ThoughtWorks®

# Technology Radar

## LO QUE PENSAMOS DE JAVASCRIPT, APIS, LEY DE CONWAY, RE-DESCENTRALIZACIÓN Y MUCHO MÁS

***JULIO 2014***

thoughtworks.com/radar

# NOVEDADES

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

## INCORPÓRESE AL MUNDO JAVASCRIPT

Creíamos que la tasa de cambio en el espacio de código abierto de Ruby era rápida hasta que llegaron los frameworks veloces de JavaScript. JavaScript solía ser una tecnología de condimento, utilizada siempre para acrecentar otras tecnologías. Si bien aún mantiene esa función, ahora amplió su rol en su propia plataforma con una tasa de cambio asombrosa. Intentar entender la magnitud de este espacio es totalmente abrumador y la innovación es increíble. Al igual que los espacios de códigos abiertos Java y Ruby, esperamos que en algún momento se calme, al menos un poco.

## LEY DE CONWAY

La ley de Conway, que establece que "las organizaciones que diseñan sistemas … se limitan a elaborar diseños de aplicaciones que son copias de las estructuras de comunicación de dichas organizaciones", sigue apareciendo en lugares inesperados. Uno de los principios clave del Agile Manifesto es "Las personas por sobre los procesos y las herramientas" y podemos ver cómo la ley de Conway refuerza esta idea, de un modo positivo aunque también negativo. Algunas empresas están inmersas en estructuras de silos que agregan una fricción innecesaria a los esfuerzos de ingeniería, al mismo tiempo que otras empresas más informadas usan la organización de equipos para impulsar los tipos de arquitectura que desean. Estamos en medio del aprendizaje sobre los peligros que corremos si ignoramos la ley de Conway y los beneficios que obtenemos si la aprovechamos.

## LOS MICROSERVICIOS Y EL AUGE DE API

Podemos observar un gran interés en la arquitectura de los microservicios y también una especial atención sobre la importancia de la API dentro de una organización y como un puente hacia el mundo real. En una arquitectura de microservicios, gran cantidad de servicios muy pequeños se implementan y están vinculados a la creación de sistemas, con los servicios asociados estrechamente a conceptos y valores comerciales. Para lograr que este enfoque funcione, los equipos necesitan una buena disciplina en torno a la creación, pruebas, integración y, luego, gestión de los servicios. Esta edición del Radar analiza algunas de las herramientas y técnicas específicas para los microservicios.

## NUEVA DESCENTRALIZACIÓN

Internet comenzó su historia como un sistema distribuido, pero durante la última década o más, hemos visto una gran centralización de servicios y de datos. A modo de ejemplo, más del 90 % de los correos electrónicos de todo el mundo se transmiten solamente a través de 10 proveedores. Algo similar sucede con la computación en la nube, ya que solo una reducida cantidad de proveedores brindan servicios a la gran mayoría de nuestras necesidades en la nube. Podemos observar una nueva tendencia de "volver a descentralizar" los datos y la infraestructura, impulsada en parte por las revelaciones acerca del dominio de los EE. UU. sobre la infraestructura de Internet y también por el deseo de mantener un control más individual y organizativo.

# ABOUT THE TECHNOLOGY RADAR

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

# COLABORADORES

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

Rebecca Parsons (CTO)
Martin Fowler(Chief Scientist)
Badri Janakiraman
Brain Leke
Claudia Melo
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

# EL RADAR

## TÉCNICAS

***ADOPT* - ADOPCIÓN**
1. Confidencialidad Directa Perfecta
2. DOM segregada y uso de Node para pruebas de JS

***TRIAL* - ENSAYO**
3. Captura Explícita de Eventos de Dominio
4. Entornos de desarrollo en la nube
5. Origen de Eventos
6. Enfoque en el tiempo medio de recuperación
7. Registro humanizado de (micro)servicio
8. Maniobra inversa de Conway
9. Guías de estilo dinámicas CSS
10. Imagen de máquina como artefacto de construcción
11. Masterless Chef/Puppet
12. Empresa sin límites
13. Pruebas de aprovisionamiento
14. Monitorización de usuarios reales
15. REST sin PUT
16. Logging estructurado
17. Plantilla de Servicio Adaptada

***ASSESS* - EVALUACIÓN**
18. Unión del mundo físico con el digital mediante hardware simple
19. Datensparsamkeit
20. Pipeline de imagenes de maquinas
21. Estrategia de aplicaciones mediante la agregación de niveles
22. Pruebas unitarias basadas en propiedades
23. Interacción tangible

***HOLD* - ESPERA**
24. Auge y desplazamiento de la nube
25. DevOps como un equipo
26. Ignorar OWASP Top 10
27. Las pruebas como una organización independiente
28. Velocidad en terminos de productividad

