ThoughtWorks®

TECHNOLOGY RADAR *VOL.16*

Análisis tecnológicos y
las tendencias que están
dando forma al futuro.

thoughtworks.com/es/radar

#TWTechRadar

CONTRIBUYENTES

El Technology Radar está preparado por el Consejo Tecnológico de ThoughtWorks, compuesto por:

Rebecca Parsons (CTO) | Martin Fowler (Chief Scientist) | Badri Janakiraman | Bharani Subramaniam | Camilla Crispim
Erik Doernenburg | Evan Bottcher | Fausto de la Torre | Hao Xu | Ian Cartwright
James Lewis | Jonny LeRoy | Marco Valtas | Mike Mason | Neal Ford
Rachel Laycock | Scott Shaw | Srihari Srinivasan | Zhamak Dehghani

¿Qué hay de nuevo?

Temas destacados en esta edición:

INTERFAZ DE USUARIO CONVERSACIONAL Y PROCESAMIENTO DE LENGUAJE NATURAL

mira el video (thght.works/ConUI)

La conversación—una nueva forma de interactuar con las aplicaciones—tomó al ecosistema por sorpresa con herramientas como Siri, Cortana y Allo, y luego se extendió hacia los hogares con aparatos como Amazon Echo y Google Home.

La construcción de interfaces de usuario conversacionales y de lenguaje natural, a pesar que presentan nuevos retos, tiene beneficios obvios. El equipo detrás de Echo intencionalmente omitió una pantalla, forzándolos a repensar muchas interacciones entre humanos y máquinas.

La tendencia conversacional no está solamente limitada a la voz; a medida que las aplicaciones de mensajería crecen, dominando tanto teléfonos móviles como lugares de trabajo, vemos que las conversaciones con otros seres humanos son reemplazadas por robots inteligentes (*chatbots*). A medida que estas plataformas mejoran, aprenderán a entender el contexto y la intención de las conversaciones, haciendo que las interacciones sean más reales y por lo tanto más convincentes.

La explosión del interés en el mercado y de los principales medios de comunicación nos lleva consecuentemente a un alza del interés en el desarrollo de este nuevo modo personal de interacción basado en dispositivos externos (*exocortex*).

INTELIGENCIA COMO SERVICIO

mira el video (thght.works/IntSer)

Una familia de plataformas a la que llamamos Inteligencia como Servicio entró en escena recientemente de manera desenfrenada. Estas plataformas abarcan una amplia variedad de potentes herramientas, que incluyen desde procesadores de voz hasta utilidades para el entendimiento de lenguaje natural, reconocimiento de imágenes y aprendizaje profundo.

Las capacidades y habilidades que hace algunos años hubieran consumido ingentes recursos ahora se presentan como plataformas de código abierto o SaaS. Parece que la "guerra de las nubes" ha pasado de competir por la cantidad de almacenamiento y capacidades de cálculo a la provisión de capacidades cognitivas, como se demostró por la voluntad de liberar como código abierto a herramientas previamente diferenciadoras como Kubernetes y Mesos.

Todas las grandes empresas del mercado tienen ofertas en este espacio, así como otras empresas "menores" dignas de evaluación. A pesar de que todavía tenemos reservas acerca de las implicaciones éticas y de privacidad de estos servicios, vemos un gran potencial al utilizar estas poderosas herramientas en formas novedosas. Nuestros clientes ya están investigando qué nuevos horizontes podrían alcanzar al combinar estas herramientas con la inteligencia de sus propios negocios.

EXPERIENCIA EN DESARROLLO COMO NUEVO DIFERENCIADOR

mira el video (thght.works/DevExp)

El diseño de experiencia de usuario ha sido por varios años un diferenciador clave para las empresas enfocadas en productos de tecnología. Sin embargo, el rápido crecimiento de herramientas y productos orientados a las personas desarrolladoras, combinado con la escasez de talentos en el área de ingeniería, ha provocado un enfoque similar en la experiencia en desarrollo.

Cada vez más organizaciones toman en cuenta las ofertas de la nube evaluando la cantidad de problemas

de ingeniería que estas pueden reducir, tratan a las APIs como productos, e impulsan la creación de equipos enfocados en mejorar la productividad. En ThoughtWorks, siempre hemos estado obsesionados con la eficiencia de las prácticas de ingeniería y en promover herramientas y plataformas que hagan más fácil la vida de los y las desarrolladoras, por lo que nos emociona ver que la industria está empezando a adoptar este mismo enfoque.

Las técnicas clave incluyen: tratar a la infraestructura interna como un producto lo suficientemente atractivo para competir frente a ofertas externas, generando interés en el auto servicio, entendiendo la ergonomía de las APIs producidas, conteniendo el "código legado en una caja", y comprometiéndose a efectuar investigaciones empáticas de usuario a los desarrolladores y desarrolladoras que utilizan los servicios.

EL ASCENSO DE LAS PLATAFORMAS

mira el video (thght.works/RiseOTP)

Los temas que aparecen en el Radar surgen de las observaciones y conversaciones efectuadas durante el proceso de investigación; recientemente, mientras construíamos el Radar, nos dimos cuenta de la cantidad de nuevas entradas en el cuadrante Plataformas. Creemos que esto indica la existencia de una tendencia más amplia en el ecosistema del desarrollo de software.

Las empresas importantes de Silicon Valley han destacado como el construir una plataforma adecuada puede entregar beneficios significativos. Parte de su éxito viene de encontrar un nivel útil de encapsulamiento, capacidades y habilidades para la plataforma. Cada vez más, "*platform thinking*" aparece en el ecosistema—partiendo de capacidades y habilidades avanzadas destacadas en el Radar, como lenguaje natural, hasta plataformas de infraestructura como Amazon.

Las empresas están empezando a pensar en las plataformas cuando exponen ciertas funcionalidades en APIs inspiradas en productos. Los equipos de desarrollo se enfocan más en construir plataformas para la integración y la mejora de la experiencia para la gente desarrolladora. Parece que la industria finalmente ha logrado una combinación razonable entre empaquetamiento, conveniencia y utilidad.

Una definición que nos gusta es que las plataformas deberían exponer APIs de auto servicio y ser fáciles de configurar y provisionar en el ambiente del equipo—lo que se cruza convenientemente con otro tema emergente: la experiencia en desarrollo como nuevo diferenciador. En el corto plazo, esperamos ver mayor refinamiento tanto en la definición de capacidades y habilidades de las plataformas.

LA UBICUIDAD DE PYTHON

mira el video (thght.works/PerPyt)

Python es un lenguaje que sigue apareciendo en lugares interesantes. Su facilidad de uso como un lenguaje de programación de propósito general en conjunto con sus fuertes fundamentos matemáticos y de computación científica ha permitido su adopción mayoritaria por las comunidades académicas y de investigación. Recientemente, las tendencias de la industria sobre la popularización de IA y sus aplicaciones, combinada con la madurez de Python 3, han ayudado a traer nuevas comunidades al mundo de Python.

Esta edición del Radar presenta algunas librerías de Python que han ayudado a promover el ecosistema, incluyendo a Scikit-learn en el dominio de aprendizaje automatizado; TensorFlow, Keras, y Airflow para gráficos de flujos de datos inteligentes; y spaCy que implementa procesamiento de lenguaje natural para potenciar APIs con capacidades conversacionales. Con más frecuencia vemos a Python como el puente entre científicos e ingenieros dentro de las organizaciones y ayudando a reducir prejuicios sobre sus herramientas favoritas.

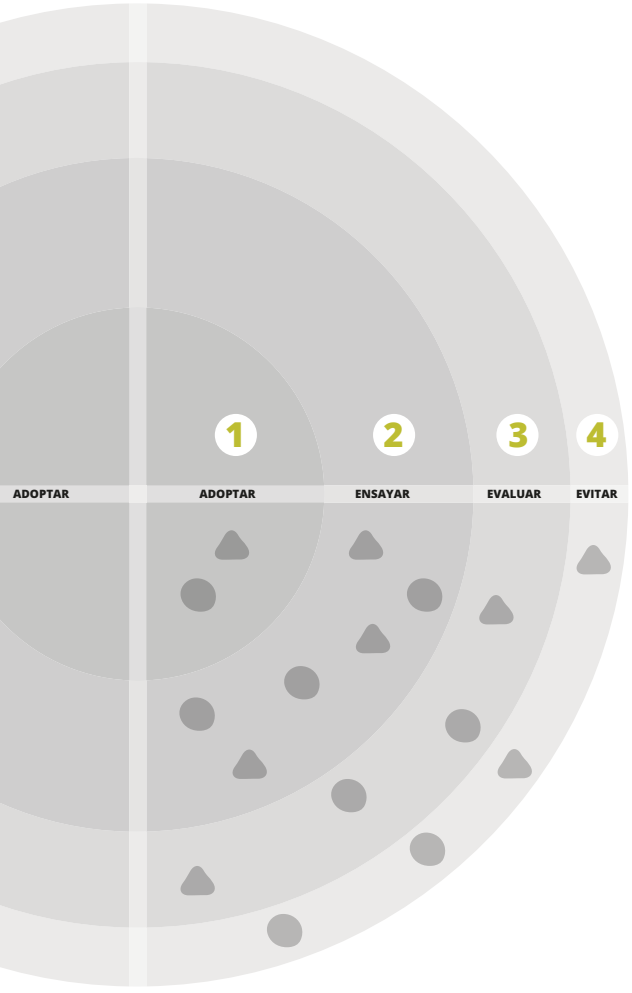
Los enfoques de arquitectura como microservicios y contenedores han facilitado la ejecución de Python en ambientes de producción. Los ingenieros pueden ahora desplegar e integrar código especializado de Python, creado por científicos, mediante APIs agnósticas a lenguajes y tecnologías. Esta fluidez es un gran paso hacia un ecosistema consistente entre investigadores e ingenieros, que contrasta con la práctica de facto de traducir lenguajes especializados como R a ambientes de producción.

ACERCA DEL RADAR

Los ThoughtWorkers son individuos apasionados por la tecnología. La construimos, la investigamos, la probamos, la hacemos con código abierto, escribimos acerca de ella, y constantemente nos esforzamos para mejorarla—para todos. Nuestra misión es defender la excelencia del software y revolucionar la Industria de TI. Creamos y compartimos el Technology Radar de ThoughtWorks en honor a esta misión. El Comité de Tecnología de ThoughtWorks, un grupo de líderes senior en tecnología, crea el Radar. Ellos se reúnen periodicamente para discutir la estrategia global de tecnología y las tendencias tecnológicas que significativamente impactan a nuestra industria. El Radar captura los resultados de las discusiones del Comité Asesor en un formato que provee valor a un amplio rango de personas interesadas, desde CIOs hasta desarrolladores. El contenido está direccionado a ser un resumen conciso.

