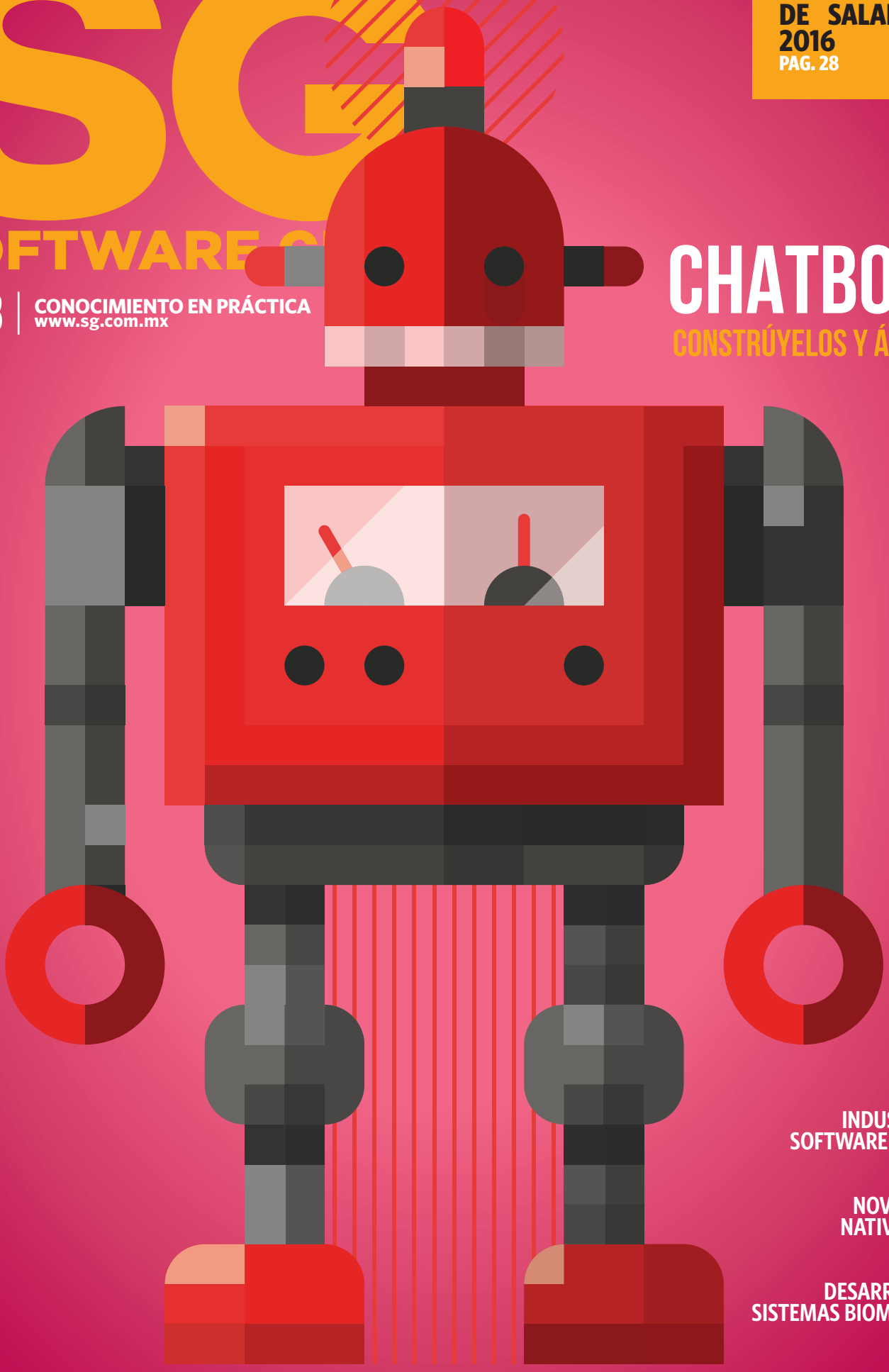
SG
SOFTWARE G
NO.53 | CONOCIMIENTO EN PRÁCTICA
www.sg.com.mx