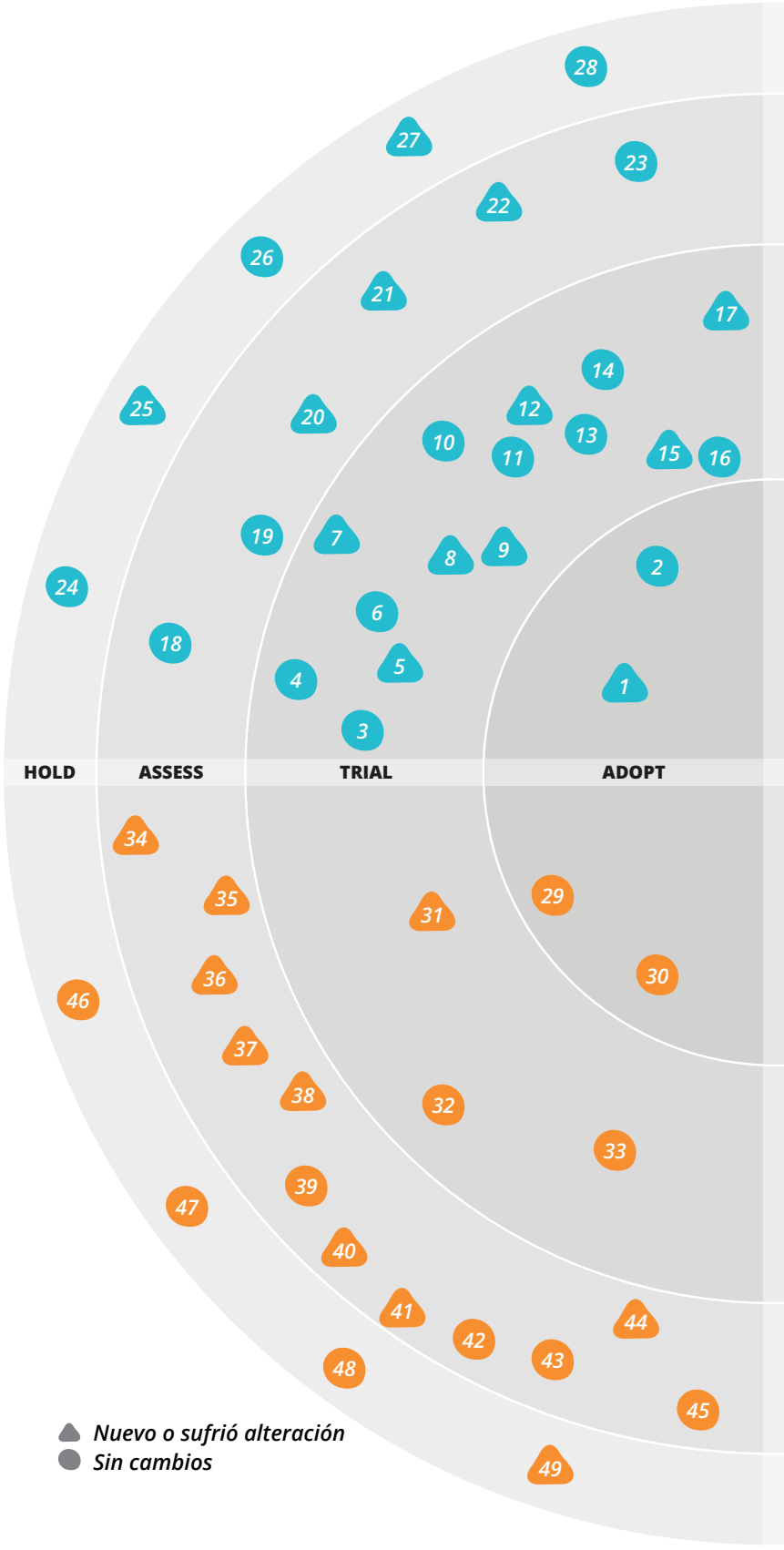
## PLATAFORMAS

***ADOPT* - ADOPCIÓN**
29. Hadoop 2.0
30. Vumi

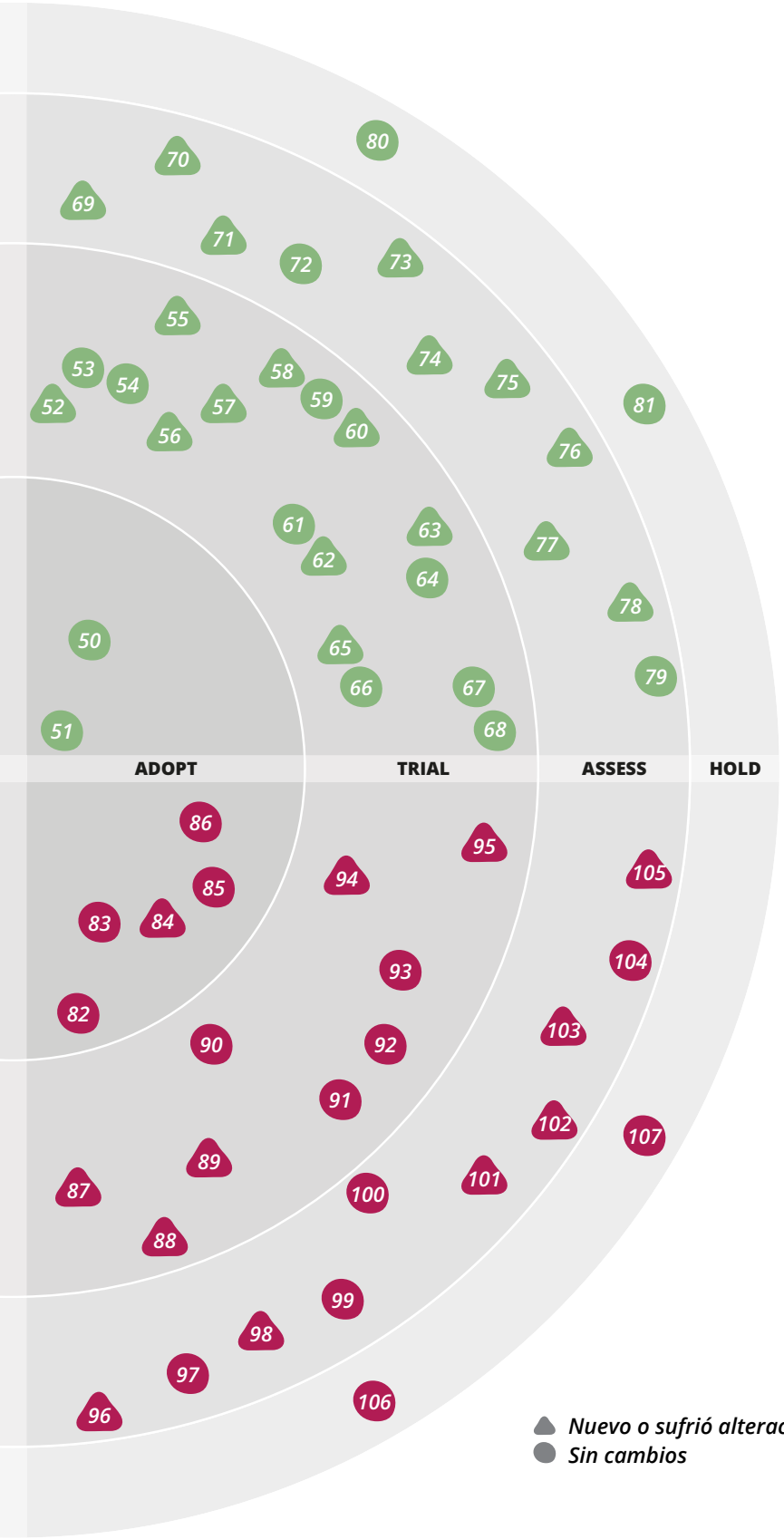
***TRIAL* - ENSAYO**
31. iBeacon
32. PostgreSQL para NoSQL
33. Nubes privadas

***ASSESS* - EVALUACIÓN**
34. SoC ARM para Servidor
35. CoAP
36. DigitalOcean
37. Espruino
38. EventStore
39. Robótica de bajo costo
40. Mapbox
41. OpenID Connect
42. SPDY
43. Storm
44. Autenticación de Doble Factor TOTP
45. Estándar de Componentes Web

***HOLD* - ESPERA**
46. Grandes soluciones empresariales
47. CMS como plataforma
48. Data Warehouse Empresarial
49. OSGi



# EL RADAR




## HERRAMIENTAS

***ADOPT* - ADOPCIÓN**
50. Ansible
51. Gestión de dependencias de JavaScript

***TRIAL* - ENSAYO**
52. CartoDB
53. Chaos Monkey
54. Docker
55. Flyway
56. Foreman
57. GenyMotion
58. Go CD
59. Grunt.js
60. Gulp
61. Moco
62. Packer
63. Pact & Pacto
64. Prototype On Paper
65. Protractor para AngularJS
66. SnapCI
67. Snowplow Analytics y Piwik
68. Herramientas de pruebas de regresión visual

***ASSESS* - EVALUACIÓN**
69. Appium
70. Consul
71. Flume
72. Soluciones Hosted para pruebas iOS
73. leaflet.js
74. Mountebank
75. Papertrail
76. Roslyn
77. Spark
78. Swagger
79. Xamarin

***HOLD* - ESPERA**
80. Ant
81. TFS

## LENGUAJES Y FRAMEWORKS

***ADOPT* - ADOPCIÓN**
82. Dropwizard
83. Lenguaje Go
84. Java 8
85. Reactive Extensions en todos los lenguajes
86. Scala, las partes buenas

***TRIAL* - ENSAYO**
87. AngularJS
88. Core Async
89. HAL
90. Hive
91. Nancy
92. Pester
93. Play Framework 2
94. Q y Bluebird
95. R como Plataforma de Computación

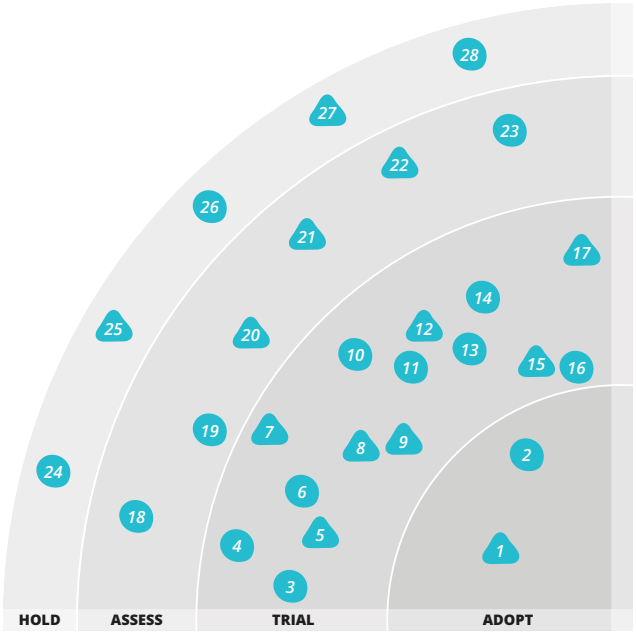
***ASSESS* - EVALUACIÓN**
96. Elm
97. Julia
98. Om
99. Pointer Events
100. Python 3
101. Rust
102. Spray/akka-http
103. Spring Boot
104. TypeScript
105. Lenguaje Wolfram

***HOLD* - ESPERA**
106. CSS escrito a mano
107. JSF

# TÉCNICAS

**Confidencialidad Directa Perfecta** (Perfect Forward Secrecy, PFS) es una técnica criptográfica que protege las sesiones de comunicaciones previas, incluso si después las llaves maestras del servidor se ven comprometidas. A pesar de que habilitar las conexiones HTTPS es muy simple, muchos servidores no se configuran de esta manera y se recomienda habilitar el PFS para aumentar la seguridad.

A medida que las aplicaciones JavaScript del lado del cliente crecen en sofisticación, aumenta la necesidad de sofisticación en la ingeniería de modo que los dos vayan a la par. 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 de la base de 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 la separación de responsabilidades es un antídoto muy útil, ya que limita en forma estricta todos los accesos al DOM (lo cual, por lo general, se traduce en el uso de jQuery) a una capa delgada de segregación. Un efecto secundario positivo que resulta de este enfoque es que todo lo que quede fuera de esta capa de **segregación del DOM** puede probarse de manera rápida y aislada desde el 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.

Los **entornos de desarrollo en la nube** le permiten externalizar por completo la infraestructura de desarrollo, haciendo que su equipo no necesite nada más que sus computadoras portátiles y una conexión a Internet. A través de una combinación de los mejores servicios, como la integración continua en la nube de los repositorios privados GitHub y Snap CI', es probable que sus equipos nunca más necesiten recurrir a la tecnología informática interna.