Te alentamos a explorar estas tecnologías a más detalle. El Radar es gráfico por naturaleza, agrupando elementos en técnicas, herramientas, plataformas, y lenguajes & marcos de trabajo. Cuando los elementos del Radar aparecen en múltiples cuadrantes, escogemos uno que nos parezca más apropiado. Más adelante agrupamos estos elementos en cuatro anillos que reflejan nuestro posición actual acerca de ellos.

Para más información acerca del radar accede a thoughtworks.com/radar/faq



UN VISTAZO AL RADAR

1 ADOPTAR

Creemos fuertemente que la industria debería adoptar estos elementos. Nosotros los usamos apropiadamente en nuestros proyectos.

2 ENSAYAR

Vale la pena seguirles la pista. Es importante entender cómo construir esta capacidad/habilidad (*capability*). Las empresas deberían probar esta tecnología en un proyecto que pueda gestionar el riesgo.

3 EVALUAR

Vale la pena explorar con el objetivo de entender como afectarán a la empresa.

4 EVITAR

Proceder con precaución.

▲ NUEVO O CAMBIADO

● SIN CAMBIO

Los elementos que son nuevos o tienen cambios significativos desde el último Radar son representados con triángulos, mientras que los elementos que no han sido cambiados son representados con círculos.

! Nuestro Radar siempre mira al futuro. Para hacer espacio a nuevos elementos, hemos quitados aquellos que no han tenido un movimiento reciente, lo cual no es un reflejo de su valor sino de nuestro radar con sus limitaciones.

EL RADAR

TÉCNICAS

ADOPTAR
1. Pipelines como código

ENSAYAR
2. APIs como producto
3. Desacoplamiento de gestión secreta de código fuente *NUEVO*
4. Hosting PII data in the EU
5. Legado en una caja *NUEVO*
6. Lightweight Architecture Decision Records
7. Aplicaciones Web Progresivas *NUEVO*
8. Prototipo con InVision y Sketch *NUEVO*
9. Arquitectura sin servidores

EVUALUAR
10. Client-directed query
11. Container security scanning
12. Conversationally aware APIs *NUEVO*
13. Differential privacy
14. Micro frontends
15. Platform engineering product teams *NUEVO*
16. Social code analysis *NUEVO*
17. Realidad Virtual más allá de los juegos

EVITAR
18. A single CI instance for all teams
19. Anemic REST
20. Big Data envy
21. CI theatre *NUEVO*
22. Enterprise-wide integration test environments *NUEVO*
23. Spec-based codegen *NUEVO*

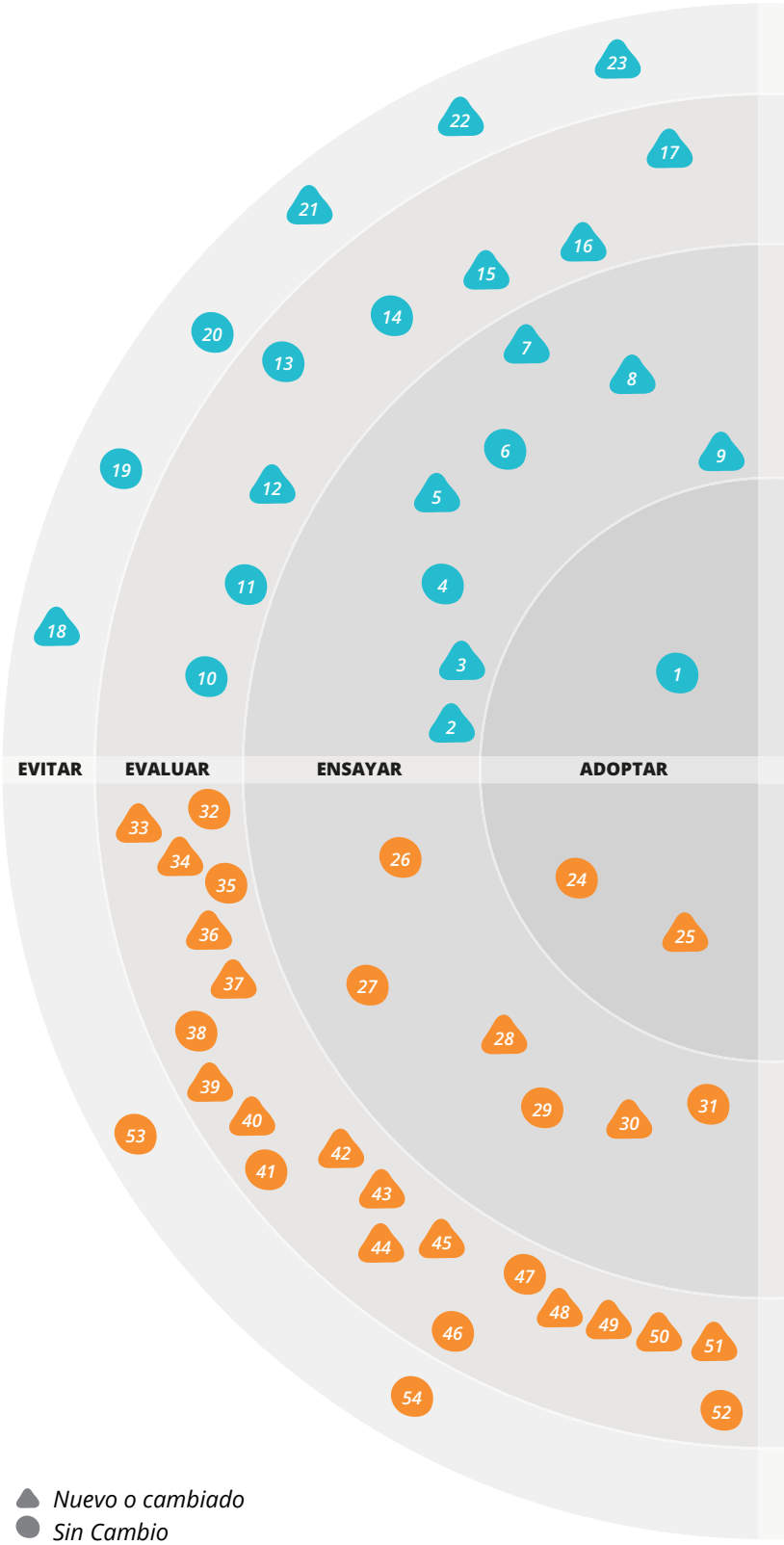
PLATAFORMAS

ADOPTAR
24. HSTS
25. Modelos de seguridad de Linux

ENSAYAR
26. Apache Mesos
27. Auth0
28. AWS Device Farm *NUEVO*
29. AWS Lambda
30. OpenTracing *NUEVO*
31. Unity beyond gaming

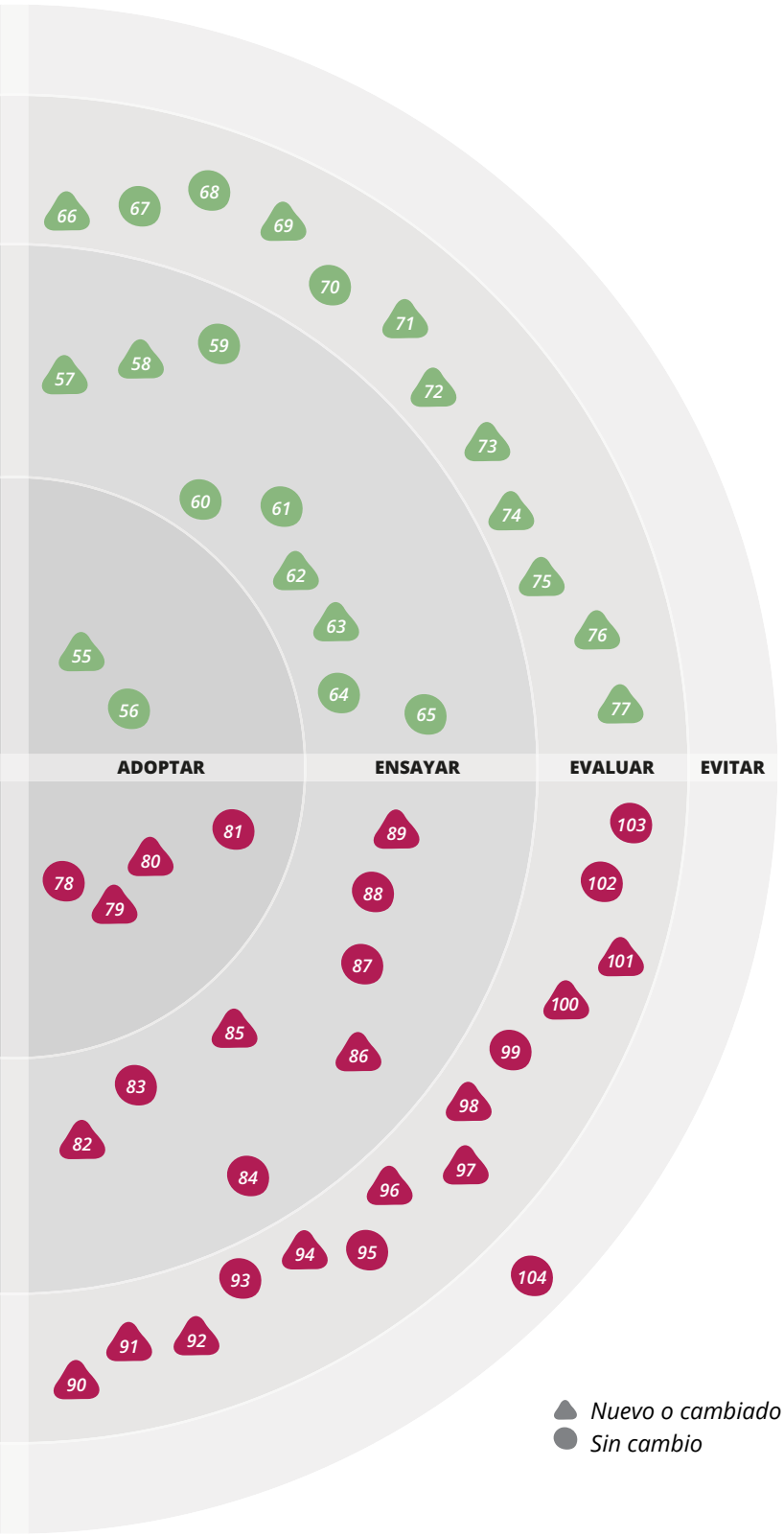
EVUALUAR
32. .NET Core
33. Amazon API Gateway
34. api.ai *NUEVO*
35. Cassandra carefully
36. Cloud-based image comprehension *NUEVO*
37. DataStax Enterprise Graph *NUEVO*
38. Electron
39. Ethereum
40. Hyperledger *NUEVO*
41. IndiaStack
42. Kafka Streams *NUEVO*
43. Keycloak *NUEVO*
44. Mesosphere DCOS
45. Mosquitto *NUEVO*
46. Nuance Mix
47. OpenVR
48. PlatformIO *NUEVO*
49. Tango *NUEVO*
50. Voice platforms *NUEVO*
51. WebVR *NUEVO*
52. wit.ai