NOVEDADES
ESTUDIO DE SALARIOS 2016
PAG. 28

CHATBOTS
CONSTRÚYELOS Y ÁMALOS



INDUSTRIA DE SOFTWARE EN PERÚ
PAG. 08

NOVEDADES: NATIVESCRIPT
PAG. 14

DESARROLLO DE SISTEMAS BIOMÉTRICOS
PAG. 44

ABIZTAR
LEARNING TECHNOLOGIES

Haciendo una diferencia en la educación en México

Cuando te capacitas en un Ambiente de Aprendizaje Abiztar, no sólo obtienes una capacitación más avanzada y eficiente a la que has probado hasta el momento; también ayudas a educar a un niño con carencias por medio de la **Fundación Ednica I.A.P.** (ednica.org.mx).

Al concluir la fase presencial de tu capacitación, recibes un dibujo hecho por el niño que ayudas en agradecimiento por tu aportación.

Conoce más en

http://abiztar.com.mx/donativo

ABIZTAR LEARNING TECHNOLOGIES

Nuestro compromiso contigo no termina con un curso, termina con el éxito de tus proyectos.

+52 (55) 5594 6411 | cursos@abiztar.com.mx | www.abiztar.com.mx

ednica
educación con niños, niñas, adolescentes y jóvenes en situación de calle

MEMBER OMG OBJECT MANAGEMENT GROUP · PMI Registered Education Provider Project Management Institute · UML · SysML · BPMN · CMMI · MDA · GRAILS · Java · SCRUM · TOGAF

SG Campus

SGCampus

@SGCampus

www.sgcampus.com.mx

NO TE QUEDES FUERA EN EL 2017

Agrega nuestros eventos a tu calendario

http://sg.com.mx/eventos

SG®
SOFTWARE GURU

SG Talento presenta

Evento de Reclutamiento y Desarrollo Profesional

Febrero 2017, CDMX

Descubre las mejores ofertas laborales en México y USA, y conoce atractivas ofertas de capacitación que te pemitirán desarrollar tu carrera.

*** Expo * Pláticas * Entrevistas**
*** Descuentos en capacitación**

¡No dejes pasar la oportunidad y regístrate!

SGtalento

http://sgtalento.com/eventos

SG SOFTWARE GURU

NO.53 | CONOCIMIENTO EN PRÁCTICA
www.sg.com.mx

EN PORTADA

CHATBOTS: CONVERSACIÓN NATURAL CON ROBOTS

018

En este número aprendemos para qué podemos usar chatbots y cómo desarrollarlos.

I INDUSTRIA Y EMPRESAS

Industria de Software en Perú 008

Entrevista con Ernesto Quiñones 010

Mejores Empresas para Trabajar 035

T HERRAMIENTAS Y TECNOLOGÍAS

Radar 012

Desarrollo de Aplicaciones Móviles con Nativescript 014

V VOCES

5 Puntos Clave para Construir una Comunidad en Línea 017

Cuatro Maneras en que la IA Cambiará Todo 026

Primero el Internet de las Cosas, ¿y luego? 027

El Caso de PHP y los Salarios 036

C COLUMNAS

Tejiendo Nuestra Red 006

Prueba de Software 040

Programar es un Modo de Vida 048

P PRÁCTICAS

Cómo Estructurar el Contenido de tu Aplicación 038

Sistemas Biométricos 044

O EN CADA NÚMERO

Noticias y eventos 005

Hardware 050

Humor 051

Carrera 052

Lo mejor para el 2017

Los profesionales de TI nos enfrentamos con un movimiento constante en todos los aspectos que nos exige estar en actualización continua. Para Software Guru es importante mantenerte a la vanguardia con temas en continua evolución como lo es la manera en que interactúan los sistemas con las personas, de forma cada vez más "inteligente". Es por ello que decidimos enfocar esta edición en el tema de chatbots, el medio cada vez más común en que a través de un chat manejado por un robot de software las compañías resuelven nuestras dudas en línea y dan atención al cliente. Te proporcionamos una base para que construyas un chatbot y desarrolles este tipo de sistemas.