**ADOPT**
1. Confidencialidad Directa Perfecta
2. DOM segregada y uso de Node para pruebas de JS

**TRIAL**
3. Captura Explícita de Eventos de Dominio
4. Entornos de desarrollo en la nube
5. Origen de Eventos
6. Enfoque en el tiempo medio de recuperación
7. Registro humanizado de (micro)servicio
8. Maniobra inversa de Conway
9. Guías de estilo dinámicas CSS
10. Imagen de máquina como artefacto de construcción
11. Masterless Chef/Puppet
12. Empresa sin límites
13. Pruebas de aprovisionamiento
14. Monitorización de usuarios reales
15. REST sin PUT
16. Logging estructurado
17. Plantilla de Servicio Adaptada

**ASSESS**
18. Unión del mundo físico con el digital mediante hardware simple
19. Datensparsamkeit
20. Pipeline de imagenes de maquinas
21. Estrategia de aplicaciones mediante la agregación de niveles
22. Pruebas unitarias basadas en propiedades
23. Interacción tangible

**HOLD**
24. Auge y desplazamiento de la nube
25. DevOps como un equipo
26. Ignorar OWASP Top 10
27. Las pruebas como una organización independiente
28. Velocidad en terminos de productividad

# TÉCNICAS *continuación*

**Event sourcing** garantiza que todos los cambios que se realicen al estado de la aplicación se almacenen como una secuencia de eventos. No solo podemos buscar estos eventos, sino que también podemos usar el registro de eventos para reconstruir estados anteriores y además, podemos utilizarlo como base para ajustar de forma automática el estado para enfrentar cambios retroactivos. Además de la selección de eventos significativos para el negocio, la técnica tiene implicaciones positivas para el análisis de cómo conseguir mayor conocimiento sobre el cliente.

En las organizaciones DevOps-savvy, generalmente los equipos de distribución configuran el monitoreo de producción y ellos mismos responden a los incidentes. Gracias a esta visibilidad y al acceso a los entornos de producción, los equipos pueden realizar cambios en sus sistemas para mejorar la capacidad de recuperación rápida cuando algo sale mal. Este **enfoque en el tiempo medio de recuperación** mejora la calidad general de servicio y les permite a los equipos llevar a cabo una implementación segura con más frecuencia. Además, esto puede reducir el énfasis sobre la costosa ejecución de pruebas en entornos no productivos. Entre las técnicas que utilizamos se encuentran el "monitoreo semántico" integral o la conciliación de transacciones comerciales reales y la inyección de "transacciones sintéticas", que aplican sistemas en la producción de manera no destructiva.

Por su propia naturaleza, la arquitectura de microservicios incrementa significativamente la cantidad de aplicaciones, servicios e interacciones en los entornos de producción. Nuestros proyectos muestran un enfoque renovado en la creación de **Registros humanizados** (http://martinfowler.com/bliki/HumaneRegistry.html) que agregan información sobre los servicios corriendo en el ambiente de producción y presentan esta información de manera tal que los humanos puedan comprenderla. Estos registros mantienen información actualizada obtenida desde los sistemas en ejecución en lugar de la documentación elaborada por humanos.

La ley de Conway establece que las organizaciones se limitan a elaborar diseños de aplicaciones que son copias de sus estructuras de comunicación. Por lo general, esto conduce a puntos de fricción no deseados. La "**Maniobra Inversa de Conway**" recomienda desarrollar su estructura organizativa y su equipo para promover la arquitectura deseada. Lo ideal sería que su arquitectura tecnológica conserve isomorfismo con su arquitectura comercial.

Una **guía de estilo vivo de CSS** es una página dentro de un sitio que usa los estilos CSS actuales del sitio y actúa como referencia para todos los elementos visuales y patrones de diseño disponibles en el momento. Esto ayuda a integrar de forma estrecha el diseño con el proceso de entrega, al promover la propiedad compartida de la IU y evitar duplicación de los estilos en toda la aplicación. Los cambios de estilos son visibles en la guía de manera inmediata y se propagan a lo largo de su sitio desde una ubicación central. Una forma práctica de hacerlo es con una estructura de archivos SASS/LESS bien organizada con elementos denominados semánticamente y que separe estructura, estética e interacción.

Muchos de nuestros equipos obtienen un gran beneficio a partir de la publicación de **imágenes de máquinas virtuales como artefactos de construcción** durante sus procesos de creación automatizados. Estas imágenes de máquinas se publican con la aplicación y todas las dependencias, generalmente en un estado inmutable. Solo con una mínima configuración adicional, la imagen puede utilizarse para crear máquinas virtuales idénticas en todos los entornos y así eliminar las fuentes de errores y residuos más comunes. Las herramientas surgen para simplificar este enfoque, por ejemplo, Packer en la sección de herramientas del Radar. Este enfoque funciona con éxito en las empresas que adoptan un enfoque maduro con respecto a la nube y la virtualización y también donde los equipos de distribución tienen derecho de acceso y responsabilidad hasta llegar a la producción.

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

Las tendencias tecnológicas han derrumbado los muros que antes rodeaban las redes de tecnología informática empresarial y esto derivó en una **empresa sin límites**. Con frecuencia, los empleados usan sus propios dispositivos de cliente para acceder a los datos corporativos a través de servicios en la nube y API para web y, por lo general, sin el conocimiento de la organización'. Debido a que los dispositivos se siguen multiplicando y cada vez más aplicaciones se ubican en la nube, los negocios se ven obligados a pensar nuevamente acerca de suposiciones fundamentales sobre el acceso a los datos y la seguridad de las redes.

La virtualizacion de servidores y la computacion en la nube han hecho sencillo el conseguir y abastecer hardware y servidores virtuales. Sin embargo, esta flexibilidad viene con escalabilidad y complejidad, y, administrar nuestros bienes virtuales se ha vuelto cada vez más complejo. El uso de técnicas más familiares con el mundo del desarrollo de software, tales como TDD, BDD y CI, ofrece un enfoque para manejar esta complejidad y nos brinda la confianza necesaria a la hora de efectuar cambios en nuestra infraestructura en forma segura, repetible y automatizada. Las herramientas para **pruebas de aprovisionamiento, como rspec-puppet, Test Kitchen y severspec, estan disponibles para la mayoria de las plataformas**.

Con la proliferación de las aplicaciones JavaScript de una sola página, hemos encontrado que las llamadas Ajax lentas, la manipulación excesiva del DOM y los errores JavaScript inesperados en el Navegador pueden tener un gran impacto en la capacidad de respuesta de un sitio web. Es muy importante recopilar y añadir esta información de perfiles desde los navegadores de los usuarios finales reales. La **monitorización de los usuarios reales** proporciona una advertencia y diagnóstico anticipados de problemas de producción, además de ayudar a identificar dichos problemas en una situación en particular. http://newrelic.com/real-user-monitoring

En el último radar hablamos acerca de la Captura Explícita de los Eventos de Dominio, con una especial atención centrada en el registro de los eventos significativos del negocio que han desencadenado transiciones de estado y no solamente de entidades CRUD. A pesar de que es común que las interfaces REST usen PUT para actualizar el estado de los recursos, generalmente POST es mejor para registrar el recurso de un nuevo evento, lo que captura la intención. **REST sin PUT** tiene la ventaja secundaria de separar interfaces de comandos y consultas, forzando a los consumidores del servicio a una consistencia eventual.

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

Observamos múltiples organizaciones que crean una **Plantilla de Servicio Adaptada** que puede utilizarse para generar rápidamente nuevos servicios, pre-configurados para operar dentro del entorno de producción de una organización. La plantilla contiene un conjunto de decisiones predeterminadas, tales como: frameworks web, registros, monitoreo, construcción, empaquetamiento y enfoques de despliegue. Esta es una técnica muy útil que impulsa la evolución colaborativa, manteniendo una gobernabilidad ligera.

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 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 nunca almacena la información, ya no tendrá que preocuparse de que alguien quiera robársela.

Muchos despliegues requieren imágenes de máquinas para distintos roles de un servidor, aplicaciones y servicios, base de datos, proxy inversos, etc. Debido a que la creacioón de una imagen de una máquina desde cero, mediante el uso del ISO de un sistema operativo y scripts de provisionamiento, puede tardar mucho tiempo, puede ser muy útil crear un ***pipeline* de imágenes de máquinas**. La primera etapa en el pipeline establece una imagen base de acuerdo con los estándares generales de la organización. Las etapas posteriores pueden mejorar la imagen base para diversos fines. Si varias aplicaciones o servicios tienen requisitos similares (por ejemplo, un servidor de aplicaciones), el pipeline puede extenderse mediante una etapa intermedia, que toma la imagen base y proporciona una imagen con un servidor de aplicaciones pero sin aplicaciones ni servicios. Estos pipelines no son lineales; son como árboles que se ramifican desde la imagen de base.

La **Estrategia de Aplicaciones categorizadas por su ritmo de cambio** de Gartner intenta articular el hecho de que las decisiones sobre la arquitectura no deberían ser un planteamiento único para todos los enfoques. En su lugar, es importante adoptar una visión equilibrada sobre su portafolio de tecnología en términos de dónde ser más conservadores y dónde tomar riesgos. A pesar de que tenemos nuestros reparos acerca de algunas de las recomendaciones más prescriptivas que parecen estar ligadas a esta estrategia, nos agrada el concepto en general y muchas organizaciones podrían beneficiarse a partir de la adoptación de modelos similares.

