Aplicaciones con arquitecturas de **próxima generación**

Foto: Jay Mantry, Creative Commons

Las ciudades inteligentes, el internet de las cosas, millones de usuarios de dispositivos, son realidades que harán indispensable el diseño cuidadoso de las arquitecturas de las aplicaciones.

Las facilidades que ofrece la nube para diferentes tipos de negocios imponen nuevas consideraciones a la hora de programar, otras formas de enfrentar las fallas y de evitar los límites de la escalabilidad.

Es imperativo pensar en la forma en que se está construyendo la arquitectura de las aplicaciones para la nube, pues tendrá que ofrecer las condiciones adecuadas para que petabytes de datos fluyan sin tropiezos en la red. Más de 300 billones de dispositivos conectados y un volumen de usuarios que escala de manera exponencial inundarán pronto el ciberespacio cuando, por ejemplo, se consoliden fenómenos y tecnologías como el internet de las cosas. Esta realidad es responsable de que los requerimientos del software varíen constantemente, ante lo cual la manera de programar debe adecuarse.

Así lo afirmó Ryan Knigth, invitado especial del 8.º Foro de *Cloud Computing* "Nuevos enfoques de arquitectura para software en la nube", que tuvo lugar el 30 de octubre del 2014. Knight es consultor senior en Typesafe y creador de marcos de referencia como la arquitectura reactiva, que busca diseñar aplicaciones capaces de responder en tiempo real a los eventos que ocurren en la red.

Las características principales de la arquitectura reactiva o de próxima generación se encuentran en www.reactivemanifesto.org. Estas son: sensibilidad, elasticidad, resiliencia y comunicación por mensajes. A continuación su significado:

Sensibilidad: Reacciona ante las necesidades del usuario y le proporciona una experiencia agradable en tiempo real. Con arquitecturas reactivas una empresa puede atender las solicitudes de sus clientes 75 % más rápido.

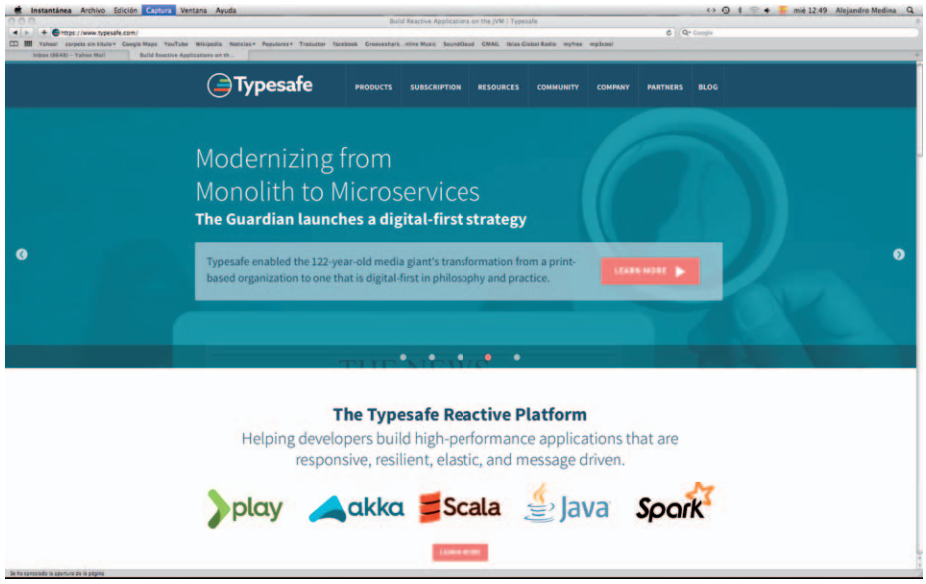
Elasticidad: Responde y permite escalar la infraestructura ante grandes cargas de trabajo, en cuyo caso el sistema reacciona incrementando o disminuyendo la cantidad de recursos asignados utilizando hardware y software estándar, y siendo eficiente en costos.

