ENERO 2014

TECHNOLOGY RADAR

Elaborado por el Consejo Consultivo de Tecnología de ThoughtWorks

thoughtworks.com/radar

ThoughtWorks®

NOVEDADES

A continuación, se presentan las tendencias destacadas en esta edición:

- **Alerta temprana y recuperación en la producción** - observamos un auge de herramientas y técnicas nuevas para el registro, monitoreo, almacenamiento y la búsqueda de datos operativos. Cuando se las combina con los tiempos cortos de recuperación que se logran gracias a la virtualización y automatización de infraestructuras, las empresas pueden disminuir la cantidad de pruebas que se necesitan antes de la implementación. Es más, quizás hasta pueden incorporar esas pruebas al entorno productivo mismo.

- **Privacidad vs gran cantidad de datos** - si bien nos entusiasman las nuevas perspectivas comerciales que fueron posibles, por un lado, gracias a la exhaustiva recolección de datos y, por el otro, debido a las nuevas herramientas y plataformas para almacenar y analizar esos datos, también nos preocupa el hecho de que muchas empresas almacenen vastas cantidades de datos personales innecesariamente. Recomendamos que las empresas adopten una actitud "datensparsamkeit" y almacenen exclusivamente la información personal mínima y necesaria de sus clientes.

- **JavaScript continua siendo una fuerza imparable** - el ecosistema que rodea a JavaScript como una plataforma sigue evolucionando. Hace poco tiempo surgieron varias herramientas nuevas e interesantes para probar, crear y gestionar dependencias, tanto del lado del servidor como del cliente, de las aplicaciones JavaScript.

- **Fusión de lo físico con lo digital** - los dispositivos de bajo costo, las plataformas de hardware abiertas y los nuevos protocolos de comunicación están llevando la experiencia informática cada vez más lejos de la pantalla y más cerca del mundo que nos rodea. Un buen ejemplo de este fenómeno es la proliferación de dispositivos portátiles para rastrear la biometría personal, y también la compatibilidad de hardware en equipos móviles para interactuar con estos dispositivos.

Las personas que trabajamos en ThoughtWorks sentimos pasión por la tecnología. La creamos, la investigamos, la probamos, la liberamos, escribimos acerca de ella y siempre buscamos mejorarla, para todos. Nuestra misión es defender la excelencia en software y revolucionar el mundo de las tecnologías de la información (TI). Por eso creamos y compartimos el Radar Tecnológico de ThoughtWorks como respaldo a esa misión. La creación del radar es obra de la Comisión Asesora de Tecnología de ThoughtWorks, un grupo de líderes sénior en tecnología de ThoughtWorks. Los miembros de este equipo se reúnen en forma periódica para debatir acerca de la estrategia global en materia de tecnología de ThoughtWorks y abordar las tendencias tecnológicas que repercuten en gran medida sobre nuestra industria.

El radar captura el resultado de los debates de la Comisión Asesora de Tecnología en un formato que aporta valor a una amplia variedad de partes interesadas, desde los directores de información hasta los desarrolladores. El contenido tiene el propósito de brindar un resumen conciso. Lo alentamos a explorar estas tecnologías para obtener mayores detalles. El radar cuenta con una naturaleza gráfica gracias a que agrupa los elementos en técnicas, herramientas, plataformas e idiomas y frameworks. Muchos elementos del radar podían aparecer en múltiples cuadrantes, nosotros elegimos aquellos que nos parecen los más adecuados. Además, agrupamos estos elementos en cuatro círculos para reflejar cuál es nuestra posición actual respecto de ellos. Los círculos son los siguientes:

- **Adopción:** Consideramos firmemente que la industria debe adoptar estos elementos. Los utilizamos en nuestros proyectos cuando resultan apropiados.
- **Ensayo:** Merece un examen atento. Es importante entender cómo se puede desarrollar esta capacidad. Las empresas deberían probar esta tecnología en un proyecto que pueda afrontar el riesgo.
- **Evaluación:** Vale la pena explorar con el fin de entender en qué manera afectará a su empresa.
- **Espera:** Actuar con precaución.

Los elementos que resultan nuevos o que han sufrido cambios importantes desde el último radar se representan con triángulos, mientras que aquellos que no se han modificado se representan con círculos. Los gráficos detallados para cada cuadrante muestran las modificaciones que han sufrido los elementos. Nos interesan muchos más elementos de los que pueden llegar a incluirse en un documento de este tamaño, por lo que quitamos algunos elementos del último radar con el fin de generar el espacio necesario para los nuevos. El hecho de que quitemos un elemento no implica que el mismo ya no nos importe.

Para obtener más información acerca del radar, consulte http://martinfowler.com/articles/radar-faq.html

EL RADAR

TÉCNICAS

***ADOPT* - ADOPCIÓN**
1 Capturing client-side JavaScript errors
2 Continuous delivery for mobile devices
3 Mobile testing on mobile networks
4 Segregated DOM plus node for JS Testing
5 Windows infrastructure automation

***TRIAL* - ENSAYO**
6 Capture domain events explicitily
7 Client and server rendering with same code
8 HTML5 storage instead of cookies
9 Instrument all the things
10 Masterless Chef/Puppet
11 Micro-services
12 Perimeterless enterprise
13 Provisioning testing
14 Structured Logging

***ASSESS* - EVALUACIÓN**
15 Bridging physical and digital worlds with simple hardware
16 Collaborative analytics and data science
17 Datensparsamkeit
18 Development environments in the cloud
19 Focus on mean time to recovery
20 Machine image as a build artifact
21 Tangible interaction