Valoramos las pruebas unitarias en los proyectos y preferimos técnicas tales como las **pruebas unitarias basadas en propiedades** que las aumentan. Esta es una práctica que utiliza generadores de datos para crear entradas aleatorias dentro de rangos definidos. Permite un control rápido de las condiciones límites y otros modos de falla inesperados y cuenta con un creciente respaldo en múltiples plataformas.

A medida que las fronteras entre hardware y software continúan difuminándose, 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 vidrio. **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. **Desafortunadamente, el auge y desplazamiento de la nube** se ve fomentado por los grandes proveedores 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 piensa que puede mudarse a la nube sin diseñar una nueva arquitectura, es probable que no lo esté haciendo del modo adecuado.

Algunas empresas con buenas intenciones crean un **equipo DevOps** independiente, que malinterpreta la definición de DevOps. Más que un rol, DevOps es un movimiento cultural que promueve la colaboración entre los especialistas de operaciones y los desarrolladores. En vez de crear otro silo y sufrir las consecuencias de la ley de Conway, le recomendamos que integre estas capacidades en los equipos, mejorando así los círculos de crítica constructiva y las vías de comunicación, al eliminar la fricción.

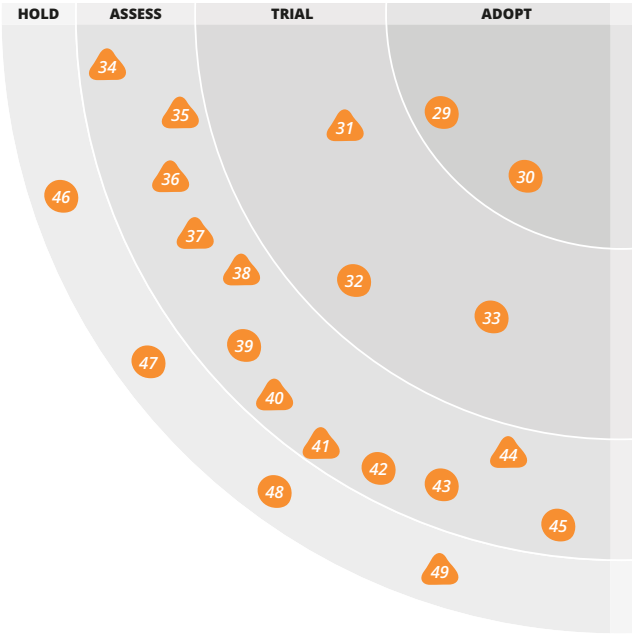
Casi nunca pasa más de una semana sin que el sector informático sufra los efectos negativos de una nueva pérdida de datos de alto perfil, fuga de contraseñas o violaciones de un sistema presuntamente seguro. Existen muy buenos recursos que ayudan a garantizar que la seguridad sea abordada como una preocupación de primera clase durante el desarrollo de software y y tenemos que dejar de ignorarlos. El **OWASP Top 10** es un buen punto de partida.

Seguimos observando cómo las organizaciones crean equipos independientes de control de calidad y desarrollo. Las críticas constructivas rápidas forman un precepto fundamental de Agile y son fundamentales para el éxito de un proyecto. Tener un equipo de control de calidad independiente demora estas críticas constructivas, crea una mentalidad de “nosotros y ellos” y dificulta el aumento de calidad del software. Las pruebas deben ser una actividad estrechamente integrada y no es algo que el equipo puede externalizar. Recomendamos equipos integrados en los que las personas encargadas de las pruebas trabajen estrechamente con los desarrolladores, en lugar de tener a **las pruebas como una organización independiente**.

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 los miembros del equipo empiezan a integrarse o al arreglar problemas como las deudas técnicas o un servidor que ha dejado de responder. Sin embargo, como cualquier otra métrica, la velocidad puede utilizarse de forma incorrecta. Por ejemplo, los gerentes de proyectos que son demasiado entusiastas tienden a insistir en la mejora continua de la velocidad. El hecho de considerar a la **velocidad en términos de productividad** genera conductas de equipo improductivas que optimizan la métrica a costa del funcionamiento real del software.

# PLATAFORMAS

La arquitectura inicial de Hadoop se basó en el paradigma del escalamiento de datos de forma horizontal y de metadatos de forma vertical. A pesar de que los nodos esclavos administraban el almacenamiento y procesamiento de datos bastante bien, los nodos maestros que gestionaban los megadatos representaban un punto único de falla y limitaban el uso web en escala. **Hadoop 2.0** ha avanzado en gran medida en el diseño de nuevas arquitecturas para HDFS y el framework Map Reduce para abordar estos problemas. El espacio de nombres de HDFS ahora puede unificarse mediante el uso de múltiples nodos de nombres en el mismo grupo e implementarse en un modo de alta disponibilidad. MapReduce se ha reemplazado por YARN, que desacopla la gestión de recursos de clúster de gestión estatal de trabajo y elimina los problemas de escala / rendimiento con la JobTracker. Más importante aún, este cambio alienta la implementación de nuevos paradigmas de programación distribuidos, además de MapReduce en clústeres de Hadoop.




En el último radar tecnológico, hablamos sobre **Vumi** como una plataforma para usar USSD como IU de teléfonos de funciones básicas. Vumi ha conseguido mucha estabilidad y su naturaleza de código abierto le aporta gran atractivo. En nuestros proyectos, hemos podido integrarnos a las redes de telecomunicaciones rápidamente y sin inconvenientes gracias a la simplicidad de configuración. Además, la plataforma es de fácil acceso y escalable.

**iBeacons** son la implementación de Apple de la categoría más amplia de beacons, que son dispositivos pequeños que utilizan Bluetooth Low Energy (BLE, por sus siglas en inglés) para proporcionar información de proximidad más detallada para teléfonos móviles y otros dispositivos. Más allá de todo el despliegue publicitario alrededor de iBeacons y las limitaciones con respecto a la precisión y confiabilidad de la información que ofrecen, ralmente sentimos que abre interesantes oportunidades como puntos desencadenantes para la interacción con sus usuarios de una manera contextualmente relevante.

**PostgreSQL** se amplía para convertirse en la opción NoSQL de las bases de datos SQL. La versión 9.2 incluye la posibilidad de almacenar datos JSON con total capacidad de búsqueda sobre el contenido del documento JSON. Otras extensiones le permiten al usuario almacenar y buscar información en el formulario de pares clave-valor. Esto le permite aprovechar las capacidades clave de almacenamiento y transacciones de una base de datos de eficacia comprobada, sin ataduras al modelo de datos relacionado. Esto es ideal para aquellos que desean aplicaciones SQL y también NoSQL, pero prefieren una sola infraestructura confiable que ya saben cómo asistir.

**ADOPT**
29. Hadoop 2.0
30. Vumi

**TRIAL**
31. iBeacon
32. PostgreSQL para NoSQL
33. Nubes privadas

**ASSESS**
34. SoC ARM para Servidor
35. CoAP
36. DigitalOcean
37. Espruino
38. EventStore
39. Robótica de bajo costo
40. Mapbox
41. OpenID Connect
42. SPDY
43. Storm
44. Autenticación de Doble Factor TOTP
45. Estándar de Componentes Web

**HOLD**
46. Grandes soluciones empresariales
47. CMS como plataforma
48. Data Warehouse Empresarial
49. OSGi

# PLATAFORMAS *continuación*

La cantidad y la madurez de las opciones de **nubes privadas** en las instalaciones continúan creciendo. Las soluciones que van desde las opciones basadas en OpenStack (como la nube privada de Rackspace) hasta las opciones PAAS (como CloudFoundry) son aquellas que deberían considerar las organizaciones que buscan utilizar la infraestructura existente o las que necesitan un mayor nivel de control sobre la nube fuera de las instalaciones.

Recientemente, AMD lanzó un **Soc (sistema en chip) ARM** de 8 núcleos diseñado para servidores y se comprometió a lanzar un SoC ARM con gráficos integrados en 2015. Los servidores basados en ARM representan una alternativa interesante de x86 por ser significativamente más eficientes en consumo de energía. Para determinados niveles de procesamiento, es preferible la creación de una nube impulsada por ARM. http://www.anandtech.com/show/7989/amd-announces-project-skybridge-pincompatible-arm-and-x86-socs-in-2015 http://www.anandtech.com/show/7724/it-begins-amd-announces-its-first-arm-based-server-soc-64bit8core-opteron-a1100

**CoAP** es un protocolo de comunicación de estándares abiertos para Internet de las cosas (Internet of things, IoT). A pesar de que actualmente existe una proliferación de estándares contrapuestos en el espacio IoT, en particular preferimos CoAP. Está específicamente diseñado para dispositivos de recursos limitados y redes de radio local. Utiliza UDP para el transporte, pero es semánticamente compatible con HTTP. CoAP utiliza un modelo basado en Internet, en el que los dispositivos poseen sus propias URLs, y un paradigma de solicitud/respuesta que soporta RESTful y enfoques descentralizados.