EVITAR
53. CMS como plataforma
54. Overambitious API gateways



Nuevo o cambiado
Sin Cambio

EL RADAR



HERRAMIENTAS

ADOPTAR
55. fastlane
56. Grafana

ENSAYAR
57. Airflow *Nuevo*
58. Cake and Fake *Nuevo*
59. Galen
60. HashiCorp Vault
61. Pa11y
62. Scikit-learn
63. Serverless Framework *NUEVO*
64. Talisman
65. Terraform

EVALUAR
66. Amazon Rekognition *NUEVO*
67. Android-x86
68. Bottled Water
69. Claudia *NUEVO*
70. Clojure.spec
71. InSpec *NUEVO*
72. Molecule *NUEVO*
73. Spacemacs *NUEVO*
74. spaCy *NUEVO*
75. Spinnaker *NUEVO*
76. Testinfra *NUEVO*
77. Yarn *NUEVO*

EVITAR

LENGUAJES & MARCOS DE TRABAJO

ADOPTAR
78. Ember.js
79. Python 3
80. ReactiveX
81. Redux

ENSAYAR
82. Avro *NUEVO*
83. Elixir
84. Enzyme
85. Hangfire *NUEVO*
86. Nightwatch *NUEVO*
87. Phoenix
88. Quick and Nimble
89. Vue.js

EVALUAR
90. Angular 2 *NUEVO*
91. Caffe *NUEVO*
92. DeepLearning.scala *NUEVO*
93. ECMAScript 2017
94. Instana *NUEVO*
95. JuMP
96. Keras *NUEVO*
97. Knet.jl *NUEVO*
98. Kotlin *NUEVO*
99. Physical Web
100. PostCSS *NUEVO*
101. Spring Cloud *NUEVO*
102. Three.js
103. WebRTC

EVITAR
104. AngularJS

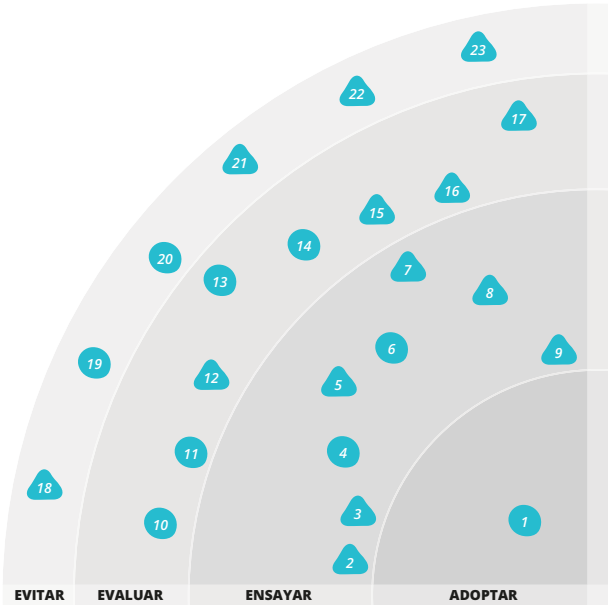
TÉCNICAS

ADOPTAR
1. Pipelines como code

ENSAYAR
2. APIs como producto
3. Desacoplamiento del manejo de secretos del código fuente *NUEVO*
4. Presentando PII data en EU
5. Legado en una caja *NUEVO*
6. Lightweight achitecture decision records
7. Aplicaciones web progresivas *NUEVO*
8. Prototipado con InVision y Sketch *NUEVO*
9. Arquitectura sin servidores

EVALUAR
10. Client-directed query
11. Container security scanning
12. APIs conscientes de conversaciones *NUEVO*
13. Privacidad diferencial
14. Micro frontends
15. Equipos de productos de ingeniería de plataformas *NUEVO*
16. El análisis de código social *NUEVO*
17. VR más allá de los juegos

EVITAR
18. Una única instancia de CI para todos los equipos
19. Anemic REST
20. Big Data envy
21. Práctica de CI *NUEVO*
22. Entornos empresariales para pruebas de integración *NUEVO*
23. Generación de código basado en especificaciones *NUEVO*



Compañías han adoptado con entusiasmo a las APIs como medios de exponer las capacidades de los negocios para desarrolladores externos e internos. Las APIs prometen la habilidad de experimentar rápidamente con nuevas ideas de negocio mediante la recombinación de capabilidades núcleo. ¿Pero qué diferencia una API de un servicio ordinario de integración empresarial? Una de las diferencias radica en el tratamiento de **APIS COMO PRODUCTO**, incluso cuando el consumidor es un sistema interno o una persona compañera de desarrollo. Los equipos que construyen las APIs deben entender las necesidades de sus clientes y desarrollar productos cautivadores para los mismos. Las pruebas de usabilidad y los estudios de UX pueden guiarnos hacia a mejores diseños y entendimiento de los patrones de usabilidad de la API y ayudar a proporcionar una mentalidad de producto a las APIs. APIs, como productos, deberían ser activamente mantenidas y soportadas así como fáciles de usar. Deberían tener a una persona dueña quien aboga por el cliente y se esfuerce por el mejoramiento continuo. En nuestra experiencia, la orientación a productos es el ingrediente faltante que hace la diferencia entre integración empresarial ordinaria y un negocio ágil construido en una plataforma de APIs.

Los equipos que construyen las APIs deben entender las necesidades de sus clientes y desarrollar productos cautivadores para los mismos. Las pruebas de usabilidad y los estudios de UX pueden guiarnos hacia a mejores diseños y entendimiento de los patrones de usabilidad de la API y ayudar a proporcionar una mentalidad de producto a las APIs.

— APIs como producto

En ediciones anteriores del Radar, mencionamos herramientas como git-crypt y Blackbox que nos permiten mantener datos secretos de forma segura dentro del código fuente. **EL DESACOPLAMIENTO DEL MANEJO DE SECRETOS DEL CÓDIGO FUENTE** es nuestra manera de recordar a las personas del mundo tecnológico que existen otras opciones para guardar datos secretos. Por ejemplo, HashiCorp vault, los servidores de CI y las herramientas de manejo de configuración proveen mecanismos para guardar secretos de manera externa al código fuente de la aplicación. Ambos enfoques son viables y recomendamos que se utilice al menos uno de ellos en los proyectos.

Trabajar con código legado, especialmente monolitos grandes, es una de las experiencias más insatisfactorias y de alta fricción para los desarrolladores y desarrolladoras. Pese a que aconsejamos evitar extender y mantener activamente monolitos heredados, continúan habiendo dependencias en nuestros entornos, y las personas desarrolladoras a menudo subestiman el costo y el tiempo necesario para desarrollar haciendo uso de estas dependencias. Para ayudar a reducir la fricción, los desarrolladores y desarrolladoras han utilizado imágenes de máquina virtualizadas o imágenes de contenedores con Docker para crear imágenes inmutables de sistemas legados y sus configuraciones. La intención es contener el código **LEGADO EN UNA CAJA** para que desarrolladores y desarrolladoras lo ejecuten localmente y eliminen la necesidad de reconstruir, reconfigurar o compartir entornos. En un escenario ideal, los equipos que poseen sistemas legados generan las correspondientes cajas legadas a través de sus distintas etapas de contrucción, y las personas desarrolladoras puedan entonces ejecutar y orquestar estas imágenes dentro de un entorno controlado de forma más fiable. Aunque este enfoque ha reducido el tiempo promedio dedicado por cada desarrollador o desarrolladora, ha tenido un éxito limitado cuando los equipos dueños de las dependencias han sido reacios a crear contenedores para que otros equipos los utilicen.

El incremento de **APLICACIONES WEB PROGRESIVAS** (Progressive Web Applications - PWAs) es el intento más reciente de traer de regreso a la web móvil en respuesta a la "fatiga de apps" de usuarios. Propuesto originalmente por Google en 2015, las PWAs son aplicaciones web que toman ventaja de las tecnologías actuales para combinar lo mejor de las aplicaciones web y nativas. Usando un conjunto de tecnologías estándares abiertas como los service workers, el manifiesto de aplicaciones, y APIs con cache and push, podemos crear aplicaciones independientes de plataforma y entregar experiencias de usuario semejantes a las de aplicaciones nativas. Esto pone a la par a las aplicaciones web y nativas, ayudando a los desarrolladores móviles a llegar a usuarios más allá de las murallas de las tiendas de aplicaciones. Piensa en los PWAs como sitios web que actúan y se ven como aplicaciones nativas.

El uso combinado de InVision y Sketch ha cambiado la forma en la que algunas personas se aproximan al desarrollo de una aplicación web. Aunque estas son herramientas, en realidad es la técnica de **PROTOTIPADO CON INVISION Y SKETCH** que hace a este blip significativo. Crear prototipos ricos, clickeables como el punto de inicio para implementar comportamiento de front-end y back-end ayuda a acelerar el desarrollo y elimina el apresuramiento fen la implementación de detalles. El uso combinado de estas tecnologías atiene el balance correcto entre elaboración temprana de detalle visual y la captura temprana de retroalimentación del usuario en la experiencia interactiva.

El enfoque de una **ARQUITECTURA SIN SERVIDORES - SERVERLESS ARCHITECTURE** reemplaza las máquinas virtuales de ejecución prolongada con poder de procesamiento efímero que inicia durante el pedido y desaparece inmediatamente después de su uso. Nuestros equipos gustan del enfoque serverless; nos ha funcionado bien y la consideramos una opción válida de arquitectura. Se toma en cuenta que la arquitectura serverless no debe ser un enfoque de "todo o nada". Algunos de nuestros equipos han desplegado una

parte “nueva” de sus sistemas con la tecnología serverless mientras que han mantenido el enfoque de arquitectura tradicional para otras partes. A pesar que, AWS Lambda es casi un sinónimo de serverless, los otros grandes proveedores de la nube tienen ofertas similares y también deberíamos evaluar servicios niche como webtask.