***HOLD* - ESPERA**
22 Cloud lift and shift
23 Ignoring OWASP Top 10
24 Siloed metrics
25 Velocity as productivity

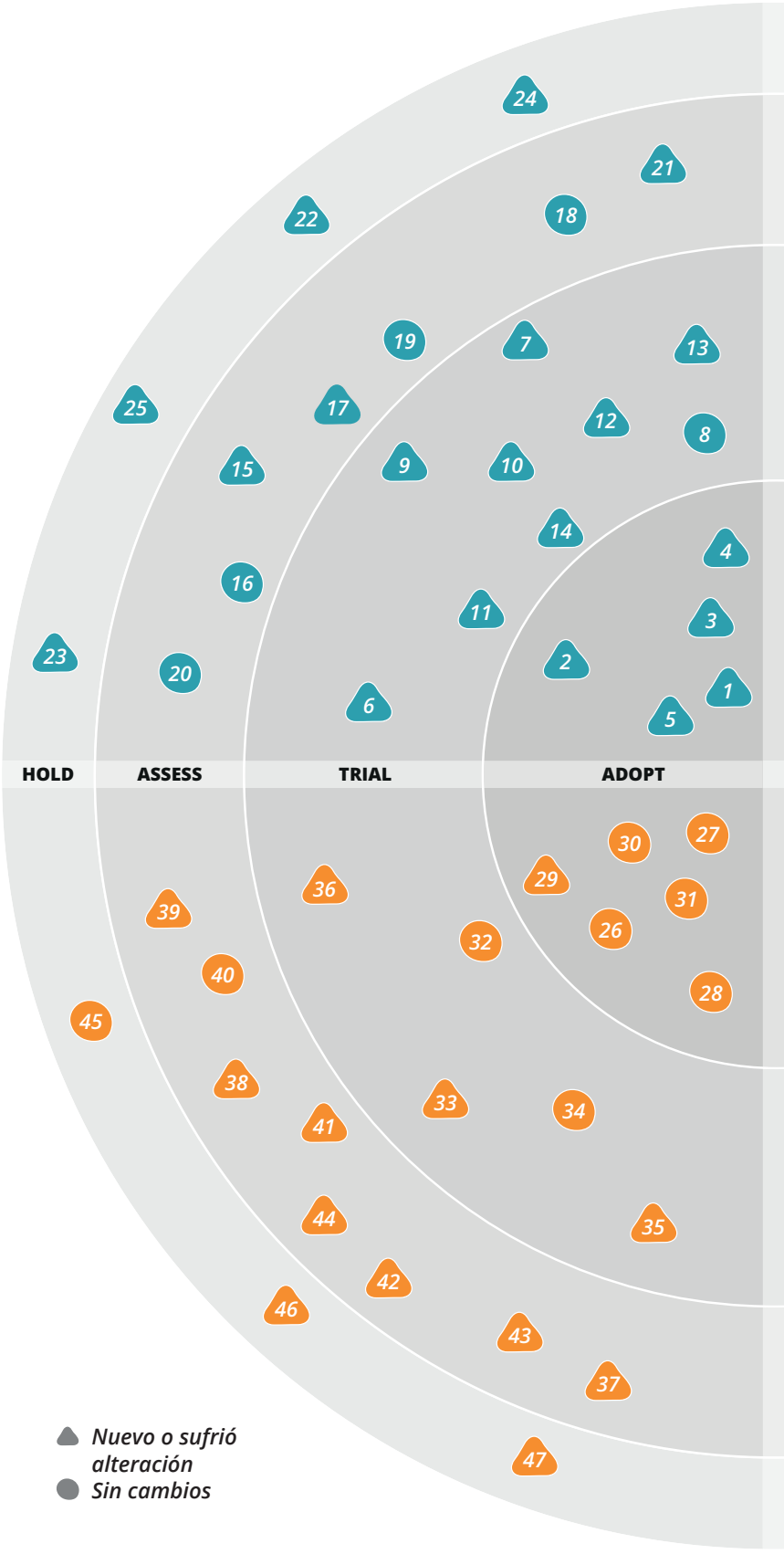
PLATAFORMAS

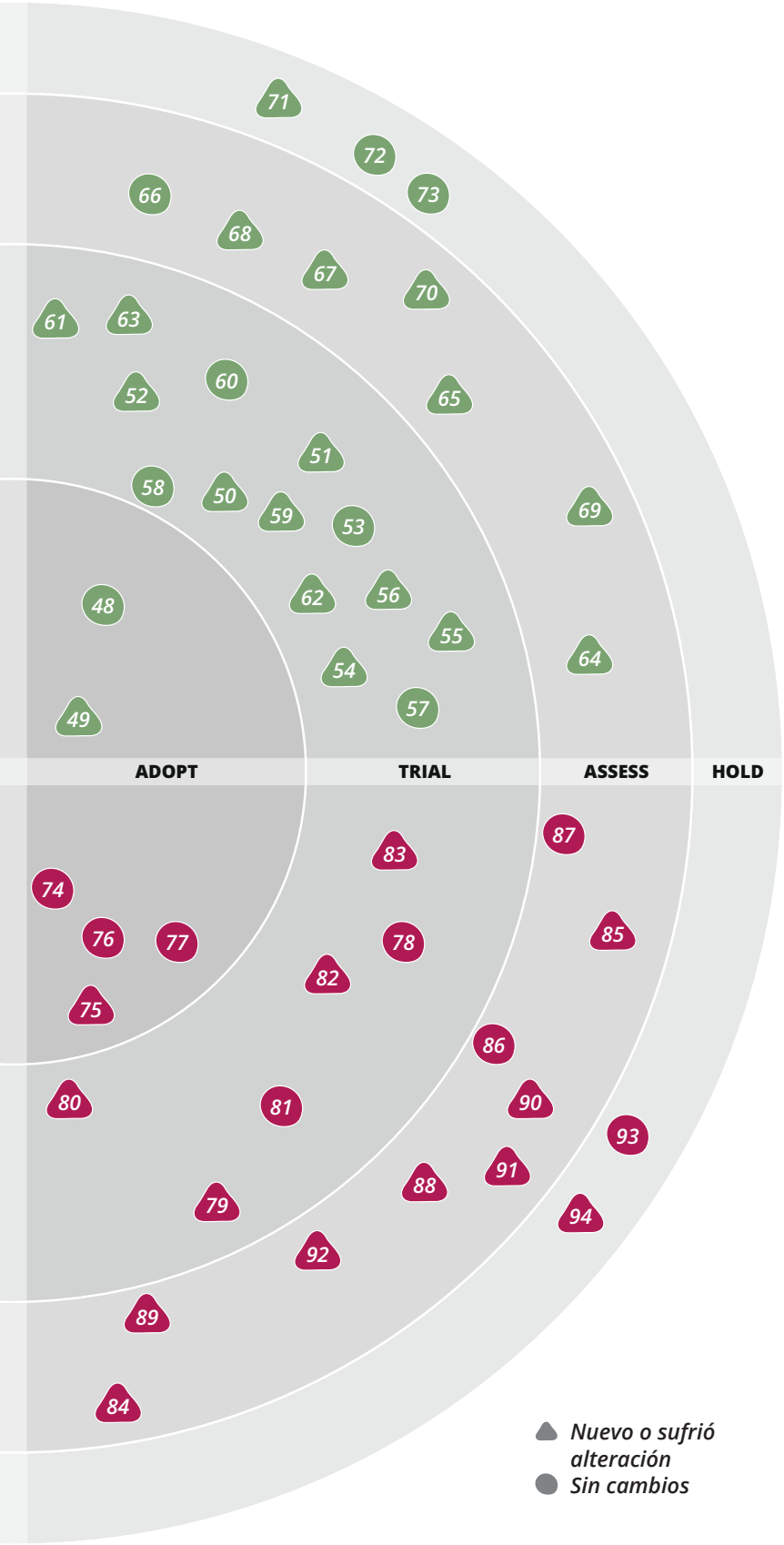
***ADOPT* - ADOPCIÓN**
26 Elastic Search
27 MongoDB
28 Neo4J
29 Node.js
30 Redis
31 SMS and USSD as a UI

***TRIAL* - ENSAYO**
32 Hadoop 2.0
33 Hadoop as a service
34 OpenStack
35 PostgreSQL for NoSQL
36 Vumi

***ASSESS* - EVALUACIÓN**
37 Akka
38 Backend as a service
39 Low-cost robotics
40 PhoneGap/Apache Cordova
41 Private Clouds
42 SPDY
43 Storm
44 Web Components standard

***HOLD* - ESPERA**
45 Big enterprise solutions
46 CMS as a platform
47 Enterprise Data Warehouse





O RADAR

HERRAMIENTAS

***ADOPT* - ADOPCIÓN**
48 D3
49 Dependency management for JavaScript

***TRIAL* - ENSAYO**
50 Ansible
51 Calabash
52 Chaos Monkey
53 Gatling
54 Grunt.js
55 Hystrix
56 Icon fonts
57 Librarian-puppet and Librarian-Chef
58 Logstash & Graylog2
59 Moco
60 PhantomJS
61 Prototype On Paper
62 SnapCI
63 Snowplow Analytics & Piwik