Aunque el espacio de la infraestructura como servicio (Infrastructure as a service, IaaS) esté bastante concurrido, hay espacio para que nuevos competidores ingresen al mercado. Recientemente, **DigitalOcean** (https://www.digitalocean.com) nos ha impresionado con su costo, velocidad y simplicidad. Si todo lo que necesitas es una infraestructura de computación básica, es interesante echarle un vistazo.

**Espruino** es un microcontrolador que ejecuta JavaScript de forma nativa y así permite que la curva de aprendizaje de una gran cantidad de programadores JavaScript sea baja. Usando un modelo basado en eventos similar a Node.js, los dispositivos Espruino pueden hacer un uso muy eficiente de la energía, sin perder capacidad de respuesta. Con menos potencia que un Raspberry Pi y levemente más lento que un Arduino, Espruino representa una alternativa interesante en entornos con limitaciones de energía que necesitan un rendimiento receptivo, pero que pueden sacrificar algunas de las funciones de alto nivel y la velocidad de ejecución de esas plataformas.

Dada la popularidad del origen de eventos, la madurez de las herramientas en este espacio no es ninguna sorpresa. **EventStore** es una base de datos funcional de código abierto para el almacenamiento de eventos inmutables y la realización de un procesamiento de eventos complejos en el flujo. A diferencia de otras herramientas de este espacio, EventStore expone los flujos de eventos como colecciones Atom y, por lo tanto, no requieren ninguna infraestructura especial para su uso, como buses de mensajes o clientes muy especializados. http://geteventstore.com/

With the cost of industrial robots dropping and their safety and ease of use increasing, the world of useful, commercial robotics is opening up. Robots like Rethink Robotics' Baxter* or Universal Robotics' U5, make it feasible for small to medium-sized businesses to automate repetitive tasks previously performed by humans. Increasingly, enterprise software will have to integrate with **low-cost robotics** as another participant in the value stream. The challenge lies in making the experience easy and productive for the human co-workers as well.

**Mapbox** (www.mapbox.com) es una plataforma abierta de mapeo que hemos utilizado en varios proyectos. Le permite al desarrollador agregar rápidamente un mapa a una aplicación y personalizarlo. Mapbox puede funcionar como una alternativa a las plataformas de mapeo convencionales y también es útil para mapas móviles flexibles.

**OpenID Connect** es un protocolo estándar para una identidad unificada construida sobre OAuth 2.0. Aborda la tradicional necesidad de un protocolo simple basado en Internet para intercambiar información de autenticación y autorización confiable. Estándares previos, como SAML u OAuth 2.0 genérico han demostrado ser demasiado amplios y complejos para asegurar la compatibilidad universal. Esperamos que OpenID Connect pueda proporcionar una base práctica para un acceso seguro a los microservicios RESTful con identidad de usuario final autenticada.

**SPDY** es un protocolo de red abierto para el transporte de baja latencia de contenido web propuesto para HTTP 2.0 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 humanos en HTTP. Es ideal para entornos de alta latencia.

Las cantidades heterogéneas y extremadamente grandes de datos no es el único aspecto a tener en cuenta con relación a big data. En determinadas 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.

**La autenticación de dos factores** mejora considerablemente la seguridad con respecto a los sistemas simples basados en contraseñas. El estándar RFC 6238 -- Time-based One-Time Password Algorithm -- describe el mecanismo de autenticación de dos factores. Las aplicaciones de autenticación "estándar" de Google y Microsoft proporcionan tokens para los usuarios de teléfonos inteligentes y hay muchas otras implementaciones de clientes y servidores de fácil acceso. En los casos de proveedores como Google, Facebook, Dropbox y Evernote que usan **TOTP**, realmente no hay ninguna excusa para continuar usando una autenticación simple basada en contraseña, ya que se necesita un nivel de seguridad mucho mayor. http://tools.ietf.org/html/rfc6238 http://en.wikipedia.org/wiki/Time-based_One-time_Password_Algorithm https://play.google.com/store/apps/details?id=com.google.android.apps.authenticator2 http://www.windowsphone.com/en-us/store/app/authenticator/e7994dbc-2336-4950-91ba-ca22d653759b

En el último radar, alertamos sobre el uso de frameworks tradicionales de componentes web que proporcionan un modelo de componentes del lado del servidor. El **estándar de Componentes Web** que se originó en Google es bastante diferente. Proporciona una forma sencilla de crear widgets reciclables al contribuir con el aislamiento del HTML, CSS y JavaScript, para que no interfieran con el resto de la página y, al mismo tiempo, la página no interfiera con ellos. Los desarrolladores pueden utilizar cuanto necesiten del framework, sea mucho o poco. El proyecto Polymer ofrece soporte temprano para este estándar.

La brecha que separa lo que ofrecen los paquetes comerciales de "clase empresarial" y lo que realmente se necesita se amplía cada vez más. Esto es particularmente cierto para las aplicaciones de Internet. Las soluciones innovadoras que son realmente escalables y que admiten fácilmente técnicas modernas como la entrega continua están escritas por profesionales, y son para profesionales. Se originan con muchas empresas de escala de Internet y se perfeccionan como un software de código abierto. Por lo general, las **grandes soluciones empresariales** impiden la distribución efectiva debido a sus acumulaciones, sus restricciones de licencias complejas y los conjuntos de funciones que son accionados por listas de comprobación y requisitos imaginarios, totalmente fuera de la realidad de la mayoría de los equipos de desarrollo.

Los sistemas de gestión de contenidos (content management systems, CMS) tienen su lugar. En muchos casos, no tiene sentido escribir las funcionalidades de redacción y flujo de trabajo desde 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.

Al tiempo que la integración centralizada de datos para la elaboración de análisis e informes continúa siendo una buena estrategia, las iniciativas tradicionales del **Data Warehouse Empresarial** (EDW, por sus siglas en inglés) presentan una tasa de fallos superior al 50 %. Las grandes modificaciones de datos por adelantado traen, como consecuencia, almacenamientos 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.

**OSGi** (del inglés Open Service Gateway initiative) es una especificación que busca corregir la falta de un sistema modular para Java, al permitir la recarga dinámica de componentes. A pesar de que algunos proyectos (en especial Eclipse) utilizan OSGi correctamente, otros han expuesto los riesgos de agregar abstracciones a plataformas que nunca se diseñaron para esos usos. Los proyectos que confían en OSGi para definir un sistema de componentes, rápidamente advierten que esto solo soluciona una parte de todo el problema y, por lo general, agrega su propia complejidad accidental a los proyectos, así como a creaciones más complejas. En la actualidad, la mayoría de los proyectos o bien usan archivos JAR pasados de moda o arquitecturas de microservicios para gestionar los componentes, al tiempo que esperan la solución nativa de Java en la especificación modular Jigsaw.

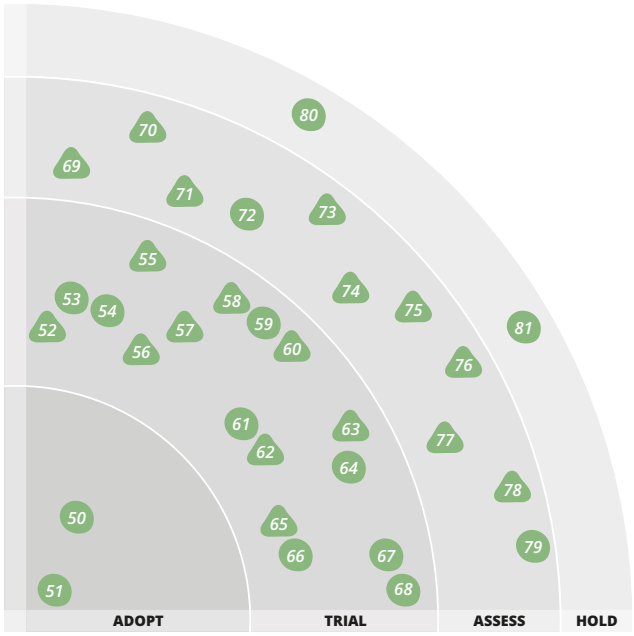
# HERRAMIENTAS

Desde la primera presentación de **Ansible** en el último radar, no dejamos de asombrarnos acerca de sus capacidades y facilidad de uso, en comparación con otras ofertas en este espacio. Si nos basamos en nuestras experiencias durante el último año, no dudamos en recomendar Ansible como una gran opción para el control automatizado de su infraestructura.

El uso de herramientas de **gestión de dependencias de JavaScript** ha ayudado a nuestros equipos de entrega a manejar grandes cantidades de JavaScript estructurando su código y cargando las dependencias en 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.

**CartoDB** es una herramienta GIS (sistema de información geográfica, por sus siglas en inglés) de código abierto, basada en PostGIS y PostgreSQL. Permite el almacenamiento y la búsqueda de datos geoespaciales a través de SQL. Además, ofrece una práctica biblioteca JavaScript, CartoDB.js, para representar los mapas y visualizar los datos.

