumpir el flujo del desarrollador o reducir su eficiencia. Turborepo está escrito en Rust lo que le proporciona un alto rendimiento; también realiza la construcción de manera incremental y cachea los pasos intermedios para lograr aun más velocidad. Sin embargo, obliga a cambios en el

flujo del desarrollador que llevan tiempo aprender y probablemente es más adecuado para grandes bases de código en las que existen múltiples procesos de construcción independientes donde se pueda garantizar un enfoque diferente. Hemos encontrado que la documentación es escasa, lo que ha llevado a algunos equipos a seguir utilizando de momento herramientas más establecidas. Sin embargo, merece la pena evaluar si Turborepo y su más nuevo acompañante,Turbopack (actualmente en beta), continúan evolucionando.

107. API para dispositivos WebXR

Evaluar

Al trabajar en la API experimental WebVR se hizo obvio que tendría más sentido tener una API combinada para VR y AR. En lugar de cambiar significativamente la API para WebVR, se creó una nueva especificación: WebXR. En esencia es la API para dispositivos WebXR la que provee capacidades clave para escribir aplicaciones VR y AR en navegadores web. La API es extensible, y al momento de escribir esto no es totalmente compatible con todos los navegadores. Nuestros equipos han usado WebXR en varias ocasiones y podemos ver los beneficios descritos por Immersive Web Working Group. Cuando trabajamos en prototipos, nos gusta especialmente que la experiencia esté inmediatamente disponible en el navegador web. El equipo de desarrollo no tiene que pasar por el proceso del app-store y los usuarios pueden jugar con la experiencia sin la necesidad de instalar una aplicación. Dado el estado de la API y el hecho de que está escondida detrás de una configuración en algunos navegadores, no la hemos visto más allá de las pruebas de concepto y prototipos.

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

Síguenos en tu red social favorita o suscríbete.

Suscríbete ahora

Thoughtworks es una consultora global de tecnología que integra estrategia, diseño e ingeniería para impulsar la innovación digital. Somos 12,500+ personas en 50 oficinas en 18 países. En los últimos 25+ años, hemos logrado un impacto extraordinario junto con nuestros clientes, ayudándoles a resolver problemas complejos de negocio a través de la tecnología como elemento diferenciador.

/thoughtworks