Resiliencia: El sistema debe continuar funcionando en caso de fallas. La resiliencia

en la arquitectura significa que el sistema resuelve los problemas de componentes individuales sin afectarlo todo. "Queremos que la falla sea un ciudadano de primera clase y una parte fundamental en nuestra arquitectura —señaló Ryan Knigth—. Podemos detectarla y aislarla en los diferentes componentes del sistema y del software para que no llegue al usuario final, sino a la persona apropiada, a quien se hace una solicitud de servicio para arreglarla. Pero, además, queremos mejorar las aplicaciones para que se autosanen y se restauren ante fallas". Para enfrentar así este tipo de sucesos es necesario utilizar un principio clave en la construcción de arquitectura reactiva: "Para confinar los problemas y que no se caiga la aplicación completa hay que hacer contención para aislar las fallas de los componentes, esto impide que se propague".

Control por mensajes: Tradicionalmente, las aplicaciones se han construido con el modelo solicitud/respuesta lo que hace que el usuario final tenga que quedarse esperando las respuestas de la aplicación de forma bloqueante; el mismo problema pasa entre los componentes de la aplicación que funcionan bajo el modelo síncrono de solicitud/respuesta. "Si le damos vuelta a este mecanismo, las aplicaciones y sus componentes reaccionarían al mundo en torno a ellos, sin quedarse bloqueados esperando las respuestas". Los servicios poco acoplados pueden ser asincrónicos, distribuidos a través de la arquitectura y de los servidores, y comunicarse por mensajes. Eso permite elasticidad, que los servidores escalen independientemente y que sean concurrentes 100 %, con una latencia más baja.

La experiencia indica que las fallas en el escalamiento de las aplicaciones ocurren porque se han desarrollado utilizando mecanismos bloqueantes y que además manejan estados mutables compartidos, candados y librerías con alto acoplamiento. Una de las principales razones para que una aplicación no escale es el bloqueo y en un entorno multi-hilo este modelo puede generar que cientos de hilos estén bloqueados. Hay que evitar tal escenario



Typesafe es una plataforma diseñada para manejar aplicaciones distribuidas altamente escalables en un ambiente de código abierto.

> "Queremos que la falla sea un ciudadano de primera clase en las aplicaciones, parte fundamental en nuestra arquitectura, y que sea posible recuperarse".
>
> **Ryan Knigth**

con un punto de contención único, que se consigue cuando se procesa de manera asincrónica.

Los estados mutables compartidos también trabajan mal. Se puede intentar manejar estados compartidos entre diferentes hilos, utilizando el bloqueo y cualquier mecanismo que permita determinar cómo sincronizarlos y establecer responsabilidades de acceso al estado. "Si se comparte el estado, el código se vuelve indeterminado. Esto ocasiona que la aplicación esté bien en el momento en que se prueba con cientos de usuarios, pero cuando son millones y deja de funcionar, no es posible enterarse de qué pasa, porque no se puede conocer con certeza el estado inmutable ni qué hilo está teniendo acceso a dicho estado. Estos problemas ocurren cuando la arquitectura tradicional tiene muchos candados y la gestión de estados compartidos es compleja".

No más aplicaciones monolíticas

Con todo esto se pueden construir más microservicios, lo opuesto a las aplicaciones monolíticas tradicionales, los cuales se comunican por mensajes. Con microservicios es posible escalar componentes individuales de una aplicación ante la demanda cambiante, no solo para manejar cargas más grandes, sino también escalar hacia abajo, cuando hay poco tráfico.