En SG pensamos que es importante que tengas un referente para el análisis de salarios de la región por lo que en este número encontrarás el estudio de salarios 2016. En este estudio encontrarás un indicador de oferta y demanda de los distintos perfiles de tecnología para distintas áreas de la región.

Nos sentimos orgullosos de finalizar un año lleno de éxitos para SG. Deseamos para ti y tus seres queridos unas felices fiestas y un próspero 2017.

Agradecemos profundamente tu preferencia, ya que sin tí todo esto no sería posible. Esperamos verte en 2017 en alguno de nuestros eventos: CTO Forum, Data Day, Feria de Reclutamiento, SG Next, Hackatour, Dev Day 4 Women; como siempre, te tendremos gratas sorpresas.

El equipo de Software Guru

SG es posible gracias a la colaboración de

Dirección Editorial **Pedro Galván** | Dirección de Operaciones **Mara Ruvalcaba** | Dirección Comercial **Claudia Perea**
Coordinación Editorial **Ana Loyo** | Arte y Diseño **Oscar Sámano** | Suscripciones **Mariana Torres**
Consejo Editorial: Luis Daniel Soto | Gunnar Wolf | Luis Vinicio León | Hanna Oktaba
Ariel Jatuff | Emilio Osorio | Gloria Quintanilla | Jorge Valdés

COLABORADORES EN ESTA EDICIÓN
Arístides Palma, Juan Lombana, Basilio Briceño, Ernesto Quiñones, Carlos Duchanoy, Ernesto Riestra, Oswaldo Herrera, Misael León, Juan Manuel Reyes, Alejandro Mercado, Amín Espinoza, Enrique Ortegón.

EQUIPO SG
Coordinación de servicio **Yoloxochitl Juárez** | SG Campus y proyectos especiales **Hilda Ramírez** | Business Development **Erika Rivero**
Developer Relations **Luis Sánchez** | Servicios online **Ivett Sánchez** | Alianzas **Lorena Mireles**
Contacto: info@sg.com.mx

SG Software Guru es una publicación trimestral editada por Brainworx, S.A. de C.V., San Francisco 238 Altos. Col. Del Valle. *Reserva de Derechos al Uso Exclusivo: En trámite. ISSN: 1870-0888. Registro Postal: PP15-5106. Distribuido por Sepomex.*

Noticias

DEV DAY 4 WOMEN 1

Con una asistencia de más de 300 personas y 23 ponentes se llevó a cabo la cuarta edición del Dev Day 4 Women en las instalaciones del ITAM, Ciudad de México. Se realizaron 5 sesiones largas, 11 Ignites y los asistentes tuvieron la oportunidad de recibir experiencias y motivación de parte de mujeres exitosas durante el Fireside Chat. Las comunidades también tuvieron su espacio en el evento en donde se presentaron 9 representantes a hablarnos de sus iniciativas. Durante el cierre se presentó el Devotion Show, un vlog que se transmite quincenalmente en el canal de YouTube de Software Guru, y se anunció que la siguiente edición del evento será el 2 de junio 2017 en Guadalajara. Más información en devday4w.com

GARTNER CIO & EXECUTIVE SUMMIT 2

La edición 2016 de la cumbre Gartner para CIOs y ejecutivos de TI de Latinoamérica se llevó a cabo en Cancún, Quintana Roo del 14 al 17 de noviembre. Durante el evento se abordaron los temas más relevantes de la agenda para CIOs, que traza el ascenso del ecosistema digital. A pesar de que solo hubo un pequeño incremento en el presupuesto de TI de las empresas (2.2% a nivel mundial), el gasto global en digitalización va en aumento.