***ASSESS* - EVALUACIÓN**
64 Cloud-init
65 Docker
66 Octopus
67 Sensu
68 Travis for OSX/iOS
69 Visual regression testing tools
70 Xamarin

***HOLD* - ESPERA**
71 Ant
72 Heavyweight test tools
73 TFS

LENGUAJES Y FRAMEWORKS

***ADOPT* - ADOPCIÓN**
74 Clojure
75 Dropwizard
76 Scala, the good parts
77 Sinatra

***TRIAL* - ENSAYO**
78 CoffeeScript
79 Go language
80 Hive
81 Play Framework 2
82 Reactive Extensions across languages
83 Web API

***ASSESS* - EVALUACIÓN**
84 Elixir
85 Julia
86 Nancy
87 OWIN
88 Pester
89 Pointer Events
90 Python 3
91 TypeScript
92 Yeoman

***HOLD* - ESPERA**
93 Handwritten CSS
94 JSF

TÉCNICAS

La captura de los errores de JavaScript del lado del cliente ha ayudado a nuestros equipos a identificar cuestiones específicas de la configuración del navegador o de los plug-ins y dichas cuestiones son las que repercuten sobre la experiencia del usuario. Durante el último año, han surgido una serie de proveedores de servicios en respuesta a esta necesidad. En lugar de almacenar estos errores en la aplicación, las aplicaciones web pueden registrar estos datos en herramientas de análisis web o de monitoreo existentes tales como New Relic para reducir los requisitos de almacenamiento.

Desde el último radar, algunos avances han logrado que la **entrega continua** para las aplicaciones nativas en los **dispositivos móviles** fuera menos problemática. Xctool, la reciente "xcodebuild mejorada" con código abierto perfecciona la automatización iOS y las pruebas de unidad. La llegada de las actualizaciones automáticas en iOS7 reduce la fricción de las lanzamientos regulares. Travis-CI ahora soporta agentes OS X y así elimina otro obstáculo en la entrega continua para plataformas móviles. Nuestra recomendación del último radar acerca del valor de los enfoques híbridos y la importancia de la automatización de pruebas para dispositivos móviles aún sigue siendo válida.

A medida que las aplicaciones JavaScript del lado del cliente crecen en sofisticación, aumenta la necesidad de sofisticación en la ingeniería de modo que vaya a la par de este crecimiento. Una falla común en la arquitectura es el acceso sin restricciones al Modelo de Objetos del Documento (DOM, por sus siglas en inglés) a través del código (al combinar la manipulación del DOM con la lógica de las aplicaciones y las llamadas AJAX). Esto hace que el código sea difícil de entender y extender. Pensar en separación de responsabilidades es un antídoto muy útil. Esto implica limitar en forma estricta todos los accesos al DOM (lo cual, por lo general, se traduce en el uso de jQuery) a una cada delgada de segregación. Un efecto secundario positivo que resulta de este enfoque es que todo lo que quede fuera de esta capa de **DOM segregativa** puede probarse de manera rápida y aislada respecto del navegador mediante el uso de un motor JavaScript como **node.js**.

Cuando se utilizan técnicas tales como "instrumentar todas las cosas" y el registro semántico, puede resultar de gran ayuda la **captura explícita de los eventos de dominios**. Usted puede evitar tener que inferir cuáles son las intenciones del usuario detrás de las transiciones de estado al transformar la modificación de estas transiciones en preocupaciones de primera clase. Un método para lograr este resultado consiste en utilizar una arquitectura de origen de eventos con el fin de generar una correspondencia entre los eventos de la aplicación y los eventos relevantes del negocio.

Cada vez más, HTML es generado no solo en el servidor, sino también en el cliente, en el navegador web. En muchos casos, esta generación dividida seguirá siendo necesaria, pero con la creciente madurez de las bibliotecas de plantillas de JavaScript, un enfoque interesante se ha convertido en una opción posible: **generación del lado del cliente y del lado del servidor, con el mismo código**.

Usted no puede reaccionar a eventos importantes de negocio a menos que pueda monitorearlos. El principio **instrumentar todas las cosas** nos incita a pensar de manera proactiva acerca de cómo lograr este resultado al comienzo de nuestro desarrollo del software. Esto nos permite exponer métricas clave, monitorearlas y producir informes a partir de las mismas a los fines de mejorar la eficacia operativa.

Los servidores Chef y Puppet constituyen un lugar central para almacenar fórmulas o manifiestos que propaguen los

***ADOPT* - ADOPCIÓN**

1 Capturing client-side JavaScript errors
2 Continuous delivery for mobile devices
3 Mobile testing on mobile networks
4 Segregated DOM plus node for JS Testing
5 Windows infrastructure automation

***TRIAL* - ENSAYO**

6 Capture domain events explicitly
7 Client and server rendering with same code
8 HTML5 storage instead of cookies
9 Instrument all the things
10 Masterless Chef/Puppet
11 Micro-services
12 Perimeterless enterprise
13 Provisioning testing
14 Structured Logging

***ASSESS* - EVALUACIÓN**

15 Bridging physical and digital worlds with simple hardware
16 Collaborative analytics and data science
17 Datensparsamkeit
18 Development environments in the cloud
19 Focus on mean time to recovery
20 Machine image as a build artifact
21 Tangible interaction

***HOLD* - ESPERA**

22 Cloud lift and shift
23 Ignoring OWASP Top 10
24 Siloed metrics
25 Velocity as productivity

TÉCNICAS *continuación*

cambios de configuración a máquinas gestionadas. Además, conforman una base de datos central de información de nodos y ofrecen control de acceso para los manifiestos o fórmulas. La desventaja de estos servidores es que el flujo de su actividad se ve limitado en los casos en que múltiples clientes se conectan a ellos al mismo tiempo. Representan un punto único de falla y se esfuerzan por ser sólidos y confiables. Frente a esta situación, recomendamos **chef-solo o puppet independiente** junto con un sistema de control de versiones para los casos en los que el servidor se utilice principalmente para almacenar manifiestos o fórmulas. Los equipos siempre pueden implementar los servidores a medida que estos resultan necesarios o si se encuentran a sí mismos reinventando soluciones a problemas que los servidores ya han solucionado.

Cada vez más, se nos presentan oportunidades ilimitadas que surgen a partir de nuestra capacidad para conseguir y proveer hardware. Sin embargo, a pesar del aumento masivo en la flexibilidad que esto nos brinda, descubrimos que nos encontramos limitados por el alcance y la complejidad de los activos de software que se utilizan para administrar nuestros bienes virtuales. Por otro lado, el uso de técnicas de mayor familiaridad en el mundo del desarrollo de software, tales como TDD, BDD y CI, ofrece un enfoque para gestionar esta complejidad y nos brinda la confianza necesaria a la hora de efectuar cambios en nuestra infraestructura en forma segura, reiterada y que puede automatizarse. **El suministro de herramientas de pruebas** como rspec-puppet, Test Kitchen y serverspec se encuentra disponible para la mayoría de las plataformas.