Según nuestra recomendación del último radar acerca de enfocarse 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 determinados casos 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 en lugar de luchar por responder ante un corte en el suministro que toma a todos por sorpresa.




El ímpetu de **Docker** continúa creciendo y ya puede observarse su uso en algunos proyectos, aunque mayormente en entornos no productivos. Docker proporciona un conjunto de herramientas para presentar y distribuir de manera eficiente las imágenes de máquinas ejecutables, que luego pueden lanzarse como contenedores livianos. Una importante comunidad crece alrededor de esta herramienta. Debemos destacar a CoreOS, que es un sistema operativo basado en ChromeOS y creado para implementar los contenedores Docker en un grupo con herramientas de implementación, descubrimiento y configuración del servicio. Ref: https://coreos.com

**ADOPT**
50. Ansible
51. Gestión de dependencias de JavaScript

**TRIAL**
52. CartoDB
53. Chaos Monkey
54. Docker
55. Flyway
56. Foreman
57. GenyMotion
58. Go CD
59. Grunt.js
60. Gulp
61. Moco
62. Packer
63. Pact & Pacto
64. Prototype On Paper
65. Protractor para AngularJS
66. SnapCI
67. Snowplow Analytics y Piwik
68. Herramientas de pruebas de regresión visual

**ASSESS**
69. Appium
70. Consul
71. Flume
72. Soluciones Hosted para pruebas iOS
73. leaflet.js
74. Mountebank
75. Papertrail
76. Roslyn
77. Spark
78. Swagger
79. Xamarin

**HOLD**
80. Ant
81. TFS

# HERRAMIENTAS *continuación*

Las migraciones de bases de datos automatizadas son comunes en los proyectos ágiles y estamos muy contentos de ver avances en las herramientas para este espacio. **Flyway** hace que la automatización de los cambios de las bases de datos sea lo menos difícil posible. A pesar que no tiene tantas características como otras herramientas, la hemos utilizado en muchos proyectos y realmente apreciamos de su sencillez.

Indudablemente, los grandes proveedores de servicio en la nube han elevado los estándares para el suministro, monitoreo y configuración de sus servicios, al simplificar dramáticamente estas tareas mediante el uso de herramientas poderosas. Las organizaciones que quieren mantener sus recursos de computación y almacenamiento en sus instalaciones buscan soluciones similares que funcionen dentro de su contexto organizacional. En nuestra experiencia, **Foreman** ha funcionado muy bien y además es un software de código abierto. http://theforeman.org

La fragmentación de dispositivos en el ecosistema de Android se percibe generalmente como un problema ya que puede ser difícil entender como van a comportarse las aplicaciones en una gran cantidad de plataformas disímiles. **GenyMotion** (http://www.genymotion.com) simula las características de muchos dispositivos Android diferentes. Nuestros equipos de trabajo descubrieron que es una herramienta muy efectiva ya que permite obtener rápidamente retroalimentación sobre el funcionamiento de nuestras aplicaciones Android.

Debido al creciente interés en la entrega continua (Continuous Delivery, CD) y en los pipelines para el despliegue de software, hemos visto a muchos equipos intentar extender sus herramientas de integración continua con complementos que proveen pipelines a un nivel visual. **Go CD** fue desarrollado considerando este concepto a todo nivel. Go CD es capaz de ordenar los flujos de trabajo en secuencias o en paralelo a múltiples niveles para ejecutar tareas específicas solo en máquinas determinadas y también para promover y propagar artefactos de manera determinista, lo que habilita la entrega continua. Todas estas son capacidades que la mayoría de las herramientas de integración continua no tienen y por eso recomendamos Go CD a aquellos equipos que hayan intentado de alguna otra forma crear un pipeline para despliegue de su servidor de integración continua. ThoughtWorks desarrolló Go CD, que es de código abierto y está disponible en forma gratuita para todos los equipos. http://www.go.cd. El código fuente está disponible bajo la licencia Apache 2.0: https://github.com/gocd/gocd

Hemos observado un crecimiento en el ecosistema de **grunt** y actualmente se utiliza en varios de nuestros proyectos. Gracias a la proliferación de plugins y a la facilidad para escribir y publicar plugins propios a npm, la automatización a través de grunt puede llevarse a cabo casi sin esfuerzos. Aconsejamos elegir el ejecutador de tareas que mejor se adapte a las necesidad del proyecto y sin dudas, grunt es uno de los ejecutadores que debería considerar.

**Gulp** es una alternativa a Grunt. Es una herramienta de automatización de tareas para la líneas de comandos, que ayuda a los desarrolladores como SaaS (Software como Servicio por sus siglas en inglés), en actividades como la compilación, los prefijos automáticos, la minificación, la concatenación y mucho más. La idea central de Gulp es el uso de flujos y sus plugins están diseñados para realizar solo una tarea.

Probar microservicios sobre HTTP puede ser dificil y problematico. Particularmente en dos escenarios, al consumir un grupo de microservicios desde el front-end y la comunicacion entre microservicios. Para solventar estos escenarios **Moco** puede ser util. Es un framework ligero de stubs para probar end points HTTP. Puedes tener un servicio de stubs ejecutandos con dos lineas de codigo Java o Groovy, o un servicio standalone con unas pocas lineas de JSON que describan el comportamiento esperado.

En el último Radar, presentamos la "imagen de una máquina como un artefacto" como un modo excepcional para implementar servidores inmutables de una forma rápida. La dificultad en la creación de imágenes fue lo que causó demora en esta técnica, en particular al enfocarse en más de una plataforma. **Packer** (http://www.packer.io) tiene la solución para esto, a través del uso de su herramienta de configuración preferida para crear imágenes para un gran número de plataformas, incluyendo AWS, Rackspace, DigitalOcean e incluso Docker y Vagrant, aunque descubrimos que el soporte para VMWare puede ser bastante problemático.

Los contratos orientados al consumidor representan un enfoque de prueba para ayudar a las interfaces de servicio a evolucionar con confianza, sin afectar a los consumidores inconscientemente. Los recursos con nombres similares **Pact** (https://github.com/realestate-com-au/pact) y **Pacto** (http://thoughtworks.github.io/pacto) son dos herramientas de código abierto nuevas que permiten la interacción de pruebas entre los proveedores de servicios y los consumidores aislados frente a un contrato. Ambas surgieron de proyectos que desarrollaban microservicios RESTful y son muy prometedoras.

# HERRAMIENTAS *continuación*

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

**Protractor** es un marco de prueba basado en Jasmine que envuelve a WebDriverJS con una funcionalidad diseñada especialmente para ejecutar pruebas de fin a fin en aplicaciones **Angular.JS**. Descubrimos que es excelente en cuanto a la rápida evolución del espacio de frameworks de prueba JavaScript. A pesar de que se diseñó para ejecutar pruebas de extremo a extremo con un backend real, las pruebas de Protractor también pueden realizarse para trabajar con una puerta de entrada HTTP con servicio de stub para ejecutar solo pruebas del lado del cliente.

En la última edición del Radar, mencionamos **SnapCI** de ThoughtWorks (un servicio alojado que brinda herramientas de implementación). Desde ese momento, hemos observado que muchos equipos utilizan SnapCI con éxito en sus proyectos. Si necesita una solución de entrega 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.

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 en un campo que aún se encuentra en desarrollo activo, creemos que las pruebas de regresión visual deben añadirse a las formas de entrega continua.

Cada vez es más importante la automatización de pruebas para dispositivos móviles. **Appium** es un framework de automatización de pruebas que puede utilizarse para probar la web móvil y aplicaciones móviles nativas e híbridas en iOS y Android. En su parte central, Appium es un servidor web que expone un REST API, recibe conexiones de un cliente y escucha los comandos, ejecuta esos comandos en un dispositivo móvil y contesta con HTTP que representa el resultado de la ejecución del comando. Permite que las pruebas puedan escribirse en múltiples plataformas (iOS, Android) mediante el uso de la misma API. Appium es de código abierto con una fácil configuración mediante el uso de npm. (www.appium.io)

**Consul** hace los servicios se registren a sí mismos fácilmente y descubran otros servicios vía DNS o HTTP. Escala de forma automática, con una búsqueda de servicios de manera local o en todos los centros de datos. Además, Consul proporciona un almacenamiento flexible de clave-valor para lograr una configuración dinámica, con notificaciones de cambios de configuración. El protocolo gossip interno que utiliza Consul está impulsado por la biblioteca Serf, que aprovecha y construye sobre las características de detección de fallos y membresía. (http://www.consul.io, http://www.serfdom.io)

Cuando se usan técnicas como “instrumentar todas las cosas” y el registro semántico, es posible que termine con una gran cantidad de datos de registros. La compilación, consolidación y movimiento de estos datos puede resultar un verdadero problema y **Flume** es un sistema distribuido que justamente aborda esos asuntos. Posee una arquitectura flexible basada en la reproducción de flujos de datos. Por el soporte integrado para HDFS, Flume puede mover fácilmente enormes cantidades (varios terabytes) de datos de registros desde múltiples orígenes hacia un almacenamiento centralizado de datos para su procesamiento posterior.