HACKATHON BBVA 3

El primer maratón de programación dentro de la Torre Bancomer contó con 280 participantes. Los más de 60 equipos tuvieron dos días para desarrollar y presentar sus proyectos, siguiendo cualquiera de ocho retos diferentes. El ganador fue Chepo Ventas, un bot de reconocimiento de productos para rápida compra/venta en Mercado Libre. Ganó el premio principal de $75,000 pesos, más premios adicionales de los patrocinadores, incluyendo una beca completa para el Wizeline Artificial Intelligence Academy.

MICROSOFT TECHTOUR MX 4

De la mano de Microsoft México y Software Guru llegó a Puebla, Querétaro y León con más de 350 asistentes el Microsoft Technical Community Tour México, un evento gratuito donde se recorren ciudades de la república para charlar sobre herramientas y prácticas de desarrollo y gestión de aplicaciones en la nube. Estas charlas se dividieron en dos sesiones por ciudad: Sesión para Developers, donde se habló de Azure, Xamarin y chatbots; Sesión para IT Pros, donde se abordaron temas como DevOps, contenedores e internet de las cosas. Este tour continua en el 2017, si quieres participar puedes consultar más información en Microsoft Technical Community Tour México.

JÓVENES MEXICANOS"ROBOTIX" GANAN EL 5TO. LUGAR EN LA OLIMPIADA MUNDIAL DE ROBÓTICA 5

Por tercer año consecutivo alumnos de RobotiX participan representando a México a nivel internacional en una competencia de robótica donde obtuvieron el quinto lugar, logrando el Score máximo que permite la competencia. Cabe destacar que el año anterior los equipos RobotiX habían quedado dentro de los 25 mejores, y la meta para este año era estar en los primeros lugares del mundo.

MoProSoft Revive

Por Hanna Oktaba

La Dra. Hanna Oktaba es profesora de la UNAM y su objetivo principal es generar conocimiento a través de la creación y promoción de estándares.
@hannaoktaba

No sé qué está pasando pero en las últimas semanas ha revivido el interés por MoProSoft. ResearchGate me está felicitando semanalmente por tener más de 20 lecturas del documento de MoProSoft 1.3 y los lectores no son nada más de México, lo leen en América Latina, en EU y hasta en la India. Los Institutos Tecnológicos de la república me piden conferencias sobre este tema y otros peregrinan de Lázaro Cárdenas a la Ciudad de México para pedirme una plática en la Facultad de Ciencias. Un maestro de UPIICSA del IPN trajo a cinco alumnos de primer semestre para que me entrevistaran sobre MoProSoft. Una de mis amigas consultoras, que normalmente da capacitaciones sobre CMMI o Project Management, me comentó sorprendida que una institución gubernamental le pidió un curso de qué creen – de MoProSoft. Y para derramar el vaso de mi felicidad un joven, @Netozack, mandó un tuit: *"Revisando MoProsoft, muy interesante conocer este estándar @hannaoktaba, es esencial para el desarrollo de software en México el tenerlo".*

No lo puedo creer. ¿Tenían que pasar 14 años para que se apreciara lo que hemos creado en México? Pero más vale tarde que nunca :) . Para las nuevas generaciones, que todavía no lo conocen, decidí hacer un pequeño recorrido por el modelo, empezando con la anécdota de su creación.

En agosto de 2002 la Secretaría de Economía dio luz verde a la AMCIS (Asociación Mexicana para la Calidad en Ingeniería de Software) para crear un modelo de procesos que ayudaría elevar el nivel de capacidades a la incipiente industria de software mexicana. Para un sábado de finales de agosto, como presidenta de la AMCIS convoque a la primera reunión a todos los interesados en el proyecto. Y ¿qué creen? ¡Llegaron puras mujeres! En esta primera reunión, bajo la batuta de Gloria Quintanilla, definimos la estructura de procesos indispensables para organizar bien el trabajo de una empresa de software (ver fig. 1). Las novedades principales con respecto a otros modelos, como CMMI o ISO/IEC12207, fue la inclusión explícita del proceso de Gestión de Negocio y la integración de los procesos entre sí. Otra innovación fue la definición del patrón (plantilla) para documentar los procesos, que tiene muchos más elementos que no van a encontrar en ningún otro modelo.