El hecho de considerar los registros en términos de datos nos brinda una perspectiva más amplia en cuanto a la actividad operativa de los sistemas que creamos. **Los registros estructurados**, los cuales consisten en el uso de un formato de mensaje coherente y predeterminado que contenga información semántica, se construyen sobre esta técnica y les permiten a las herramientas como Greylog2 y Splunk plantear reflexiones más profundas.

La reducción de costos, el tamaño, el consumo de energía y la simplicidad de los dispositivos físicos han generado una explosión en los dispositivos que abren los dominios físicos al software. Por lo general, estos dispositivos no contienen mucho más que un sensor y un componente de comunicación, como Bluetooth Low Energy o WiFi. Como ingenieros de software, necesitamos ampliar nuestras ideas para incluir **la unión del mundo físico con el digital mediante hardware simple**. Ya podemos notar la presencia de este fenómeno en el auto, el hogar, el cuerpo humano, la agricultura y otros entornos físicos. El tiempo y los costos necesarios para realizar un prototipo de estos dispositivos disminuyen para ajustarse a las rápidas iteraciones posibles en software.

En nuestro afán por respaldar los modelos de negocio que cambian constantemente, aprender de las conductas pasadas y brindar la mejor experiencia para cada uno de los visitantes, sentimos la tentación de querer grabar la mayor cantidad de datos posible. Al mismo tiempo, los hackers están más feroces que nunca y protagonizan impresionantes violaciones de la seguridad sin descanso. A su vez, ahora nos enteramos de la existencia de una vigilancia masiva sin precedentes por parte de las agencias gubernamentales. El término **Datensparsamkeit** proviene de la legislación alemana en materia de privacidad y describe la idea de almacenar solo tanta información personal como sea absolutamente necesaria para la empresa o las leyes pertinentes. Algunos ejemplos de esto son, en lugar de almacenar la dirección IP completa del cliente en los registros de acceso, utilizar solamente los primeros dos o tres octetos y, en vez de registrar trayectos de tránsito con un nombre de usuario, utilizar un símbolo anónimo. Si usted nunca almacena la información, ya no tendrá que preocuparse por que alguien quiera robársela.

A medida que las fronteras entre hardware y software continúan desdibujándose, podemos observar cómo la informática tradicional se integra cada vez más con los objetos cotidianos. A pesar de que en la actualidad los dispositivos conectados están presentes en todos los locales comerciales, automóviles, casas y lugares de trabajo, seguimos sin comprender cómo combinarlos para lograr una experiencia informática útil que vaya más allá de una simple pantalla de cristal. **La interacción tangible** es una disciplina que combina tecnología, arquitectura, experiencia del usuario y diseño industrial de software y hardware. El objetivo es ofrecer entornos naturales conformados por objetos físicos donde los humanos puedan manipular y comprender los datos digitales.

Desafortunadamente, a medida que crece la adopción de la nube, observamos una tendencia a tratar la nube como un proveedor de alojamiento más. Esta tendencia de **auge y desplazamiento de la nube** es fomentada, lamentablemente, por grandes distribuidores que crean nuevas denominaciones para las ofertas de alojamiento ya existentes como “nubes”. Solo una mínima cantidad ofrece flexibilidad real o tarifas de pago por consumo. Si usted piensa que puede moverse a la nube sin diseñar una nueva arquitectura, es probable que no lo esté haciendo del modo adecuado.

Casi nunca pasa más de una semana sin que el sector informático sufra los efectos negativos de una nueva pérdida de datos de alto perfil, fuga de contraseñas o violaciones de un sistema presuntamente seguro. Existen muy buenos recursos que ayudan a garantizar que la seguridad sea abordada como una preocupación de primera clase durante el desarrollo de software y ya no podemos serles indiferentes. El **OWASP Top 10** es un buen punto de partida.

TÉCNICAS *continuación*

A medida que las empresas se trasladan a la plataforma en línea, hemos observado una tendencia a usar **métricas en silos**. Se implementan herramientas específicas para recopilar y exhibir métricas específicas: una herramienta para la vista de páginas y el comportamiento del navegador, otra para los datos operativos y otra para consolidar los mensajes de registro. Esto nos conduce a los silos de datos y a la necesidad de integrar una por una las herramientas con el fin de recopilar la inteligencia comercial que resulta imprescindible a la hora de dirigir una empresa. Esto constituye una división generada por herramientas en el dominio analítico y, en consecuencia, se producen daños en la capacidad del equipo de tomar decisiones. Una solución mucho más productiva es obtener una visión consolidada del análisis casi en tiempo real mediante el uso de paneles integrados que muestren información relevante de los dominios afectados por el factor tiempo y del equipo.

De todos los enfoques con los que podemos no estar de acuerdo, equiparar velocidad con productividad se ha convertido en un tema tan frecuente que pensamos que teníamos que abordarlo en nuestro círculo de espera. Cuando se la utiliza correctamente, la velocidad permite la incorporación del "clima de ayer" en el proceso de planificación de iteraciones. La velocidad es simplemente la capacidad estimada para un equipo dado en un momento dado. Puede mejorar a medida que se los miembros del equipo empiezan a integrarse o al arreglar problemas como las deudas técnicas o un servidor de cimpilación agrietado. No obstante, como todas las métricas, se puede hacer un uso incorrecto de la velocidad. Por ejemplo, los gerentes de proyectos que son demasiado entusiastas tienden a insistir en la mejora continua de la velocidad. El hecho de considerar la **velocidad en términos de productividad** genera conductas de equipo improductivas que optimizan las métricas a costa del funcionamiento real del software.

HERRAMIENTAS

Podemos observar que las organizaciones que han probado la infraestructura Hadoop de manera exitosa comienzan a consolidar sus servicios de infraestructura Hadoop en una plataforma centralizada y gestionada antes de ampliar este fenómeno a toda la empresa. Estas plataformas de **Hadoop como servicio** se caracterizan por el nivel de control que interactúa y se coordina entre distintos componentes centrales de la infraestructura de Hadoop. Por lo general, las capacidades de la plataforma se exponen a la empresa a través de abstracciones del más alto nivel. Este tipo de plataforma gestionada le ofrece a las organizaciones la posibilidad de instalar los procesos, la infraestructura y los conjuntos de datos de un modo bastante coherente en toda la organización. Estos servicios se construyen en centros de datos privados e infraestructura de nube pública.

Akka es un kit de herramientas y motor en tiempo de ejecución para la creación de aplicaciones en gran medida simultáneas, distribuidas, con tolerancia a fallos y dirigidas por eventos en la JVM. Ofrece procesos ligeros dirigidos por eventos con aproximadamente 2,7 millones de actores por GB de RAM y un modelo de tolerancia a fallos "let-it-crash" diseñado para trabajar en un entorno distribuido. Akka puede utilizarse como una biblioteca para aplicaciones web o como un núcleo independiente en el que pueden almacenarse las aplicaciones.