Tecnologías como Amazon Alexa, Google Voice y Siri han disminuido drásticamente la barrera para las interacciónes basadas en voz con el software. Sin embargo, una entrada más conversacional (sea voz o texto) puede ser difícil de desarrollar sobre muchas API existentes

— APIs Conscientes de Conversaciones

Tecnologías como Amazon Alexa, Google Voice y Siri han disminuido drásticamente la barrera para las interacciónes basadas en voz con el software. Sin embargo, una entrada más conversacional (sea voz o texto) puede ser difícil de desarrollar sobre muchas API existentes, especialmente cuando se trata de una interacción prolongada donde se debe mantener constantemente el contexto de la conversación. Este tipo de conversaciones se daría, por ejemplo, al momento de preguntar sobre el horario de los trenes desde Manchester a Glasgow y luego preguntar “¿A qué hora es la primera salida?” sin tener que volver a dar el contexto de la conversación. Normalmente este contexto estaría presente en la primera respuesta que enviamos de vuelta al navegador, pero en el caso de interfaces de voz necesitamos manejar este contexto en otro lugar. Las **APIS CONSCIENTES DE CONVERSACIONES** pueden ser un ejemplo del patrón backend para front-end, donde el front-end es la plataforma de chat o interacción de voz. Este tipo de API puede manejar las especificidades de este estilo de interacción al gestionar diferentes tipos de conversación mientras llama a los servicios subyacentes en nombre de el front-end de voz.

La adopción de la nube y DevOps, al mismo tiempo que ha incrementado la productividad de los equipos permitiéndoles moverse más rápido al reducir sus dependencias con los equipos de operaciones centralizadas e infraestructura, también ha limitado a los equipos que carecen de habilidades para auto gestionar una aplicación y la operación de la misma en su totalidad. Algunas organizaciones han enfrentado este reto con la creación de **EQUIPOS DE PRODUCTOS DE INGENIERÍA DE PLATAFORMAS**. Estos equipos operan una plataforma interna permitiendo a los equipos de entrega desplegar y operar sistemas en una modalidad de autoservicio con un tiempo de entrega bajo y una complejidad técnica reducida. Aquí, el énfasis está en el autoservicio basado en APIs y herramientas de apoyo, manteniendo a los equipos de entrega como responsables de dar soporte a lo que despliegan en la plataforma. Las organizaciones que consideran establecer dichos equipos de plataforma deben ser cautos en no crear accidentalmente un equipo de DevOps separado, ni renombrar simplemente a su eestructura existente de hosting y operaciones como una plataforma.

EL ANÁLISIS DE CÓDIGO SOCIAL enriquece nuestro conocimiento de la calidad del código superponiendo el comportamiento del desarrollador o desarrolladora con el análisis estructural del código. Utiliza datos del sistema de control de versiones como frecuencia y tiempo de cambio, así como la persona que realiza los cambios. Puedes escoger entre escribir tus propios scripts para analizar los datos o usar herramientas como CodeScene. CodeScene puede ayudarte a ganar un mejor entendimiento de tus sistemas identificando puntos calientes y complejidades, subsistemas difíciles de mantener, acoplamiento entre subsistemas distribuidos a través del acoplamiento temporal, así como la visión de la ley de Conway en tu organización. Creemos que con las tendencias tecnológicas como sistemas distribuidos, microservicios y equipos distribuidos, la dimensión social de nuestro código es vital en nuestra comprensión holística de la salud de nuestros sistemas.

La idea de la realidad virtual ha existido por más de 50 años, y con avances sucesivos en la tecnología de computación muchas ideas han sido promocionadas y exploradas. Creemos que hemos llegado a un punto de inflexión. Cascos de VR orientados al consumidor, a precios razonables, fueron introducidos al mercado el año pasado y las tarjetas gráficas modernas proporcionan suficiente capacidad de procesamiento para crear experiencias de inmersión con ellos. Estos equipos están dirigidos principalmente a los entusiastas de videojuegos, pero estamos convencidos que abrirán puertas a muchas posibilidades de **VR MÁS ALLÁ DE LOS JUEGOS**. Los equipos sin experiencia en el desarrollo de videojuegos no deben subestimar el esfuerzo y las habilidades necesarias para crear buenos modelos tridimensionales y texturas convincentes.

La idea de la realidad virtual ha existido por más de 50 años, y con avances sucesivos en la tecnología de computación muchas ideas han sido promocionadas y exploradas. Creemos que hemos llegado a un punto de inflexión. — VR más allá de los juegos

Nos vemos obligados a advertir, una vez más, contra la creación de **UNA ÚNICA INSTANCIA DE CI PARA TODOS LOS EQUIPOS**. Si bien en teoría es una buena idea consolidar y centralizar la infraestructura de CI, en realidad no vemos suficiente madurez en las herramientas y productos en este espacio para lograr el resultado deseado. Los equipos de entrega de software que deben utilizar la oferta centralizada de CI tienen retrasos largos con regularidad ya que dependen de un equipo central para realizar tareas de configuración de menor importancia o para solucionar problemas en la infraestructura y herramientas compartidas. En esta etapa, seguimos recomendando que las organizaciones limiten su inversión en sistemas centralizados a establecer patrones, directrices y que apoyen a los equipos de entrega para operar su propia infraestructura de CI.

Desde hace tiempo hemos sido defensores de la integración continua (CI), y fuimos pioneros en la construcción de herramientas CI de servidor para compilar proyectos automáticamente por cada check-in de código. Cuando son bien utilizados, estos programas se ejecutan como procesos en segundo plano (como daemon o servicio) sobre una línea principal (mainline) del repositorio del proyecto, donde se realizan commits diariamente. El servidor de CI construye el proyecto y ejecuta pruebas completas para asegurar que todo el sistema de software esté integrado y siempre disponible para ser puesto en producción, satisfaciendo así los principios de entrega continua (*continuous delivery*). Sin embargo, muchos desarrolladores y desarrolladoras simplemente configuran un servidor de CI y asumen equivocadamente que están "haciendo CI" cuando en realidad están perdiendo todos los beneficios. Las formas comunes de falla incluyen: ejecutar CI con una línea principal compartida pero con commits poco frecuentes, por lo que la integración no es realmente continua, ejecutar una compilación con una cobertura de pruebas pobre provocando que la compilación permanezca en rojo durante largos períodos, o ejecutar CI en ramas (branches) distintas, lo que resulta en aislamiento continuo. La falsa "**PRÁCTICA DE CI**" resultante podría hacer sentir bien a la gente pero, fallaría cualquier prueba seria de certificación de CI.

Sin embargo, muchos desarrolladores y desarrolladoras simplemente configuran un servidor de CI y asumen equivocadamente que están "haciendo CI" cuando en realidad están perdiendo todos los beneficios. La falsa "práctica de CI" resultante podría hacer sentir bien a la gente pero, fallaría cualquier prueba seria de certificación de CI.

— Práctica de CI

Cuando las versiones trimestrales o mensuales en la empresa se consideraron las mejores prácticas, fue necesario mantener un entorno completo para realizar ciclos de pruebas antes de su despliegue a producción. Estos **ENTORNOS EMPRESARIALES PARA PRUEBAS DE INTEGRACIÓN** (a menudo denominado SIT o Staging) hoy es un cuello de botella común para la entrega continua. Los entornos en si mísmos son frágiles y costosos de mantener, a menudo con componentes que necesitan una configuración manual por parte de un equipo independiente de gestión del entorno. Las pruebas proporcionan retroalimentación poco fiable y lenta, y el esfuerzo de pruebas se duplica con lo que se puede realizar en los componentes de forma aislada. Recomendamos que las organizaciones creen incrementalmente un camino independiente para puesta en producción de componentes clave. Las técnicas importantes incluyen pruebas de contrato, desacoplamiento de despliegue y lanzamiento, enfoque en tiempo medio de recuperación y pruebas en producción.

En los días en que SOAP dominaba la industria de software empresarial, la práctica de generar código de cliente a partir de las especificaciones de WSDL era una práctica aceptada, incluso alentada. Desafortunadamente, el código resultante era a menudo complejo, no verificable, difícil de modificar y con frecuencia no funcionaba en distintas plataformas de implementación. Con el surgimiento de REST, encontramos que era mejor evolucionar a las API clientes que utilizan el patrón lector tolerante para extraer y procesar sólo los campos necesarios. Recientemente hemos observado un regreso inquietante a los viejos hábitos con personas desarrolladoras que generan el código desde las especificaciones API escritas en Swagger o RAML—una práctica a la cual nos referimos como **GENERACIÓN DE CÓDIGO BASADO EN ESPECIFICACIONES**. Aunque estas herramientas son muy útiles para impulsar el diseño de las API y extraer la documentación, nosotros advertimos contra el atajo tentador de simplemente generar código de cliente directamente desde estas especificaciones. Lo más probable es que este código sea difícil de probar y mantener.

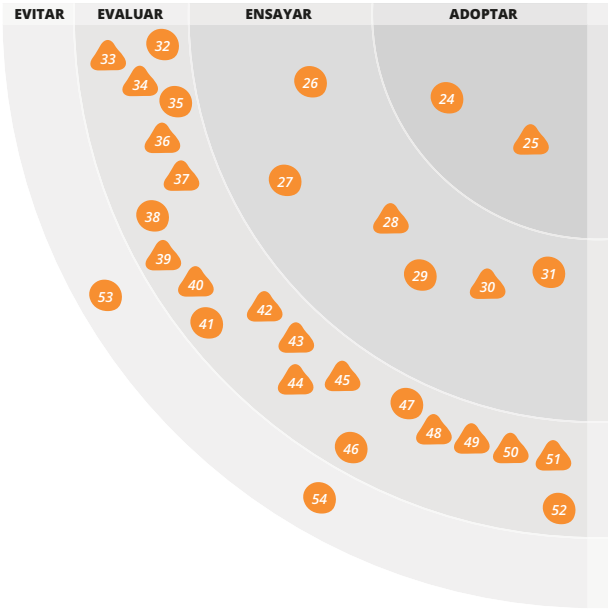
PLATAFORMAS

ADOPTAR
24. HSTS
25. Linux Security Modules

EVUALUAR
26. Apache Mesos
27. Auth0
28. AWS Device Farm *NUEVO*
29. AWS Lambda
30. OpenTracing *NUEVO*
31. Unity beyond gaming