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

**Leaflet.js** (www.leafletjs.com) es una biblioteca JavaScript para mapas interactivos flexibles para dispositivos móviles. La biblioteca se enfoca principalmente en el rendimiento, la usabilidad y la simplicidad y, por ello, funciona de manera muy eficaz en todas las plataformas móviles y navegadores de escritorio. Es una biblioteca que se debe tomar cuenta si se desea crear mapas interactivos para dispositivos móviles.

Cuando probamos servicios, comúnmente necesitamos disociar los servicios de flujos colaborativos. **Mountebank** (http://www.mbtest.org), escrito por un ThoughtWorker, es un servicio ligero que puede configurarse vía HTTP y es capaz de disociar y simular HTTP, HTTPS, SMTP y TCP.

**Papertrail** es un servicio de incorporación de log que agrega datos de una gran variedad de fuentes, entre ellas servidores web, routers, bases de datos y servicios de plataforma como servicio (PaaS, por sus siglas en inglés). Además de la incorporación, también proporciona búsquedas, filtros, alertas y notificaciones listas para usar. A pesar de que es imposible negar que es práctico y conveniente en la mayoría de los casos, aún nos preocupa la adopción generalizada de servicios que centralizan grandes cantidades de datos agregados por múltiples partes.

**Roslyn**, una plataforma de compilación para .NET bajo la licencia Apache 2.0, es un conjunto de compiladores de última generación para C# y VB.NET, completamente escritos con código administrado. Brinda acceso al compilador como un servicio y proporciona APIs de análisis de código que permiten a los desarrolladores acceder a información provista por el compilador, como por ejemplo, los modelos sintácticos y semánticos. El impacto más inmediato debería presentarse como mejoras para los IDEs .NET a través de herramientas de refactorización y generación de código. También esperamos ver mejoras en el diagnóstico y análisis estático del código, aunque sería muy interesante ver las propuestas de la comunidad. Mientras tanto, Xamarin tiene una copia compatible con Mono del código fuente de Roslyn alojada en GitHub y planea integrar los compiladores de Roslyn con Mono a medida que se estabiliza, además de integrar las mejores partes en su base de código.

En tareas de procesamiento iterativo, como el aprendizaje automático y el análisis interactivo, las funciones map-reduce de Hadoop no funcionan correctamente debido a su naturaleza basada en lotes. **Spark** es un motor rápido y general para el procesamiento de datos a gran escala. Busca ampliar map-reduce para soportar algoritmos iterativos y la minería de datos interactiva de baja latencia. **Spark** también incluye una biblioteca de aprendizaje automático.

**Swagger** es una manera estándar para describir un RESTful API de modo que los ejemplos de documentación y clientes puedan generarse de forma automática. Creemos que existe la necesidad de contar con algunos estándares en esta área y esperamos que este enfoque incluya la ley de Postel y evite el acoplamiento demasiado estrecho y la inflexibilidad de los estándares como WSDL. Ahora hay disponible una gran cantidad de herramientas para generar páginas del cliente y documentación desde las descripciones que cumplen con swagger. https://helloreverb.com/developers/swagger http://en.wikipedia.org/wiki/Robustness_principle http://en.wikipedia.org/wiki/Robustness_principle https://github.com/wordnik/swagger-ui

Entre las varias opciones disponibles para crear aplicaciones móviles para todas las plataformas, **Xamarin** ofrece un conjunto único de herramientas. Permite el uso de C# y F# como el lenguaje de programación principal con vínculos hacia los SDKs específicos de las plataformas y al entorno de tiempo de ejecución de Mono, que funciona en iOS, Android y Windows Phone. Las aplicaciones se compilan a código nativo en lugar del enfoque tradicional que presenta interfaces de usuario HTML mediante un control de navegador embebido. Esto permite que las aplicaciones tengan una apariencia y funcionamiento más nativo. Cuando se utiliza estas herramientas, es indispensable que la capa de interfaz de usuario específica para una plataforma se encuentre separada del resto de capas, para así garantizar la reutilización del código en las diferentes plataformas. La aplicación binaria tiende a ser un poco más grande debido a que se incluye el entorno de tiempo de ejecución.

Seguimos observando cómo los equipos invierten esfuerzos significativos en scripts de construcción insostenibles de **Ant** y Nant. Éstos son difíciles de comprender y ampliar debido a la inherente falta de expresividad y modularidad que proporcionan estas herramientas. Alternativas como Gradle, Buildr y PSake han demostrado, sin dudas, un mantenimiento y una productividad superiores.

Seguimos observando que los equipos enfrentan problemas de productividad al intentar usar **TFS** como un sistema de control de versiones. Los equipos que buscan hacer checkins frecuentemente (una parte central de la integración continua) han descubierto que su enfoque pesado reduce considerablemente la productividad. Por lo general, esto provoca que los equipos chequeen su código con menos frecuencia, lo que causa fusiones más problemáticas. En su lugar, recomendamos herramientas, tales como Git, Perforce y Subversion.

# LENGUAJES Y FRAMEWORKS

**Dropwizard** es una combinación sólida y práctica de varias herramientas y marcos de trabajo ligeros Java, muchos de los cuales tienen méritos por su cuenta. El paquete incorpora muchas de nuestras técnicas favoritas, entre ellas un servidor HTTP integrado, soporte para puntos de acceso RESTful, métricas operativas y controles de salud integrados y además, la capacidad de poner en marcha la aplicación con facilidad. Hacer las cosas bien es muy simple con Dropwizard, ya que permite enfocarse en resolver los problemas "de negocio" en lugar de preocuparse mucho por la infraestructura.

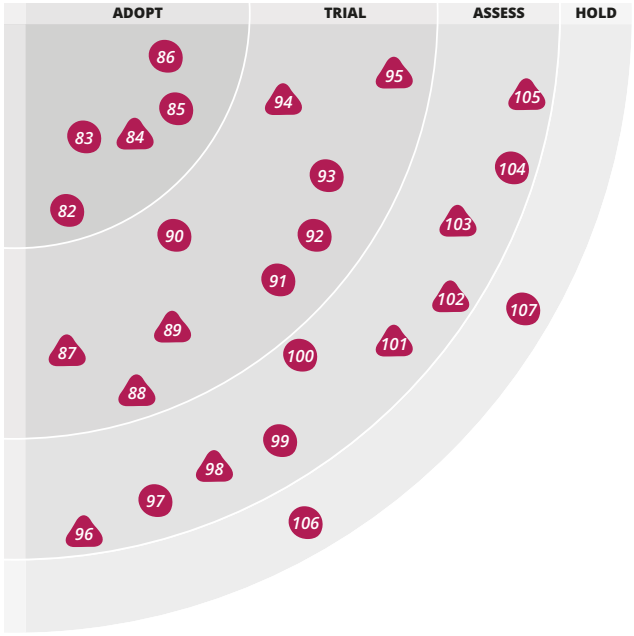
El **lenguaje Go** ha pasado gradualmente de ser "Simplemente un Lenguaje Más" a convertirse en una valiosa herramienta para muchos proyectos. A pesar de ser resueltamente de paradigma simple en un mundo de lenguajes cada vez más complejos, parece mantener un buen equilibrio entre expresividad, potencia y simplicidad.

El equipo que se encuentra detrás de **Java 8** tuvo que luchar dos batallas: las fuerzas de la comunidad siempre a favor de la retrocompatibilidad (un sello tradicional de Java) y el desafío técnico de llevar a cabo un cambio profundo en las bibliotecas y funciones existentes. Triunfaron en ambos frentes, ya que le dieron una nueva vida al lenguaje Java y lo ubicaron a la par de otros lenguajes tradicionales en cuanto a características funcionales de programación. En particular, Java 8 cuenta con una magia sintáctica excepcional que permite una interoperabilidad perfecta entre los bloques Lambda, la nueva característica conocida como funciones de orden superior, y las interfaces SAM (del inglés Single Abstract Method), la forma tradicional de transmitir comportamiento.

La **arquitectura reactiva** se sigue extendiendo a lo largo de las plataformas y los paradigmas, simplemente porque soluciona un problema común de una forma elegante y así logra ocultar las inevitables conexiones de la aplicación en un agradable aislamiento.

Scala es un lenguaje de gran tamaño que es popular gracias a su accesibilidad para los desarrolladores nuevos. Este abanico de características 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 que no desee mediante el uso de un sistema denominado "banderas de funcionalidades".

Seguimos viendo a los frameworks JavaScript como una buena manera para estructurar código y aportar mejores técnicas de programación a JavaScript. **AngularJS** se utiliza en gran medida en los proyectos de ThoughtWorks. Sin embargo, siempre les aconsejamos a los equipos que evalúen otras buenas alternativas, como Ember.js y Knockout.js.