En la siguiente reunión se incorporaron tres compañeros y en total participamos 8 mujeres y 3 hombres. Todos trabajaron voluntariamente y en sus ratos libres, nos reuníamos presencialmente los sábados (no existía skype, ni hangouts), el intercambio de documentos se hacía por correo electrónico (no existía dropbox, ni drive) y la primera versión de MOPRIS (Modelo de Procesos para la Industria de Software – así se llamaba en su primera versión) estuvo lista a finales de diciembre de 2002. Tiempo record, que yo sepa, para definir modelo de procesos!!!! Luego, gracias a Laura Almada Vélez, lo rebautizamos como MoProSoft y este nombre nos gustó a todos, incluyendo a la Secretaría de Economía, que luego bautizó a su programa para el apoyo a la industria de software como PROSOFT.

Para definir el contenido de los procesos los colaboradores escogieron uno en función de su conocimiento y experiencia. Así, aunque usted no lo crea, Mara Ruvalcaba – la directora de operaciones de la revista SG, fue responsable por el contenido del proceso de Gestión de Negocio. Los que lo han implementado bien reconocen, que tener un plan estratégico para la empresa, los objetivos y metas cuantitativas del negocio y, controlarlos es una aportación muy importante que incentiva a los directivos a ser más visionarios pero también obliga a ser más disciplinados.

Angélica Sú armó el contenido del proceso de Gestión de Procesos. Este proceso ayuda a definir las formas de trabajar que necesita la organización en función de los objetivos del negocio, controla su aplicación y las mejoras. Es el mecanismo a través del cual la organización acuerda cómo quiere trabajar, por ejemplo introduciendo prácticas ágiles, y les ayuda a verificar si realmente estas prácticas les sirven al negocio.

Francisco López Lira se ocupó del proceso de Gestión de Proyectos que abarca desde la búsqueda de proyectos nuevos (ventas) hasta el control cuantitativo y cualitativo de los proyectos ya contratados. Este proceso tiene un valor particular para las empresas pequeñas para poder controlar su crecimiento cuando les llueven proyectos.

Alfonso Martínez fue responsable por la integración y coherencia del proceso de Gestión de Recursos. Este proceso es peculiar, porque tiene tres subprocesos enfocados en diferentes tipos de recursos. El cuidado de recursos humanos quedó plasmado en el subproceso Recursos Humanos y Ambiente de Trabajo a cargo de Maria Elena Rivera y el subproceso de Bienes,

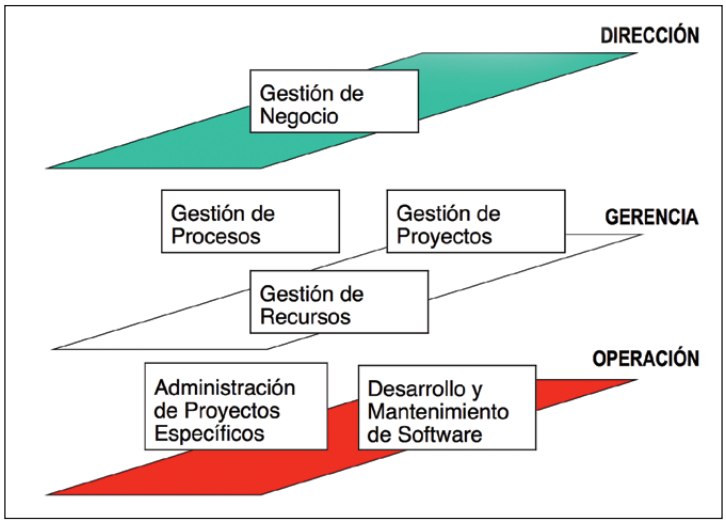


Figura 1. Procesos de MoProSoft.