El reciente auge de productos orientados a los dispositivos móviles, sumado a la incorporación generalizada de enfoques "Lean Start-up" que aproximan con excelencia las nuevas ideas a su lanzamiento al mercado, han generado un ecosistema de ofertas **Backend como servicio** (BaaS, por sus siglas en inglés) que les permiten a los desarrolladores concentrarse en la aplicación del cliente, y al mismo tiempo, liberarse de las preocupaciones del backend. Evalúe incorporar estos servicios a su kit de herramientas para aquellos casos en que resulta muy importante poner a prueba un producto nuevo de forma rápida y económica. Nuestras recomendaciones habituales para las decisiones en materia de crear/comprar/prestar aún siguen siendo válidas: determine con claridad cuáles son las áreas funcionales que resultan estratégicas para su empresa y cuáles representan insumos básicos. Para áreas potencialmente estratégicas, asegúrese de planificar una ruta de migración que le permitará a suproveedor BaaS iniciar rápidamente y, a su vez, evitar las fricciones que se generan cuando su arquitectura

evoluciona y surge la necesidad de migrarla para obtener esta funcionalidad y personalizarla como un factor de diferenciación.

Dado que el costo de los robots industriales disminuye y su seguridad y facilidad de uso aumentan, el mundo de la robótica útil y comercial se está abriendo cada vez más. Los robots como Baxter* de Rethink Robotics o U5 de Universal Robotics hacen que sea posible para las pequeñas o medianas empresas automatizar las tareas repetitivas que solían llevar a cabo los humanos. Cada vez más, el software empresarial tendrá que integrarse con la **robótica de bajo costo** como un participante más en el flujo de valor. El desafío radica en hacer que la experiencia resulte sencilla y productiva también para los compañeros de trabajo humanos.

La necesidad de almacenamiento físico de datos dentro de las naciones u organizaciones ha aumentado de manera significativa en los últimos años. Existe una preocupación alrededor de la confidencialidad de los datos alojados en los entornos de nube. Las organizaciones empiezan a ver las **nubes privadas** como una alternativa en los casos en que los datos

***ADOPT* - ADOPCIÓN**

48 D3
49 Dependency management for JavaScript

***TRIAL* - ENSAYO**

50 Ansible
51 Calabash
52 Chaos Monkey
53 Gatling
54 Grunt.js
55 Hystrix
56 Icon fonts
57 Librarian-puppet and Librarian-Chef
58 Logstash & Graylog2
59 Moco
60 PhantomJS
61 Prototype On Paper
62 SnapCI
63 Snowplow Analytics & Piwik

***ASSESS* - EVALUACIÓN**

64 Cloud-init
65 Docker
66 Octopus
67 Sensu
68 Travis for OSX/iOS
69 Visual regression testing tools
70 Xamarin

***HOLD* - ESPERA**

71 Ant
72 Heavyweight test tools
73 TFS

HERRAMIENTAS *continuación*

necesiten alojarse conservando el control sobre el acceso y la distribución. La nube privada ofrece una infraestructura de nube habilitada para el uso exclusivo de una sola organización con las siguientes características: auto servicio bajo demanda, amplio acceso de red, conjunto de recursos disponibles, rápida elasticidad y un servicio medido.

SPDY es un protocolo de red abierto para el transporte de baja latencia de contenido web propuesto para HTTP2 que ha observado un aumento en la compatibilidad de navegadores modernos. SPDY reduce el tiempo de carga de páginas al priorizar la transferencia de subrecursos, de modo que solo se necesita una conexión por cliente. La seguridad de la capa de transporte se utiliza en las implementaciones SPDY con los encabezados de transmisión gzip o la deflación del texto comprimido en lugar del texto legible por los humanos en HTTP. Es ideal para entornos de alta latencia.

Las cantidades heterogéneas y extremadamente grandes de datos no representan el único tema vinculado con los grandes datos. En ciertas circunstancias, la velocidad del procesamiento puede ser tan importante como el volumen. **Storm** es un sistema de computación distribuido en tiempo real. Cuenta con una escalabilidad semejante a la de Hadoop y un rendimiento tan rápido que alcanza un millón de tuplas por segundo. Además, permite el procesamiento en tiempo real de lo que Hadoop realiza por lotes.

En el último radar, alertamos sobre el uso de frameworks tradicionales de componentes web que proporcionan un modelo de componente del lado del servidor. El **estándar de componentes web** que se originó en Google es algo muy diferente. Proporciona una forma sencilla de crear widgets reciclables al contribuir con el aislamiento de HTML, CSS y JavaScript, por lo que no interfieren con el resto de la página y, al mismo tiempo, la página no interfiere con ellos. Los desarrolladores pueden utilizar lo mucho o poco del marco que sea necesario. Polymer Project ofrece soporte temprano.

Al tiempo que la integración centralizada de datos para la elaboración de análisis e informes continúa siendo una buena estrategia, las iniciativas tradicionales del **Almacén de datos empresariales** (EDW, por sus siglas en inglés) presentan una tasa de fallas superior al 50 %. Las grandes modificaciones de datos por adelantado traen, como consecuencia, almacenes sobredimensionados que tardan años en completarse y son muy costosos de mantener. En esta edición del radar, dejamos estas técnicas y EDW antiguos en espera. En cambio, recomendamos evolucionar hacia un EDW. Pruebe y aprenda al lograr incrementos pequeños pero valiosos que con frecuencia se liberan a la producción. Las herramientas y técnicas no tradicionales pueden ayudar. Por ejemplo, se puede utilizar un diseño de esquema Data Vault o incluso un almacenamiento de documentos NoSQL como HDFS.

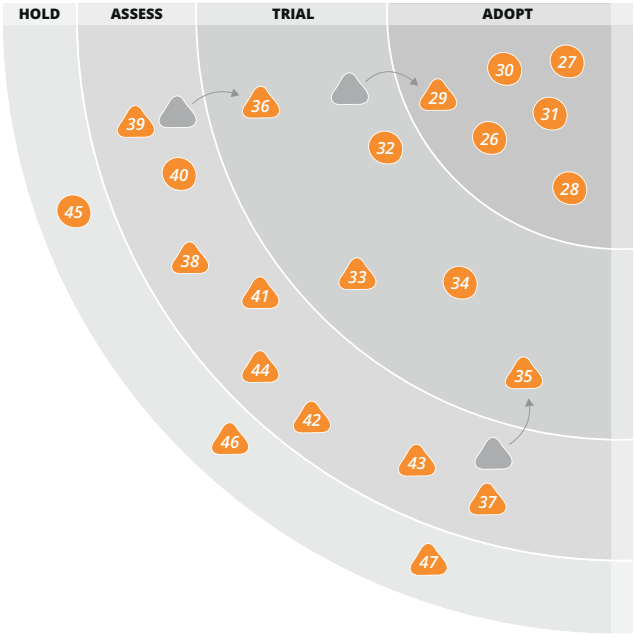
Los sistemas de gestión de contenidos (CMS, por sus siglas en inglés) tienen su lugar. En muchos casos, no tiene sentido comenzar la redacción y la funcionalidad de flujo de trabajo de cero. No obstante, hemos experimentado graves problemas cuando un **CMS como plataforma** se convierte en una solución informática que crece más allá de la gestión de contenido simple.

PLATAFORMAS

El uso de las herramientas de **gestión de dependencias para JavaScript** ha ayudado a nuestros equipos a administrar grandes cantidades de JavaScript al estructurar su código y cargar las dependencias en el tiempo de ejecución. A pesar de que esto simplificó los esfuerzos en la mayoría de los casos, las cargas diferidas complican la compatibilidad con el modo fuera de línea. Las diferentes herramientas de gestión de dependencias cuentan con distintas fortalezas, por lo que deberá elegir según su contexto.