Una característica muy interesante de la arquitectura reactiva es que es inmutable por defecto, lo cual se apoya en los mecanismos construidos dentro del lenguaje de programación, que facilita el trabajo en este tipo de arquitecturas y aumenta el nivel de abstracción. Así, no es necesario pensar en las variables temporales de transformación, sino definir qué tipo de transformación se hará. También se eliminan los efectos secundarios, hay

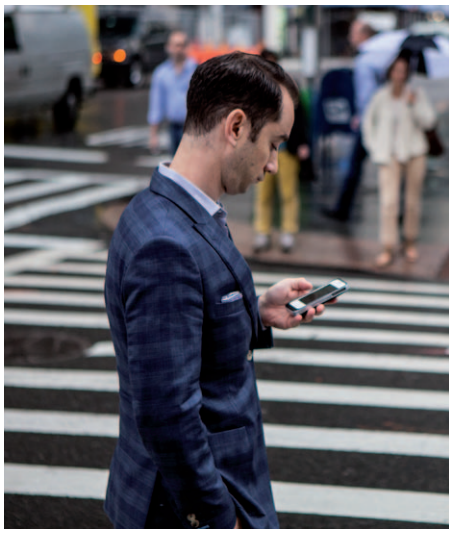


Foto: Tom Eversley, ISO Republic

> “Con arquitectura reactiva, una empresa puede atender las solicitudes de los usuarios 75 % más rápido”.

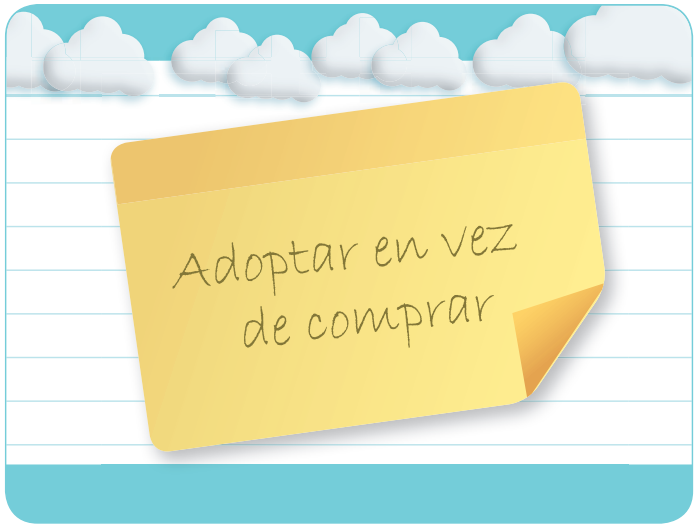
La arquitectura reactiva responde ante las necesidades del usuario y le proporciona una experiencia agradable en tiempo real.

menos dificultad en el manejo de eventos y los resultados son asincrónicos.

Otro componente importante de la arquitectura reactiva son los futuros y las promesas. “La idea es diseñar un valor futuro, crear una tarea independiente en una base de hilos que espera el evento que indica que la respuesta ha sido generada. Cuando se recibe la respuesta, se puede continuar con el procesamiento. Esto se maneja a través de herramientas denominadas “futuros” que son las que controlan la base de hilos que se queda a la espera de las respuestas sin ser bloqueantes. Como resultado, estas pueden procesarse de manera asincrónica”. ■

Talento y técnica aseguran el éxito en *cloud*

Yuji Kiriki habló de la importancia de conocer el detalle de los sistemas distribuidos, de la consistencia eventual, de qué debe tener todo programador y de adoptar software *open source.* Luis Emilio Linares se refirió a cómo ha evolucionado Microsoft para adaptarse a la nube.



Según Yuji Kiriki, adoptar software libre implica un compromiso con las comunidades innovadoras.

Para llegar a la nube, una organización necesita equipos de desarrollo talentosos, que sepan de sistemas distribuidos y conozcan las complejidades asociadas, que se hagan cargo del ambiente de producción, que operen la aplicación y respondan por ella, un proceso denominado recientemente DevOps (nombre que resulta de juntar desarrollo y operaciones).

Para Yuji Kiriki, de la empresa Seven4n, es importante que estos programadores conozcan de infraestructura inmutable, de aprovisionamiento automático de máquinas, de despliegue continuo, de integración continua, pues un cambio en el plan de desarrollo incide inmediatamente en el de producción.

También advirtió sobre los problemas del *cloud*: “La nube no es mágica, es caótica, hay latencia, hay catástrofes y bombas” y por ello deben crearse aplicaciones considerando esas situaciones.