ENSAYAR
32. .NET Core
33. Amazon API Gateway
34. api.ai *NUEVO*
35. Cassandra carefully
36. Cloud-based image comprehension *NUEVO*
37. DataStax Enterprise Graph *NUEVO*
38. Electron
39. Ethereum
40. Hyperledger *NUEVO*
41. IndiaStack
42. Kafka Streams *NUEVO*
43. Keycloak *NUEVO*
44. Mesosphere DCOS
45. Mosquitto *NUEVO*
46. Nuance Mix
47. OpenVR
48. PlatformIO *NUEVO*
49. Tango *NUEVO*
50. Voice platforms *NUEVO*
51. WebVR *NUEVO*
52. wit.ai

EVITAR
53. CMS as a platform
54. Overambitious API gateways



El Principio de Privilegio Mínimo nos alienta a restringir a los componentes de software a que accedan unicamente a los recursos que estos necesitan. Por defecto, un proceso de Linux puede hacer cualquier cosa que su usuario de ejecución puede hacer — desde atarse a puertos arbitrarios hasta crear nuevos shells. El marco de trabajo **MÓDULOS DE SEGURIDAD LINUX** (LSM) que permite a extensiones de seguridad ser conectadas al kernel, ha sido usado para implementar Contro de Acceso Mandatorio (MAC) en Linux. SELinux y AppArmor son las más predominantes y conocidas implementaciones compatibles con LSM que vienen con el kernel. Recomendamos que los equipos aprendan a usar estos uno de estos marcos de trabajo de seguridad (por lo cual lo hemos puesto en el anillo Adoptar). Estos ayudan a los equipos a evaluar preguntas sobre quien tiene acceso a que recursos en hosts compartidos, incluyendo servicios contenidos. Esta aproximación conservadora al manejo de accesos ayudará a los equipos a construir seguridad en sus procesos del ciclo de vida del desarrollo de sistemas..

AMAZON API GATEWAY permite a los desarrolladores exponer servicios API a clientes en Internet. Ofrece las características comunes de un API gateway incluyendo administración del tráfico, monitoreo, autenticación y autorización. Nuestros equipos han tenido experiencias positivas utilizando este servicio junto con AWS Lambda como parte de las arquitecturas sin servidores. Por otro lado, hemos tenido más retos al usarlo como una puerta de enlace de propósito más general al exponer servicios HTTP/HTTPS que se ejecutan en EC2, experimentando obstáculos por la falta de interoperabilidad con VPCs y la dificultad en establecer autenticación certificada de clientes con la puerta de

enlace. Debido a esta experiencia mixta, nos gustaría aconsejar a los equipos que prueben AWS API Gateway con Lambda, pero evalúen su idoneidad al utilizarla en un entorno más general.

El gran número de dispositivos móviles hace casi imposible para las empresas probar sus aplicaciones móviles en todos ellos. Que pase **AWS DEVICE FARM**, un servicio para pruebas de aplicaciones que permite correr e interactuar con sus Android, iOS y aplicaciones web dentro de una amplia variedad de dispositivos físicos que están alojados en la nube, de manera simultánea. Logs detallados, gráficos de rendimiento y capturas de pantalla son generados durante cada corrida proporcionando una retroalimentación general y específica por dispositivo. El servicio ofrece gran flexibilidad para permitir que el estado y la configuración de cada dispositivo sea alterado para reproducir escenarios de prueba muy específicos. Nuestro equipo está usando AWS Device Farm para correr pruebas de extremo a extremo sobre dispositivos con la mayor cantidad de aplicaciones base instaladas.

A medida que las aplicaciones monolíticas están siendo reemplazadas por ecosistemas de (micro)servicios más complejos, el rastreo de solicitudes a través de múltiples servicios se está convirtiendo en la norma.

— OpenTracing

A medida que las aplicaciones monolíticas están siendo reemplazadas por ecosistemas de (micro)servicios más complejos, el rastreo de solicitudes a través de múltiples servicios se está convirtiendo en la norma. Afortunadamente **OPENTRACING** se está convirtiendo rápidamente en el estándar de facto para el rastreo distribuido. Desarrollado por Uber, Apple, Yelp y varios otros grandes, soporta múltiples rastreadores como Zipkin, Instana, y Jaeger. Actualmente, OpenTracing ofrece una implementación vendor-neutral para seis lenguajes: Go, JavaScript, Java, Python, Objective-C y C++.

Paralelamente a la reciente oleada de chatbots y plataformas de voz, hemos visto una proliferación de herramientas y plataformas como **API.AI** que proporcionan un servicio para extraer la intención detrás de un texto y manejar el flujo conversacional que puede cautivar. Recientemente adquirida por Google, esta oferta de "entendimiento de lenguaje natural como un servicio" compite con otros jugadores en esta tendencia como wit.ai y Lex de Amazon.

La comprensión de imágenes solía ser un arte obscuro y requería de un equipo presencial de científicos de datos. Sin embargo en los últimos años, nos hemos acercado a resolver problemas tales como clasificación/categorización, comparación facial, punto de referencia de identificación facial, y reconocimiento facial de imágenes. La **COMPRENSIÓN DE IMÁGENES BASADA EN LA NUBE** provee acceso a capacidades de apredizaje automatizado a través de servicios tales como Amazon Rekognition, Microsoft Computer Vision API y Google Cloud Vision API los cuales pueden suplementar a aplicaciones de realidad aumentada (AR) y cualquier cosa que involucre el etiquetado y clasificación de imágenes.

Hemos tenido algunos éxitos iniciales con **DATASTAX ENTERPRISE GRAPH** (DSE Graph) para gestionar grandes bases de datos gráficas. Construido sobre Cassandra, DSE Graph apunta al tipo de grandes conjuntos de datos donde nuestro favorito Neo4j comienza a mostrar algunas limitaciones. Esta escala tiene sus sacrificios, por ejemplo, se pierden las transacciones ACID y la naturaleza de esquema libre en tiempo de ejecución de Neo4j, pero el acceso a las tablas subyacentes de Cassandra, la integración de Spark para cargas de trabajo analíticas y el poderoso lenguaje de consulta ThinkerPop/Gremlin hacen de esta una opción que vale la pena considerar.

El entusiasmo acerca de los blockchains y las cripto-monedas parece haber alcanzado su punto máximo, como lo demuestra la reducción gradual en la cantidad de anuncios alrededor, y esperamos que algunos de los esfuerzos más especulativos desaparezcan con el

tiempo. Uno de los blockchains, **ETHEREUM**, aunque no es universalmente popular entre los aficionados acérrimos del blockchain, está apareciendo cada vez más en nuevas iniciativas. Ethereum es un blockchain público con un lenguaje de programación incorporado que permite a los desarrolladores y desarrolladoras construir "contratos inteligentes", que son movimientos algorítmicos de ether (la cripto-moneda de Ethereum) en respuesta a la actividad que sucede sobre el blockchain. R3CEV, un consorcio que construye la tecnología blockchain para bancos, construyó sus primeras pruebas de concepto sobre Ethereum. Ethereum ha sido utilizado para construir una organización autónoma distribuida (DAO, distributed autonomous organization) —una de las primeras "corporaciones algorítmicas"— aunque un reciente robo de $150 million en ether demostró que los blockchains y las cripto-monedas siguen siendo parte del Salvaje Oeste del mundo de la tecnología.

HYPERLEDGER es una plataforma construida alrededor de tecnologías blockchain. Consiste en una implementación de blockchain denominada Fabric y otras herramientas asociadas. Haciendo caso omiso de la promoción que rodea blockchain, nuestros equipos han encontrado fácil familiarizarse con estas herramientas. El hecho de que sea una plataforma de código abierto soportada por la "*Linux Foundation*" también contribuye a nuestro interés por Hyperledger.

KAFKA STREAMS es una librería ligera para construir aplicaciones de streaming. Ha sido diseñada con el objetivo de simplificar lo suficiente el procesamiento de la transmisión, para hacerla fácilmente accesible como un modelo general de programación de aplicaciones para servicios asíncronos. Puede ser una buena alternativa en escenarios donde quieras aplicar un modelo de procesamiento de transmisión a tus problemas, sin abarcar la complejidad de desplegar un clúster (usualmente incluido en la utilización de marcos de trabajo integrales de procesamiento de transmisiones).

En una arquitectura de microservicios o en cualquier otro tipo de arquitectura distribuida, una de las necesidades más comunes es asegurar los servicios o las APIs a través de las funcionalidades de autenticación y autorización. En estas situaciones es dónde **KEYCLOAK** interviene. Keycloak es una solución de código abierto para el manejo de acceso e identidad que facilita el aseguramiento de aplicaciones o microservicios utilizando poco o ningún código. De paquete, provee inicio de sesión único, inicio de sesión con redes sociales, y protocolos estándares como OpenID Connect, OAuth2 y SAML.

MESOSPHERE DCOS es una plataforma construida sobre la base de Mesos tla cual abstrae aparte tu infraestructura base por aplicaciones contenerizadas también como por aplicaciones que no se quiere que corran dentro de Docker. Esto puede ser sobresaturado por despliegues mas modestos, pero estamos empezando a ver logros con versiones comerciales y versiones de código abierto. Particularmente, a nosotros nos gusta el que posibilite la portabilidad entre diferentes provedores en la nube, así como en hardware dedicado, y porque no te ata a un framework de contenedores de orquestación en cargas de trabajo contenerizadas. Aunque las actualizaciones pueden ser un poco más complejas de lo que quisieramos, la gran mayoría se encuentra estable y en buena forma.

En nuestra experiencia, para soluciones de Internet de las Cosas (Internet of Things - IoT) donde una gran cantidad de dispositivos se comunican entre sí y/o con un concentrador de datos central - el protocolo de conectividad MQTT ha demostrado su eficacia. También nos ha gustado el agente MQTT **MOSQUITTO**. Es posible que no satisfaga todas las demandas, en particular en lo que respecta a la escalabilidad, pero su naturaleza compacta y fácil configuración lo hace ideal para el desarrollo y pruebas.

PLATFORMIO provee un rico ecosistema para el desarrollo de IoT proporcionando construcciones multiplataforma, manejo de librerías y una buena integración con IDEs existentes. Las funcionalidades inteligentes de autocompletado y análisis de código (Smart Code Linter), junto con una terminal incorporada y un monitor del puerto serial, mejoran notablemente la experiencia del desarrollador. También, organiza y mantiene un gran número de librerías y provee un gestor de dependencias limpio con versionamiento semántico para facilitar el desarrollo de IoT. Hemos empezado a utilizar PlatformIO en unos cuantos proyectos de IoT y, realmente nos gusta por su simplicidad y su amplio soporte de platformas y tableros.