En la categoría de motores de orquestación DevOps, **Ansible** ha recibido un reconocimiento casi mundial dentro de los proyectos de ThoughtWorks. Posee herramientas y abstracciones útiles en un nivel práctico de granularidad.

En los proyectos móviles, nos ha sorprendido la funcionalidad y la evolución gradual de las capacidades y la madurez de **Calabash**. Se trata de una herramienta de pruebas de aceptación automatizada tanto para aplicaciones Android como iOS que admite herramientas de entornos comunes como Cucumber. Constituye una elección atractiva para los proyectos heterogéneos.



Según nuestra recomendación del último radar acerca de tener en cuenta un enfoque en la reducción del tiempo medio de recuperación, deseamos destacar a **Chaos Monkey** de la serie Simian Army de Netflix. Se trata de una herramienta que desactiva en forma aleatoria determinadas instancias en el entorno de producción durante la operación normal. Cuando se ejecuta acompañada de un monitoreo integral y el respaldo de un equipo, ayuda a descubrir debilidades inesperadas en el sistema. A su vez, esto le permite al equipo de desarrollo construir mecanismos de recuperación automática con antelación y, de esta manera, no tiene que luchar por responder ante un apagón que toma a todos por sorpresa.

Varios de nuestros equipos de ThoughtWorks que desarrollan aplicaciones Node.js utilizan **Grunt** para automatizar muchas de las actividades de desarrollo como minificación, compilación y linting. Una gran cantidad de las tareas comunes se encuentran disponibles como plugins de Grunt. Si es necesario, incluso puede generar la configuración mediante programación.

La gestión de la red de dependencias en un sistema distribuído es complicado y representa un problema que cada vez más personas deben enfrentar con el traspaso a los microservicios más detallados. **Hystrix** es una biblioteca para JVM de Netflix que implementa patrones para hacer frente a las fallas posteriores, ofrece monitoreo de las conexiones en tiempo real y mecanismos de caché y de preparación por lotes para lograr que las dependencias de servicios internos sean más eficientes. Combinada con el panel hystrix y Turbine, esta herramienta se puede utilizar para crear sistemas más resistentes y proporcionar datos casi en tiempo real sobre tolerancia de rendimiento, latencia y fallas.

Las pruebas de los microservicios basados en HTTP pueden ser difíciles y problemáticas. Esto sucede en dos escenarios en particular: el consumo de un grupo de microservicios front-end y la comunicación entre los microservicios. Para abordar estos escenarios, **Moco** puede ser muy útil. Se trata de un marco de stub ligero para la prueba de puntos de acceso HTTP. Permite obtener un servicio de stub embebido que con 2 líneas de código Java o Groovy se puede ejecutar, o un servicio independiente que con pocas líneas de JSON permita describir el comportamiento requerido.

Siempre hemos defendido el uso de prototipos hechos a mano y de baja fidelidad para ilustrar las interacciones del

***ADOPT* - ADOPCIÓN**
26 Elastic Search
27 MongoDB
28 Neo4J
29 Node.js
30 Redis
31 SMS and USSD as a UI

***TRIAL* - ENSAYO**
32 Hadoop 2.0
33 Hadoop as a service
34 OpenStack
35 PostgreSQL for NoSQL
36 Vumi

***ASSESS* - EVALUACIÓN**
37 Akka
38 Backend as a service
39 Low-cost robotics
40 PhoneGap/Apache Cordova
41 Private Clouds
42 SPDY
43 Storm
44 Web Components standard

***HOLD* - ESPERA**
45 Big enterprise solutions
46 CMS as a platform
47 Enterprise Data Warehouse

PLATAFORMAS *continuación*

usuario sin quedar atrapado en los detalles del diseño gráfico. **Prototype On Paper** es una herramienta que permite capturar las maquetas individuales elaboradas en papel a través de una cámara con iOS o Android y vincularlas para permitir las pruebas de interacción del usuario. Esto acorta la distancia entre los prototipos estáticos de baja fidelidad y las técnicas de prototipos de una mayor fidelidad.

Mencionamos **SnapCI** de ThoughtWorks (un servicio alojado que brinda herramientas de implementación) en la última edición del Radar. Desde ese momento, hemos observado que muchos equipos utilizan SnapCI con éxito en sus proyectos. Si necesita una solución de distribución simple y constante en la nube, SnapCI puede brindársela con un solo clic. Sin hardware, sin problemas.

Dado el creciente control de la privacidad de los datos, cada vez son más las empresas que se preocupan a la hora de compartir los análisis web con terceros. **Snowplow Analytics y Piwik** son ejemplos de plataformas de análisis de código abierto que pueden alojarse a sí mismas y proporcionar un conjunto de funciones y planes de trabajo prometedores.

Cloud-init constituye una técnica simple pero poderosa para llevar a cabo acciones en una instancia en la nube al momento de inicio. Resulta especialmente útil cuando se la utiliza con metadatos de instancia para permitir que una instancia que recién se inicia extraiga la configuración, las dependencias y el software necesarios para llevar a cabo un rol en particular. Si se utiliza junto con los patrones de servidores Immutable o Phoenix, se puede crear un mecanismo muy receptivo y ligero para la gestión de implementaciones en la nube.

El proyecto de código abierto **Docker** ha despertado gran interés dentro de ThoughtWorks y está creciendo en ímpetu y madurez. Docker permite que las aplicaciones se envasen y publiquen como contenedores ligeros y portátiles que pueden ejecutarse del mismo modo en una computadora portátil o en un conjunto de producción. A su vez, proporciona las herramientas para la creación y gestión de contenedores de aplicaciones, así como un entorno de ejecución basado en LXC (LinuX Containers).

Muchas de las herramientas de monitoreo se elaboran a partir de la idea de la máquina. Monitoreamos lo que hace la máquina y el tipo de software que está ejecutando. Si hablamos de infraestructura basada en la nube, se trata de un enfoque problemático, en especial con patrones como los servidores Immutable y Phoenix. Las máquinas van y vienen, pero lo que en verdad importa es que los servicios sigan funcionando. **Sensu** le permite a una máquina registrarse para desarrollar un determinado rol y luego Sensu la monitorea a partir de esa base. Una vez que terminamos con la máquina, podemos simplemente borrar su registro.

Todos los avances para iOS deben llevarse a cabo en OS X. Debido a las restricciones técnicas y de licencias, la ejecución de torres de servidores con OS X no es una opción común ni sencilla. A pesar de estas dificultades, **Travis CI**, con el soporte de Sauce Labs, ahora brinda servicios de integración continua basados en la nube para proyectos iOS y OS X.

La creciente complejidad en las aplicaciones web ha aumentado la concientización respecto de que la apariencia también debe ser probada además de la funcionalidad. Esto ha provocado el surgimiento de una variedad de **herramientas de pruebas de regresión visual**, incluidas CSS Critic, dpxdt, Huxley, PhantomCSS y Wraith. Las técnicas abarcan desde las afirmaciones directas de valores CSS hasta las comparaciones reales de las capturas de pantalla. A pesar de que este es un campo que aún se encuentra en desarrollo activo, creemos que las pruebas de regresión visual deben añadirse a los procesos de distribución continua.