**ADOPT**
82. Dropwizard
83. Lenguaje Go
84. Java 8
85. Reactive Extensions en todos los lenguajes
86. Scala, las partes buenas

**TRIAL**
87. AngularJS
88. Core Async
89. HAL
90. Hive
91. Nancy
92. Pester
93. Play Framework 2
94. Q y Bluebird
95. R como Plataforma de Computación

**ASSESS**
96. Elm
97. Julia
98. Om
99. Pointer Events
100. Python 3
101. Rust
102. Spray/akka-http
103. Spring Boot
104. TypeScript
105. Lenguaje Wolfram

**HOLD**
106. CSS escrito a mano
107. JSF

# LENGUAJES Y FRAMEWORKS *continuación*

La librería de Clojure **core.async** permite la comunicación asíncrona mediante el uso de canales, con una sintaxis y capacidades similares a las del lenguaje Go de Google. La librería core.async soluciona muchos problemas comunes de una forma elegante, limpiando el uso de llamadas de devolución y agregando primitivas simples para la concurrencia. Además, destaca una de las ventajas de la naturaleza Lisp de Clojure: los canales incorporan operadores que son consistentes con los operadores de Clojure ya existentes y, de esta manera, la nueva funcionalidad se va enlazando sin problemas con el núcleo del lenguaje. Además, core.async es compatible con Clojure y ClojureScript (a pesar de la falta de hilos en JavaScript) utilizando las abstracciones propias de la plataforma para ofrecer una interfaz consistente en ambos lenguajes.

Podemos ver cómo muchos equipos crean interfaces RESTful sin prestar atención a los hipermedios. **HAL** es un formato simple para incorporar hipervínculos en las representaciones JSON, que es algo fácil de implementar y consumir. Las bibliotecas admiten HAL sin problemas para el análisis y representación de JSON y hay bibliotecas de clientes HAL-aware REST, como Hyperclient, que facilitan la navegación de recursos siguiendo enlaces. http://stateless.co/hal_specification.html https://github.com/codegram/hyperclient

**Hive** es un data warehouse 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 través del cluster Hadoop. Al igual que todas las abstracciones útiles, Hive no intenta negar la existencia de los mecanismos fundamentales de Hadoop y apoya 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. Desaconsejamos firmemente el uso de Hive con fines de consulta en línea ad hoc.

**Nancy** continua ganando terreno en la comunidad Alt.NET y hemos encontrado que es muy práctica para poner en funcionamiento fakes (imitaciones) ligeros y simples para pruebas en un entorno de microservicios.

PowerShell continúa siendo una opción muy utilizada para lograr una automatización de nivel bajo en máquinas Windows. **Pester** constituye una biblioteca 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.

La situación de **Play Framework 2** ha generado distintos debates internos. Existen sugerencias acerca de si resulta conveniente desplazar este framework a adopción 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 del mismo. A pesar de que ninguna de estas cuestiones es única para Play, esto ha generado más polémica de la habitual en el debate entre la librería estándar y el 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.

**Q** (https://github.com/kriskowal/q) es una implementación completa del estándar Promises/A+ escrita en JavaScript, que permite a los usuarios crear promesas de manera arbitraria sin necesidad de usar devoluciones de llamadas anidadas profundamente que dificultan el seguimiento del flujo del control. Q se encarga de dirigir los valores cumplidos y las promesas rechazadas a través de las rutas de código adecuadas. Actualmente, el espacio de las bibliotecas de Promises/A+ es muy activo, con alternativas como **Bluebird** (https://github.com/petkaantonov/bluebird), que también está ganando reconocimiento rápidamente.

Tradicionalmente, los equipos de investigación usaban R como una herramienta de análisis independiente. Con mejoras en paquetes como Rook y RJSONIO, desarrollar la lógica computacional y exponerla como una API se ha convertido en algo trivial. Los equipos de ThoughtWorks utilizan **R como plataforma de computación** para analizar grandes conjuntos de datos en tiempo real, usando almacenamiento en memoria integrado con sistemas empresariales.

**Elm** es un lenguaje de programación funcional que se utiliza para crear interfaces de usuario basadas en la web en un estilo reactivo-funcional. Elm es de tipado fuerte y estático, y está construido sobre la plataforma de Haskell. Elm tiene una sintaxis tipo Haskell, pero compila a HTML, CSS y JavaScript. A pesar de que aún es muy reciente, los individuos y los equipos que estén interesados en explorar las interfaces gráficas de usuario de gran interacción basadas en la web deberían echarle un vistazo a este pequeño pero interesante lenguaje.

**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 del lenguaje 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.

La adopción de todo el conjunto Clojure (los lenguajes Clojure y ClojureScript y, como una opción adicional, la base de datos Datomic) proporciona algunas ventajas, como las estructuras de datos inmutables que van desde la interfaz del usuario hasta el backend. Han aparecido muchos frameworks en el espacio Clojure para aprovechar estas características únicas, pero hasta el momento la más prometedora es **Om**. Om es una cubierta de ClojureScript en torno al marco de programación reactiva React JavaScript de Facebook. No obstante, Om aprovecha la inherente inmutabilidad de ClojureScript, al aprovechar las funciones automáticas como imágenes del estado de la IU y la anulación. Por lo tanto, debido a la eficiencia de las estructuras de datos de ClojureScript, algunas de las aplicaciones Om se ejecutan con mayor velocidad que otras aplicaciones similares que se basan en los frameworks fundamentales React sin procesar. Esperamos gran evolución e innovación para que sigan acompañando a Om.

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.

**Python 3** representó un cambio importante desde su predecesor Python 2.x que introdujo modificaciones incompatibles con versiones anteriores. Se destacó porque eliminó las características de los lenguajes, haciendo que el uso de Phyton 3 resulte más fácil y consistente, sin disminuir su poder. Esto ha generado problemas en la adopción, ya que algunas de las bibliotecas 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, vuelva a echarle un vistazo a Python 3.

**Rust** es un lenguaje de programación con varias funcionalidades modernas. Cuenta con un interesante sistema de tipeo, manejo seguro de memoria y currencia basada en tareas. En comparación con el lenguaje Go, Rust es una mejor opción para aquellas personas a las que les gustaría escribir código con un estilo funcional.

**Spray/akka-http** es un conjunto de bibliotecas Scala ligeras que ofrecen compatibilidad RESTful cliente/servidor sobre Akka. Soporta plenamente los modelos de programación basados en Actor, Futuro y Flujo utilizados por la plataforma subyacente. Esto permite trabajar sobre las aplicaciones RESTful con códigos idiomáticos Scala, pero sin preocuparse por utilizar otras bibliotecas Java.

**Spring boot** permite una fácil configuración de aplicaciones autónomas basadas en Spring. Es ideal para impulsar nuevos microservicios y es de fácil puesta en producción. Además, contribuye a que el acceso a los datos sea menos doloroso, gracias al uso de hibernate reduciendo la duplicación de códigos. (http://projects.spring.io/spring-boot/)

**TypeScript** representa un enfoque interesante si hablamos de introducir un nuevo lenguaje de programación para los navegadores. 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 futuras extensiones planificadas de JavaScript.

Estamos realmente muy intrigados sobre las posibilidades que ofrece el **lenguaje Wolfram**. Creado sobre los enfoques simbólicos del lenguaje Mathematica, también tiene acceso a una gran cantidad de algoritmos y datos del proyecto Wolfram Alpha, lo que significa que programas muy sencillos pueden analizar y visualizar poderosas combinaciones de los datos del mundo real.

Junto con JavaScript y HTML, CSS representa una tecnología básica para la creación de sitios web. Lamentablemente, el lenguaje en sí mismo carece de funcionalidades claves, lo que conduce a un gran nivel de duplicaciones y a una falta de abstracciones útiles. A pesar de que CSS3 busca corregir algunos de estos problemas, pasarán años antes de que estos módulos que lo constituyen sean soportados correctamente en la mayoría de los navegadores. Por suerte, en la actualidad existe una solución mediante el uso de **preprocesadores CSS** como SASS y LESS. Debido a su calidad y soporte, creemos que los días del **CSS escrito a mano** (para cualquier otra cosa que no sean trabajos triviales) han llegado a su fin.

Seguimos observando cómo los equipos enfrentan diversos inconvenientes al utilizar **JSF** (JavaServer Faces) y por lo tanto recomendamos evitar esta tecnología. Parece ser que los equipos eligen JSF porque es un estándar JEE sin evaluar realmente si el modelo de programación les resulta conveniente o no, según sus necesidades. Creemos que JSF es deficiente porque intenta abstraer HTML, CSS y HTTP, lo cual representa exactamente lo opuesto a lo que hacen el resto de los frameworks web modernos. JSF, al igual que los formularios web ASP.NET, intenta crear estado sobre el protocolo HTTP el cual por naturaleza no tiene esta característica 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.

---

**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.

ThoughtWorks®