De la mano de la realidad virtual (VR), la cuál ha tenido una barrera relativamente alta para su introducción, debido a los requerimientos de hardware y el esfuerzo para crear mundos virtuales, realidad alternativa (AR) y realidad mixta (MR) también fueron introducidos el último año. Pokémon Go mostró que los teléfonos inteligentes comunes tienen el potencial suficiente para crear experiencias AR/MR convincentes. **TANGO** es una nueva tecnología de hardware de sensores para teléfonos móviles que mejora sus posibilidades para AR/MR. Permite a las aplicaciones adquirir mediciones detalladas en 3-D de los alrededores del usuario para que los objetos virtuales puedan ser localizados y renderizados de manera más convincente para la cámara. Los primeros teléfonos con tecnología Tango se encuentran ya disponibles.

LAS PLATAFORMAS DE VOZ como Amazon Alexa y Google Home están en la cresta de la ola promocional; algunas personas incluso proclaman la omnipresencia de la interfaz de voz conversacional. Ya estamos integrando interfaces conversacionales de usuario en productos y viendo el impacto de esta nueva interacción en cómo diseñamos interfaces. Alexa fue construida específicamente desde cero sin una pantalla y trata la interfaz conversacional de usuario como de primera clase. Pero aún es demasiado pronto para confiarse, y esperamos que otros grandes jugadores entren al juego.

Ya estamos integrando interfaces conversacionales de usuario en productos y viendo el impacto de esta nueva interacción en cómo diseñamos interfaces.

— Las Plataformas de Voz

WEBVR es una API experimental de JavaScript que posibilita el acceso a dispositivos de realidad virtual desde tu navegador. Ha obtenido apoyo de la comunidad y está disponible mediante las construcciones nocturnas así como en algunas versiones de lanzamiento. Si estás buscando cómo construir experiencias de realidad virtual en tu navegador, entonces es un lugar excelente por dónde comenzar. Esta tecnología así como herramientas complementarias como Three.js, A-Frame, ReactVR, Argon.js y Awe.js proporcionan experiencias de realidad aumentada a tu navegador. La proliferación de herramientas en este espacio, junto a los estándares de la comisión de Internet, pudieran ayudar a promover una adopción más fuerte de la realidad aumentada y la realidad virtual.

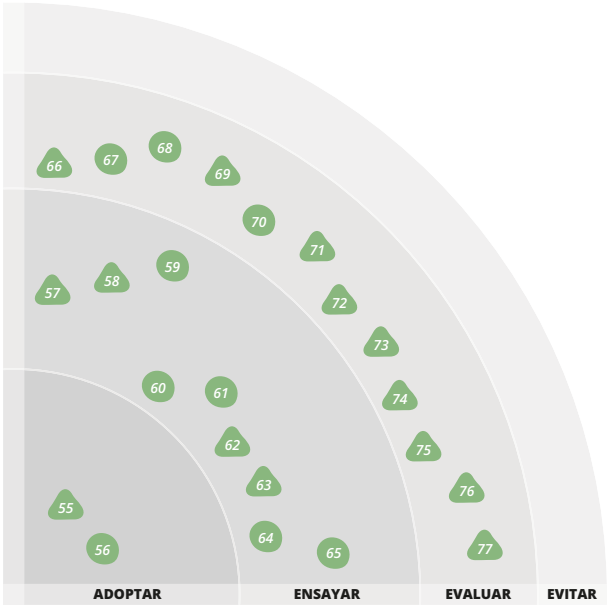
HERRAMIENTAS

ADOPTAR
55. fastlane
56. Grafana

ENSAYAR
57. Airflow *NUEVO*
58. Cake and Fake *NUEVO*
59. Galen
60. HashiCorp Vault
61. Pa11y
62. Scikit-learn
63. Serverless Framework *NUEVO*
64. Talisman
65. Terraform

EVALUAR
66. Amazon Rekognition *NUEVO*
67. Android-x86
68. Bottled Water
69. Claudia *NUEVO*
70. Clojure.spec
71. InSpec *NUEVO*
72. Molecule *NUEVO*
73. Spacemacs *NUEVO*
74. spaCy *NUEVO*
75. Spinnaker *NUEVO*
76. Testinfra *NUEVO*
77. Yarn *NUEVO*

EVITAR



Los desarrolladores y desarrolladoras de aplicaciones Web la tienen fácil cuando se trata de simplificar y automatizar diversos flujos de trabajo de las aplicaciones ya que pueden elegir entre múltiples soluciones que ayudan a automatizar los procesos de entrega. Sin embargo, al desarrollar para dispositivos móviles, se tiene que tratar con dos sistemas operativos que usan mecanismos diferentes para construir, probar, distribuir, generar capturas de pantalla, firmar y distribuir las aplicaciones. Para ayudar a aliviar el dolor, nuestros equipos han adoptado a **FASTLANE** como herramienta para automatizar el proceso de entrega de aplicaciones para iOS y Android. Usando configuraciones sencillas y múltiples pipelines, pueden lograr entrega continua para desarrollo de aplicaciones móviles.

AIRFLOW es una herramienta para crear, programar y supervisar de forma programada pipelines de datos. Al tratar Diagramas Acíclicos Dirigidos (DAGs) como código, facilita pipelines de datos mantenibles, versionables y verificables. Hemos aprovechado esta configuración en nuestros proyectos en donde se ha sido capaz de crear pipelines dinámicas que dieron lugar a flujos de datos ligeros y explícitos. Airflow facilita la definición de operadores y ejecutores y la extención de la librería de forma que se ajuste al nivel de abstracción adecuada para su entorno.

Nuestros equipos han adoptado a fastlane como herramienta para automatizar el proceso de entrega de aplicaciones para iOS y Android.

— fastlane

MSBuild ha sido el principal sistema de construcción en el ecosistema de .NET desde su introducción en 2005; sin embargo, tiene algunas de las mismas debilidades que habíamos mencionado previamente de Maven. La comunidad .NET ha empezado a desarrollar alternativas a MSBuild que sean más fáciles de mantener, más flexibles y que evolucionen más fluidamente con el crecimiento del proyecto. Dos de estas alternativas

son **CAKE Y FAKE**. Cake utiliza una construcción DSL en C#, mientras que Fake utiliza F#. Cada una ha tenido un crecimiento significativo en el último año y ha demostrado ser una alternativa viable a MSBuild para organizar tareas comunes de construcción en proyectos .NET.

La comunidad .NET ha empezado a desarrollar alternativas a MSBuild que sean más fáciles de mantener, más flexibles y que evolucionen más fluidamente con el crecimiento del proyecto.

— Cake y Fake

SCIKIT-LEARN no es una herramienta nueva (se acerca a su décimo cumpleaños); lo que es nuevo es la tasa de adopción de herramientas y técnicas de aprendizaje automatizado fuera de la academia y de las grandes empresas de tecnología. Con un conjunto robusto de modelos y un rico conjunto de funcionalidades, Scikit-learn desempeña un papel importante en hacer que los conceptos y capacidades de aprendizaje automatizado sean más accesibles para una audiencia más amplia (y a menudo no experta).

El popular **SERVERLESS FRAMEWORK** provee herramientas para el andamiaje y despliegue de aplicaciones serverless, principalmente usar AWS Lambda y otras ofertas AWS. Serverless Framework provee suporte para plantillas en JavaScript, Python, Java y C#, y tiene una comunidad activa que contribuye plugins que extienden el marco de trabajo. El marco también apoya el proyecto OpenWhisk de incubación de Apache como una alternativa a AWS Lambda.

AMAZON REKOGNITION es una de las herramientas de reconocimiento de imágenes basadas en la nube que hemos mencionado en algún otro lugar del Radar. Lo que nos gusta de la misma es que Amazon ha tomado el enfoque novedoso de anonimizar los rostros (utilizando GUIDs) desde AWS para acomodar algunas de las preocupaciones de privacidad que vienen con el reconocimiento facial.

La combinación de AWS Lambda con Amazon API Gateway ha tenido un gran impacto en la manera en la que desplegamos servicios y APIs. Sin embargo, incluso en este tipo de configuraciones sin servidor, la cantidad de configuración requerida para atar juntos a los componentes involucrados no es trivial. **CLAUDIA** es una herramienta que automatiza el despliegue de funciones AWS Lambda escritas en JavaScript y las configuraciones asociadas del API Gateway. Provee un conjunto razonable de valores por defecto y nuestros equipos han descubierto que les permite iniciar rápidamente con microservicios basados en Lambda.

¿Cómo un negocio entrega autonomía a equipos de entrega y al mismo tiempo se asegura de que sus soluciones desplegadas sean seguras y regularizadas? ¿Cómo se asegura de que los servidores, una vez desplegados, permanezcan seguros y conformes durante su vida útil? Estos son los problemas que **INSPEC** busca resolver. InSpec es una herramienta para pruebas de infraestructura inspirada en Serverspec, pero con modificaciones que la hacen una herramienta más útil para los profesionales de la seguridad quienes necesitan garantizar la regularización de miles de servidores. Las pruebas individuales pueden ser combinadas en perfiles de seguridad completos y ser ejecutadas remotamente desde la línea de comandos. InSpec es útil para personas desarrolladoras y además puede abarcar la realización de pruebas sobre la infraestructura de producción de forma continua, aproximándose al Análisis de la Calidad en producción.

MOLECULE está diseñado para asistir en el desarrollo y pruebas de roles Ansible. El construir el andamiaje para correr pruebas de rol Ansible en una maquina virtual o contenedor de elección, no se necesita instalar manualmente nuestro ambiente de pruebas. Molecule aprovecha de Vagrant, Docker, y OpenStack para manejar máquinas/contenedores virtuales, y soporta a Serverspec, Testinfra, o Goss tpara correr las pruebas. Los pasos por defecto en el modelo de orden de instalación incluyen: manejo de máquinas virtuales, Ansible linting, pruebas de idempotencia y pruebas de convergencia. Aunque es un proyecto bastante joven, vemos un gran potencial para su uso.

Como diría cualquier fan de Emacs, Emacs es más que un editor de texto; es una plataforma para aplicaciones de mapeo de caracteres. Los últimos años han visto una explosión de nuevos desarrollos en esta plataforma, pero nosotros pensamos que **SPACEMACS** merece atención de forma particular. Spacemacs brinda una introducción a la plataforma Emacs, con una nueva interfaz de usuario de teclado, capas más simples de personalización, y una distribución precisa de paquetes de Emacs. Uno de los objetivos del proyecto es lograr tomar lo mejor de los dos mundos mediante la combinación de la interfaz de usuario de Vim con la programación interna de Emacs. Nosotros consideramos a las herramientas para la productividad de desarrollo como parte vital del proceso efectivo de desarrollo de software, y si no has considerado utilizar Emacs desde hace un tiempo, te sugerimos que le des una mirada a cómo el Spacemacs repiensa esta plataforma clásica de desarrollo.