Entre las múltiples elecciones disponibles para crear aplicaciones móviles en todas las plataformas, **Xamarin** ofrece un conjunto de herramientas realmente único. Admite C# y F# como el principal lenguaje con un vínculo a los SDK específicos de la plataforma y el entorno de ejecución Mono que trabaja mediante iOS, Android y Windows Phone. Las aplicaciones se redactan en el código nativo y no a partir del tradicional enfoque de todas las plataformas que presentan las interfaces de usuario basadas en HTML en un navegador integrado. Esto brinda una apariencia y una percepción más nativa a la aplicaciones. Cuando se utiliza este conjunto de herramientas, es indispensable que el nivel de la interfaz de usuario específica de la plataforma se separe del resto de los niveles para garantizar la reutilización del código en las diversas plataformas. La aplicación binaria tiende a ser un poco más grande debido al entorno de ejecución que se incluye.

Seguimos observando cómo los equipos invierten esfuerzos significativos en la creación de scripts no sostenibles **Ant** y Nant. Estos son difíciles de comprender y ampliar debido a la inherente falta de expresividad y modularidad clara que proporcionan las herramientas. Alternativas como Gradle, Buildr y PSake han demostrado, sin dudas, un mantenimiento y una productividad superiores.

LENGUAJES Y FRAMEWORKS

Scala es un lenguaje de gran tamaño que es popular gracias a su accesibilidad para los desarrolladores nuevos. Este abanico de caracteristicas es un problema, ya que muchos aspectos de Scala como: las conversiones implícitas y el dinamismo, pueden generar inconvenientes. Para utilizar Scala de manera exitosa, es preciso investigar el lenguaje y formar una opinión sólida acerca de cuáles son las partes adecuadas para usted y así crear su propia definición de las partes buenas de Scala. Puede desactivar las partes con las que no desee mediante el uso de un sistema denominado “banderas de funciones”.

El **lenguaje Go** fue desarrollado originalmente por Google como un lenguaje de programación de sistemas para reemplazar C & C++. En un período de cuatro años, Go sigue ganando terreno en otras áreas. La combinación de códigos binarios muy pequeños, vinculados estáticamente entre sí junto con una excelente biblioteca HTTP significa que Go ha ganado gran popularidad entre las organizaciones mediante el uso de arquitecturas de microservicios más granulares.

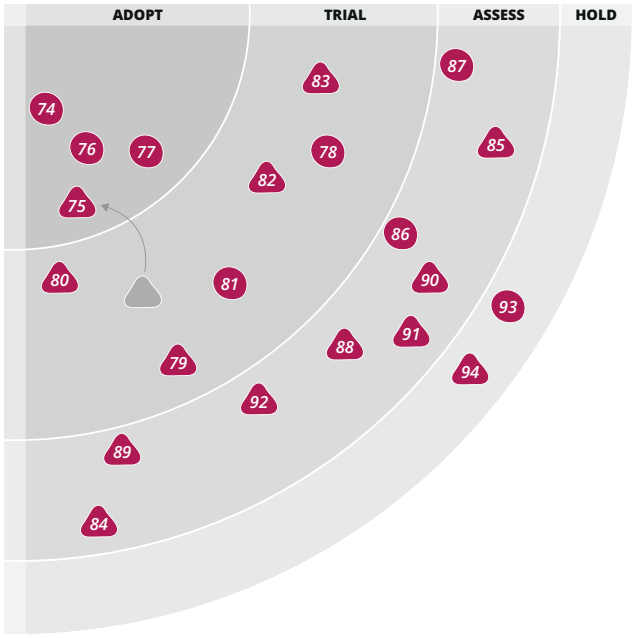
Hive es un almacén de datos construido sobre Hadoop que proporciona un lenguaje de consulta y definición de datos tipo SQL que transforma las consultas en trabajos MapReduce que pueden ejecutarse a lo largo de todo el conjunto Hadoop. Al igual que todas las abstracciones útiles, Hive no intenta negar la existencia de los mecanismos fundamentales de Hadoop y admite operaciones personalizadas de map-reduce como un poderoso mecanismo de extensión. Más allá de las semejanzas superficiales con SQL, Hive no pretende reemplazar los motores de consulta de baja latencia, en tiempo real que se pueden encontrar en los sistemas de base de datos relacionales. Bajo ningún punto de vista recomendamos el uso de Hive con fines de consulta en línea ad hoc.

La situación de **Play Framework 2** ha generado una gran discusión interna. Existen sugerencias acerca de si resulta conveniente desplazar este framework hasta que se adapte o dejarlo en espera Estas diferencias se remiten, principalmente, a las aplicaciones específicas para las que se usa, cómo se usa y cuáles son las expectativas que la gente tiene acerca del mismo. A pesar de que ninguna de estas cuestiones son únicas para Play, esto ha generado más polémica de la habitual en la biblioteca estándar frente al debate del framework. Reiteramos las precauciones que se establecieron en el último radar y seguiremos controlando la manera en que Play continúa madurando para mantener su nivel óptimo.

La programación reactiva trata con flujos o valores que cambian con el tiempo. El uso de flujo de datos, concurrencia implícita y propagación transparente de eventos, permiten una gestión de eventos a gran escala con un alto grado de eficiencia y baja latencia. En el último radar, mencionamos Reactive Extensions en .NET debido al trabajo exhaustivo que llevó a cabo Microsoft al hacer de Rx una parte central del framework .NET. Desde entonces, con la adopción de la librería Reactive Cocoa para Objective C, el puerto Java de Reactive Extensions, la librería React JavaScript, el lenguaje Elm basado en el lenguaje Haskell y la librería Flapjax JavaScript, ampliamos este fenómeno para incluir **Reactive Extensions en todos los lenguajes**.

Hasta hace poco tiempo, la **API para Web** de Microsoft era la opción “menos mala” para la elaboración de un servicio RESTful a través de ASP.NET. Web API 2 corrige una gran cantidad de situaciones problemáticas mediante un mejor soporte de enrutamiento flexible, subrecursos, tipos de medios y métodos de prueba mejorados. Continúa siendo nuestra librería preferida a la hora de crear .NET REST APIs.

Elixir es un lenguaje de programación dinámico, funcional y homoicónico creado a partir de la máquina virtual Erlang con un poderoso macrosistema que lo hace ideal para la construcción



***ADOPT* - ADOPCIÓN**
74 Clojure
75 Dropwizard
76 Scala, the good parts
77 Sinatra

***TRIAL* - ENSAYO**
78 CoffeeScript
79 Go language
80 Hive
81 Play Framework 2
82 Reactive Extensions across languages
83 Web API

***ASSESS* - EVALUACIÓN**
84 Elixir
85 Julia
86 Nancy
87 OWIN
88 Pester
89 Pointer Events
90 Python 3
91 TypeScript
92 Yeoman

***HOLD* - ESPERA**
93 Handwritten CSS
94 JSF

LINGUAGENS & FRAMEWORKS *continuación*

de lenguajes de dominio específico. Elixir cuenta con funciones distintivas tales como el operador Pipe, el cual les permite a los desarrolladores construir un pipeline de funciones como las que uno generaría en el shell de comando UNIX. El byte code compartido le permite a Elixir operar internamente con Erlang y aprovechar las librerías existentes al tiempo que admite herramientas como la herramienta de construcción Mix, el shell interactivo lex y el framework de prueba de unidad ExUnit. Se trata de una alternativa práctica para Erlang en la construcción de lenguajes de dominio específicos, o DSL, por sus siglas en inglés.