SPACY es una biblioteca para el Procesamiento de Lenguaje Natural (NLP, Natural Language Processing) escrita en Python. Esta es una biblioteca de alto rendimiento, diseñada para el uso en producción por desarrolladores, que aplica modelos NLP apropiados para el procesamiento de texto que contiene mezclas de emoticonos y signos de puntuación inconsistentes. A diferencia de otros marcos de trabajo NLP, spaCy es una biblioteca extensible y no una plataforma, enfocada en las aplicaciones en producción en lugar de la generación de modelos de entrenamiento para investigación. Se integra bien con TensorFlow y el resto de los ecosistemas de inteligencia artificial de Phyton. Nosotros la hemos utilizado en el contexto empresarial para construir un motor de búsqueda que recibe encuestas en lenguaje humano y ayuda a los usuarios a tomar decisiones de negocios.

Netflix ha abierto el código de **SPINNAKER**, su plataforma de microservicios para entrega continua (CD). Comparado con otras plataformas CI/CD, Spinnaker implementa manejo en cluster y despliegue de imágenes preparadas para la nube como funcionalidades de primer orden. Soporta en su paquete de soluciones el despliegue y manejo de cluster para múltiples provedores de servicios en la nube tales como Google Cloud Platform, AWS y Pivotal Cloud Foundry. Se puede integrar Spinnaker con Jenkins para correr una tarea de construcción Jenkins. Nos agrada el enfoque pragmático de Spinnaker para desplegar microservicios hacia la nube—con la excepción que los pipelines de Spinnaker son creados via interfaz de usuario y no pueden ser configurados como código.

Dado el amplio uso de las herramientas de infraestructura hoy en día, no debería sorprendernos que la infraestructura como código haya aumentado en los proyectos actuales. Con esta tendencia viene la necesidad de probar este código. Con **TESTINFRA** puedes probar el estado real de los servidores configurados manualmente o mediante herramientas como Ansible, Puppet y Docker. Testinfra busca ser un equivalente a Serverspec en Python y está escrito como un plugin para el motor de pruebas Pytest.

Dado el amplio uso de las herramientas de infraestructura hoy en día, no debería sorprendernos que la infraestructura como código haya aumentado en los proyectos actuales.

— Testinfra

YARN es un nuevo manejador de paquetes que reemplaza el flujo de trabajo existente para el cliente npm a la vez que permanece compatible con el registry de npm. Con el cliente npm, podemos terminar con una estructura de árbol diferente bajo node_modules basado en el orden en el que se instalan las dependencias. Esta naturaleza no determinista puede causar problemas de "funciona en mi maquina". Mediante la separación de los pasos de instalación resolución, búsqueda y enlace, Yarn evita estas dificultades utilizando algoritmos deterministas y lockfiles y de esta forma garantiza instalaciones repetibles. También hemos visto construcciones significativamente más rápidas en nuestro ambiente de CI debido al uso de la caché que realiza Yarn para todos los paquetes que descarga.

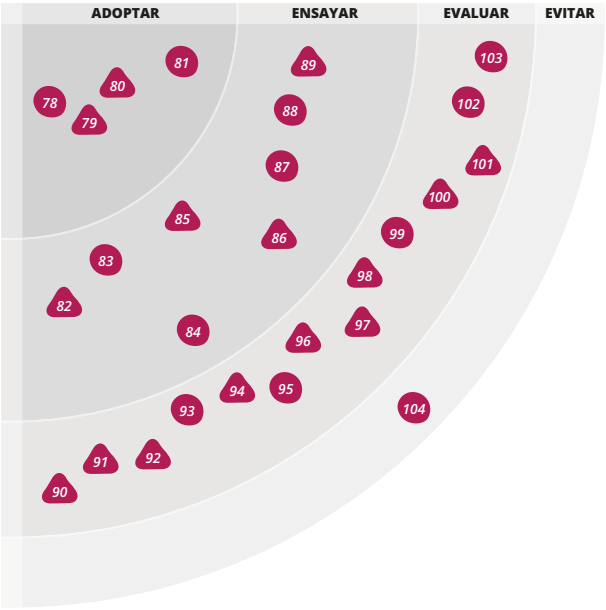
LENGUAJES & MARCOS DE TRABAJO

ADOPTAR
78. Ember.js
79. Python 3
80. ReactiveX
81. Redux

ENSAYAR
82. Avro *NUEVO*
83. Elixir
84. Enzyme
85. Hangfire *NUEVO*
86. Nightwatch *NUEVO*
87. Phoenix
88. Quick and Nimble
89. Vue.js

EVALUAR
90. Angular 2 *NUEVO*
91. Caffe *NUEVO*
92. DeepLearning.scala *NUEVO*
93. ECMAScript 2017
94. Instana *NUEVO*
95. JuMP
96. Keras *NUEVO*
97. Knet.jl *NUEVO*
98. Kotlin *NUEVO*
99. Physical Web
100. PostCSS *NUEVO*
101. Spring Cloud *NUEVO*
102. Three.js
103. WebRTC

EVITAR
104. AngularJS



PYTHON 3 introdujo muchas características útiles que no son compatibles con Python 2.x. También eliminó numerosas características de Python 2.x que se mantuvieron para compatibilidad con versiones anteriores, lo que hace que Python 3 sea más fácil de aprender y usar y más consistente con el resto del lenguaje. Nuestra experiencia en el uso de Python 3 en los dominios como el aprendizaje automatizado y el desarrollo de aplicaciones web muestra que tanto el idioma como la mayoría de sus librerías de soporte han madurado para su adopción. Pudimos bifurcar y corregir problemas menores de librerías existentes o evitar utilizar librerías Python 2.x incompatibles que habían sido abandonadas. Si estás desarrollando en Python, recomendamos fuertemente el uso de Python 3.

Los sistemas distribuidos a menudo utilizan comunicación multihilo, comunicación basada en eventos y E/S sin bloqueo para mejorar la eficiencia general del sistema. Estas técnicas de programación imponen desafíos tales como procesamiento de bajo nivel, sincronización, seguridad de subprocesos, estructuras de datos concurrentes y E/S no bloqueantes. La librería de código abierto **REACTIVEX** abstrae estas preocupaciones de una bella forma, provee la infraestructura requerida y extiende el patrón observable en corrientes de eventos asíncronos. ReactiveX también tiene una comunidad de desarrolladores activa y soporta una creciente lista de idiomas, la más reciente siendo RxSwift. También implementa enlaces a plataformas móviles y de escritorio.

La librería de código abierto ReactiveX abstrae estas preocupaciones de una bella forma, provee la infraestructura requerida y extiende el patrón observable en corrientes de eventos asíncronos.

— ReactiveX

AVRO es un marco de trabajo para la serialización de datos. Al almacenar el esquema junto con el contenido del mensaje, estimula la evolución del mismo. Los productores pueden editar nombres de campos, agregar nuevos campos o eliminar campos existentes y Avro garantiza que los clientes pueden continuar consumiendo los mensajes. Tener un esquema permite que datos se escriban sin sobrecarga, lo que resulta en una codificación de datos compacta y un procesamiento de datos más rápido. Aunque el intercambio de mensajes sin estructura entre el productor y el consumidor es flexible, hemos visto equipos enfrentando problemas con mensajes incompatibles no procesados en la cola durante los despliegues. Hemos utilizado Avro en una serie de proyectos y recomendamos su uso en lugar de enviar mensajes no estructurados.

Hangfire, como nuestros equipos lo descubrieron, puede hacer esto y mucho más en el entorno .NET. Hangfire es fácil de usar y flexible, y abarca un estilo funcional. De particularmente interés es su capacidad de guardar el estado de una tarea para que pueda reanudarse cuando una aplicación se reinicie después de un fallo o un cierre inesperado.

— Hangfire

Un problema común en el desarrollo de aplicaciones es cómo programar tareas que se ejecuten fuera del proceso principal periódicamente o cuando se cumplen ciertas condiciones. El problema se vuelve más complicado cuando se producen eventos inesperados, como un cierre de la aplicación. El framework **HANGFIRE**, como nuestros equipos lo descubrieron, puede hacer esto y mucho más en el entorno .NET. Hangfire es fácil de usar y flexible, y abarca un estilo funcional. De particularmente interés es su capacidad de guardar el estado de una tarea para que pueda reanudarse cuando una aplicación se reinicie después de un fallo o un cierre inesperado.

NIGHTWATCH es un marco de trabajo que permite que pruebas de aceptación automatizadas para aplicaciones basadas en el navegador sean creadas en JavaScript y ejecutadas en Node.js. Nightwatch permite que las pruebas se definan utilizando una API fluida; estas pueden entonces ser ejecutadas sobre un servidor Selenium/WebDriver. En el caso de aplicaciones de página única u otras páginas con alto contenido de JavaScript, esto permite que las pruebas automatizadas se creen y ejecuten dentro del mismo idioma y entorno que la mayor parte del código.

En el siempre cambiante mundo de los marcos de trabajo front-end de JavaScript, uno de los favoritos emergentes parece ser **VUE.JS**. Vue.js es una alternativa ligera a AngularJS. Está diseñado para ser una biblioteca muy flexible y menos restrictiva que ofrece un conjunto de herramientas para construir interfaces web interactivas en torno a conceptos como la modularidad, componentes y el flujo reactivo de datos. Tiene una curva de aprendizaje baja, lo que lo hace interesante para los desarrolladores junior y principiantes. Tenga en cuenta, sin embargo, que Vue.js no es un marco completo; se centra sólo en la capa de vista y por lo tanto es fácil de integrar con otras bibliotecas o proyectos existentes.

En la edición anterior del Radar, movimos a AngularJS al anillo Evitar (donde permanece en esta edición). Cuando hablamos de **ANGULAR 2**, encontramos mensajes contradictorios. Durante el año pasado algunos equipos de ThoughtWorks han utilizado Angular 2 con éxito y lo consideran una opción sólida. Sin embargo, Angular 2 es una reimplementación, no una evolución, de AngularJS, y cambiar de AngularJS a Angular 2 no es muy diferente de cambiar de AngularJS a otro marco de trabajo. Dado que existen, en nuestra experiencia, alternativas superiores como React.js, Ember.js y Vue.js, estamos aún reacios a recomendar Angular 2 con fuerza. Sin embargo, queremos destacar que no es una mala elección, especialmente si gustan de utilizar TypeScript.

CAFFE es una biblioteca de código abierto para aprendizaje profundo creada por Berkeley Vision and Learning Center. Su foco principal son las redes convulacionales y aplicaciones de visión computarizada. Caffe es una elección popular para tareas relacionadas con visión computarizada y puedes descargar muchos modelos exitosos creados por los usuarios de Caffe desde el Caffe Model Zoo para su uso inmediato. Así como Keras, Caffe es una API basada en Python. En Keras, sin embargo, modelos y componentes son objetos creados directamente en código Python, mientras que en Caffe los modelos son descritos por archivos de configuración Protobuf. Cualquiera de estas alternativas tiene sus pros y sus contras, y la conversión entre estos modelos también es posible.

DEEPLEARNING.SCALA es un conjunto de herramientas para aprendizaje profunco de código abierto en Scala, creado por nuestros colegas de ThoughtWorks. Este proyecto nos estusiasma porque usa programación funcional diferenciable para crear y construir redes neuronales; una persona desarrolladora simplemente escribe código en Scala con tipeado estático. DeepLearning.scala actualmente proporciona tipos básicos como float, double, arreglos N-dimensionales acelerados via GPU , así como tipos de datos algebraicos. Esperamos los futuros lanzamientos del conjunto de herramientas, en los cuales se dice que brindarán funciones de mayor orden y entrenamiento distribuido en Spark.

INSTANA es otro entrante más al espacio abarrotado de gestión de rendimiento de aplicaciones. El hecho que está construido desde cero para arquitecturas nativas de la nube diferencia a Instana de muchos de sus competidores. Sus carácteristicas incluyen descubrimiento dinámico, trazamiento distribuido y salud de servicio más la habilidad de "mover el tiempo" al mirar tu infraestructura hacia el momento cuando ocurrió un incidente. Todavía no se conoce si este producto ganará tracción sobre la combinación de proyectos de codigo abierto—tales como Consul, Prometheus y las implementaciones de OpenTracing – que hacen las mismas cosas; sin embardo, merece hecharle un vistazo si necesitas una solución lista para usar.

KERAS es una interface de alto nivel en Python para construir redes neuronales. Creada por un ingeniero de Google, Keras es de código abierto y corre sobre TensorFlow o Theano. Provee una interface sorprendentemente simple para crear poderosos algoritmos de aprendizaje profundo para entrenar en CPUs o GPUs. Keras está bien diseñado con modularidad, simplicidad y extensibiidad de mente. Diferente a una librería como Caffe, Keras soporta más arquitecturas generales de red como las mallas recurrentes, haciéndola en general más útil para análisis de texto, proceso de lenguaje natural y aprendizaje automatizado en general. Si la visión de cómputo, o cualquier otra rama especializada de aprendizaje automatizado, es su principal preocupación, Caffe puede ser tal vez la elección más apropiada. Sin embargo, si buscas aprender un marco de trabajo más simple y poderoso, Keras debería ser tu primera opción.

KNET.JL es el marco de trabajo para aprendizaje automatizado de Koç University implementado en Julia por Deniz Yuret y otras personas colaboradoras. A diferencia de compiladores que generan gradientes como Theano y TensorFlow cuales obligan a los usuarios a un mini lenguaje restringido, Knet permite la definición y entrenamiento de modelos de aprendizaje automatizado usando todo el poder y expresividad de Julia. Kent usa grafos computacionales dinámicos generados al tiempo de ejecución para la diferenciación automática de casi cualquier código de Julia. Nos gusta mucho el respaldo de operaciones GPU mediante el tipo KnetArray, y por si acaso no tengas acceso a una maquina GPU, el equipo tras de Knet también mantiene una imagen Amazon Machine preconfigurada (AMI) para que puedas evaluarla en la nube.

El lenguaje de programación **KOTLIN** está en muchas de las listas de cosas por evaluar de nuestros desarrolladores y desarrolladoras para este año, y algunos ya lo han usado exitosamente en producción. Es un lenguaje JVM de código abierto creado por JetBrains. A nuestros desarrolladores de Swift les gusta ya que es sintácticamente cercano a Swift e igualmente conciso. Nuestros desarrolladores de Java han disfrutado de su interoperabilidad con el lenguaje y herramientas Java y lo ven más facil de aprender que Scala. Kotlin soporta conceptos de programación

funcional pero con menos características que Scala. Desarrolladores y desarrolladoras en nuestros equipos a quienes les gusta el tipado estático, delegando al compilador la tarea de detectar defectos de punteros a nulos, se han encontrado escribiendo menos pruebas de código repetitivo.

POSTCSS es un framework de Node.js-basado en JavaScript, para operar documentos CSS basándose en una representación de árbol de sintaxis abstracta, con un abundante ecosistema de plugins. A menudo pensado incorrectamente como un preprocesador (como un SaaS o Less), encontramos que el poder real de PostCSS viene de un número de cosas que pueden ser hechas con el abundante set de plugins que incluye linting (el plugin stylelint), recopilación cruzada (el plugin sugarss), retorcer nombres para evitar colisión de selectores (el plugin modules), generación de código CSS (el plugin autoprefixer), minification y muchos otros. A pesar de diferentes niveles de madurez de los plugins, PostCSS sigue siendo un framework de trabajo simple y poderoso para trabajar con CSS como un completo lenguaje de desarrollo para front-end.

Los equipos construyendo sistemas compuestos de microservicios necesitan pensar sobre técnicas de coordinación como descubrimiento de servicios, balanceo de carga, circuitos interruptores y revisión de salud. Muchas de estas técnicas requieres que los equipos configuren herramientas, lo cual no siempre es trivial. El proyecto **SPRING CLOUD** provee herramientas para que la gente desarrolladora pueda usar estas técnicas de coordinación en el ambiente familiar de Spring. Estas herramientas soportan a Consul, ZooKeeper y a Netflix OSS full stack, todas herramientas que nos gustn. Puesto simple, hace fácil hacer los correcto con este conjunto de herramientas. Aunque nuestras preocupaciones usuales con Sprin aún siguen en pie, principalmente que esconde mucha complejidad, deberías considerar usar Spring Cloud si estas en el escosistema y necesitas resolver estos problemas.

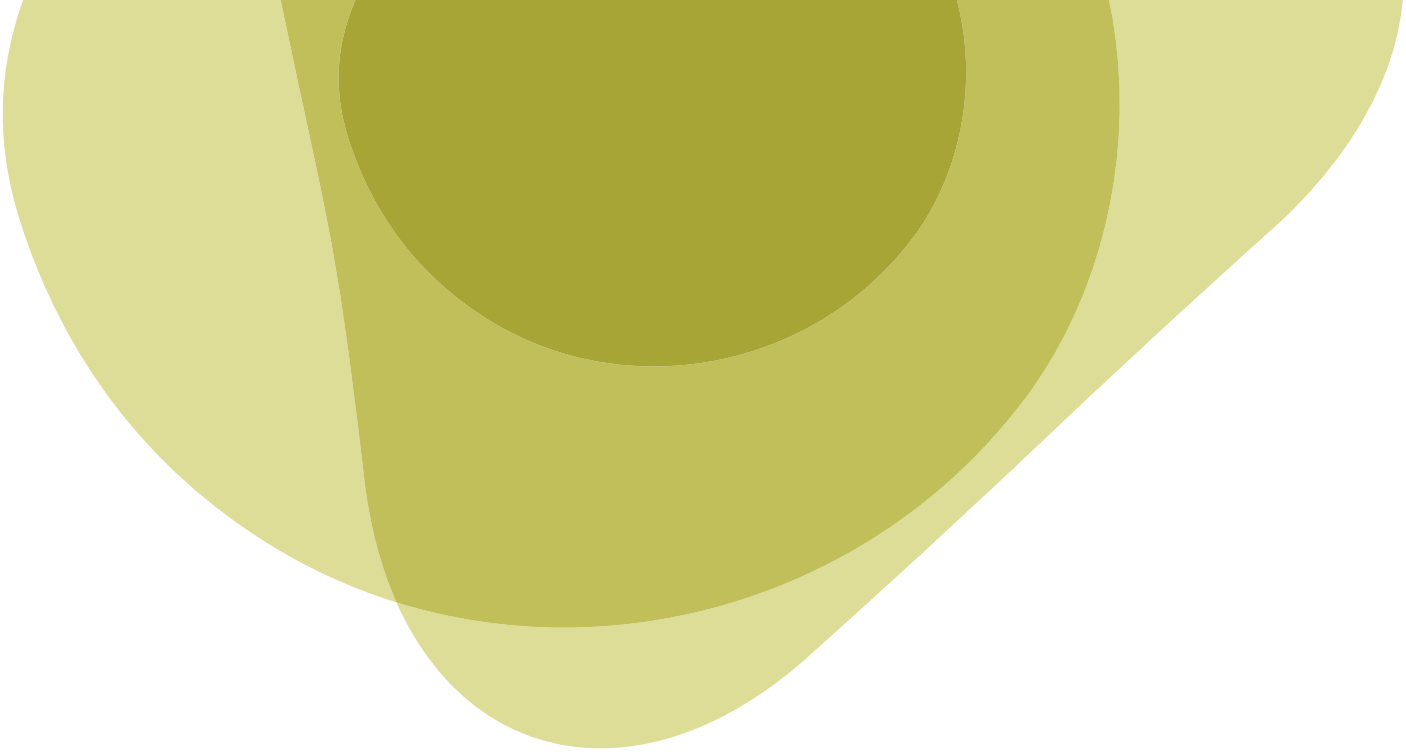
Los equipos construyendo sistemas compuestos de microservicios necesitan pensar sobre técnicas de coordinación como descubrimiento de servicios, balanceo de carga, circuitos interruptores y revisión de salud.

— Spring Cloud

Sé el primero en saber los lanzamientos del Radar Tecnológico, y mantente al tanto con webinars y contenido exclusivo.

SUSCRÍBETE AHORA

thght.works/Sub-ES



ThoughtWorks®

ThoughtWorks es una consultora de tecnología y una comunidad de individuos apasionados guiados por propósitos. Ayudamos a nuestros clientes a incorporar la tecnología en el corazón de su negocio, y juntos creamos software relevante para ellos.

Fundada hace 20 años, ThoughtWorks se ha convertido en una compañía de más de 4000 personas, incluyendo una división de productos que hace productos pioneros de software para equipos. ThoughtWorks tiene 40 oficinas en 14 países: Australia, Brasil, Chile, China, Ecuador, Alemania, India, Italia, Singapore, Sudáfrica, España, Turquía, Reino Unido y los Estados Unidos.

thoughtworks.com