Julia es un lenguaje de programación dinámico, de procedimiento y homoicónico que fue diseñado con el objetivo de cubrir las necesidades de computación científica de alto rendimiento . La implementación del lenguaje se organiza en torno al concepto de funciones genéricas y métodos de distribución dinámicos. Los programas Julia son, en su mayoría, funciones que pueden contener múltiples definiciones para diferentes combinaciones de tipos de argumentos. La combinación de estas características de los lenguajes y el compilador en tiempo real basado en LLVM contribuyen a que Julia alcance un rendimiento de alto nivel. Además, Julia admite un entorno de multiprocesamiento basado en el intercambio de mensajes que permite que los programas se ejecuten en procesos múltiples. Esto les permite a los programadores crear programas distribuidos basados en cualquiera de los modelos de programación paralela.

PowerShell continúa siendo una opción muy utilizada para lograr una automatización de nivel bajo en máquinas Windows. **Pester** constituye una librería de pruebas que posibilita la ejecución y validación de los comandos PowerShell. Pester simplifica las pruebas de scripts durante el desarrollo gracias a un poderoso sistema de simulación (mocking) que posibilita la definición de stubs y dobles en las pruebas. Las pruebas de Pester también pueden añadirse a un sistema de integración continua para prevenir los defectos de regresión.

Python 3 representó un cambio importante desde su predecesor Python 2.x que introdujo modificaciones incompatibles con versiones anteriores. Este fue notable porque eliminaba las características de los lenguajes para que su uso resultara más fácil y coherente, sin disminuir su poder. Esto ha generado problemas en la adopción, ya que algunas de las librerías de soporte en las que las personas más confiaban no fueron tomadas en cuenta y con frecuencia los desarrolladores de Python deben encontrar nuevas formas de hacer las cosas. Aún así, es importante reconocer el gran esfuerzo hacia la elaboración de un lenguaje más simple y, si usted basa su desarrollo en Python en forma activa, eche un nuevo vistazo a Python 3.

Luego de algunas demoras, causadas principalmente por los reclamos de patentes por parte de Apple, el consorcio W3C ya ha finalizado las recomendaciones Touch Events. No obstante, el que parece tomar gran impulso es **Pointer Events**, un estándar más nuevo, amplio y enriquecido. Recomendamos tener en cuenta Pointer Events para las interfaces HTML que deban funcionar a través de diferentes métodos de entrada.

TypeScript representa un enfoque interesante al momento de introducir un nuevo lenguaje de programación en el navegador. Con TypeScript, las nuevas características del lenguaje compilan hacia el JavaScript normal, y a pesar de que representa un conjunto superior de JavaScript, no se percibe como un lenguaje del todo nuevo. No representa una propuesta de tómelo o déjelo y no relega a JavaScript a una plataforma de ejecución intermedia. Muchas de las características del lenguaje se basan en extensiones futuras planificadas de JavaScript.

Yeoman intenta que los desarrolladores de aplicaciones web puedan aumentar su productividad a través de la simplificación de actividades como anclajes, elaboración y gestión del paquete. Se trata de una colección de las herramientas Yo, Grunt y Bower que funcionan muy bien en conjunto.

Seguimos observando cómo los equipos enfrentan diversos inconvenientes al utilizar **JSF** (JavaServer Faces) y por lo tanto recomendamos que evite esta tecnología. Parece ser que los equipos eligen JSF porque es un estándar J2EE y no evalúan realmente si el modelo de programación les resulta conveniente o no según sus necesidades. Creemos que JSF tiene tantos defectos porque intenta abstraer HTML, CSS y HTTP, lo cual representa exactamente lo opuesto a lo que hace el resto de lps frameworks web modernos. JSF, al igual que los formularios web ASP.NET, intenta crear estados sobre el protocolo sin estado HTTP y esto termina causando muchos problemas relacionados con el estado compartido del lado del servidor. Si bien reconocemos que se presentaron mejoras en JSF 2.0, pensamos que la base fundamental del modelo está rota. Nuestras recomendaciones para los equipos es que utilicen frameworks simples y que adopten y comprendan las tecnologías web que incluyan HTTP, HTML y CSS.

REFERENCIAS

Tangible Interactions
http://www.interaction-design.org/encyclopedia/tangible_interaction.html
http://www.computer.org/csdl/mags/co/2013/08/mco2013080070-abs.html
http://www.theverge.com/2012/9/21/3369616/co-working-robots-baxter-home
http://robohub.org/rethink-robotics-baxter-and-universal-robots-ur5-and-ur10-succeeding/

Web Components standard
http://www.polymer-project.org

Hystrix
https://github.com/Netflix/Hystrix/wiki
https://github.com/Netflix/Hystrix/tree/master/hystrix-dashboard
https://github.com/Netflix/Turbine/wiki

Reactive extensions en todos los lenguajes
https://github.com/blog/1107-reactivecocoa-for-a-better-world
http://facebook.github.io/react/
http://techblog.netflix.com/2013/02/rxjava-netflix-api.html
http://elm-lang.org/
http://www.flapjax-lang.org/

Pointer Events
http://www.w3.org/TR/pointerevents/
http://www.w3.org/TR/touch-events/
http://www.w3.org/2012/te-pag/pagreport.html
http://msopentech.com/blog/2013/06/17/w3c-pointer-events-gains-further-web-momentum-with-patch-for-mozilla-firefox

Sobre ThoughtWorks - Es una consultora global, empresa de productos de software y una comunidad de personas apasionadas cuyo propósito es revolucionar el desarrollo y creación de software, promoviendo impacto social positivo en los países y comunidades donde actua. Su división de productos, ThoughtWorks Studios, desarrolla herramientas pioneras para equipos de software - tales como Mingle ®, Go ™ y Twist ®, y que ayudan a las organizaciones a colaborar y entregar software de calidad. Los clientes de ThoughtWorks son organizaciones com misiones ambiciosas que buscan abordajes y tecnologías innovadoras como forma de alcanzar sus objetivos. Con 20 años de experiencia en el mercado, ThoughtWorks tiene más de 2.500 empleados - los "ThoughtWorkers" - atendiendo clientes en oficinas de Sudáfrica, Alemania, Australia, Brasil, Canada, China, Estados Unidos, Ecuador, India, Inglaterra, Singapur y Uganda.

COLABORADORES - Los colaboradores que integran la Comisión Asesora de Tecnología de ThoughtWorks son los siguientes:

Rebecca Parsons (CTO)
Martin Fowler (Cientista Chefe)
Badri Janakiraman
Brain Leke
Claudia Melo
Darren Smith
Erik Doernenburg
Evan Bottcher
Hao Xu
Ian Cartwright
James Lewis
Jeff Norris
Jonny LeRoy
Mike Mason
Neal Ford
Rachel Laycock
Sam Newman
Scott Shaw
Srihari Srinivasan
Thiyagu Palanisamy