Servicios e Infraestructura nos ayudó a definir Miguel Ángel Flores. Otra aportación de MoProSoft a los modelos existentes fue la inclusión del tercer tipo de recursos que hay que cuidar en este tipo de empresas. Lo llamamos el subproceso de Conocimiento de la Organización, a cargo de Yolanda Fernández, el cual se refiere a la creación, resguardo y mantenimiento de la Base de Conocimiento de la organización. En una organización de desarrollo de software tener acceso al conocimiento generado es fundamental.

Finalmente, María Julia Orozco incorporó prácticas básicas en el proceso de Administración de Proyectos Específicos y yo con Claudia Alquicira fuimos responsables por el contenido del proceso de Desarrollo y Mantenimiento del Software. Los dos procesos conforman la base para las actividades operativas de una empresa de software. La redacción de partes generales y la revisión de la consistencia del modelo estuvo a cargo de Claudia Alquicira y su servidora.

Durante los 14 años MoProSoft ha transitado por varios caminos. Desde volverse norma mexicana NMX-059-NYCE en 2005 a ser reconocido a nivel internacional como estándar ISO/IEC 29110 para Very Small Entities. Fue probado en empresas españolas, colombianas, brasileñas, peruanas, chilenas, ecuatorianas y uruguayas gracias a los proyectos COMPETISOFT y RELAIS. En México más de 500 empresas lo han implementado con mayor o menor éxito y como ISO/IEC 29110 se está implementando en Canadá, Tailandia, Irlanda entre otros. Ni hablar de Japón cuyo representante en el ISO/IEC me regaló un ejemplar de su guía para los procesos de operación en japonés :).

CONVOCATORIA PARA RENOVAR MOPROSOFT

En estos 14 años se han popularizado nuevas estrategias de desarrollo como Agile+Lean y DevOps; también han crecido nuevas generaciones de desarrolladores para las cuales leer textos largos con tablitas de "word" es aburridísimo. Creo que MoProSoft tiene muchas ideas todavía valiosas pero ya amerita una "manita de gato" y una forma de presentación más moderna.

Por lo tanto, como hace 14 años, lanzo un llamado a acción para los que tienen ideas y ganas de participar en la renovación de MoProSoft. Interesad@s favor de comunicarse a hanna.oktaba@ciencias.unam.mx , también se aceptan hombres :).

Industria de **Software en Perú**

Continuando con la serie de artículos que abordan la industria del software en américa latina, en esta ocasión hablaremos de la situación en Perú.

Según el Consejo Nacional de Competitividad Peruano, la economía peruana ha crecido significativamente a lo largo de los últimos años. Entre el 2001 y 2012, el PBI (Producto Bruto Interno) pasó de US$130 a US$328 mil millones, y las exportaciones totales, en el mismo periodo crecieron de 16% a 25.5% del PBI. Dentro de estas cifras, la participación de las exportaciones de alta tecnología solo incrementaron de 4.3% a 6.3%. De esta manera, la inversión constante en Ciencia, Tecnología e Innovación (CTI) impulsa la mejora y renovación de bienes y servicios así como el cambio estructural hacia una sofisticación tecnológica y una diversificación de la matriz productiva del país.

Es por lo anterior, que dentro de la Agenda de Competitividad 2014-2018 se incluyen acciones que permitan favorecer a la industria tecnológica a través de estrategias como estrechar la vinculación del sector empresarial con la academia para que el conocimiento generado circule y se utilice para incrementar el valor agregado de las empresas y del conjunto de la economía. Así mismo, el área de TIC se convierte en un área estratégica de crecimiento para el Consejo Nacional de Competitividad.

De la mano de esta Agenda de Competitividad evolucionan y se crean iniciativas que permiten que la Industria del Software sea impulsada de manera importante.

APESOFT

En el año 2000 surge una entidad privada sin fines de lucro con el objetivo de promover la industria nacional del software, mejorar la competitividad de sus afiliados y fomentar las exportaciones de software Peruano, la Asociación Peruana de Productores de Software (ApeSoft).

ApeSoft está compuesto actualmente por 300 empresas que generan el 80% del software, representando para Perú 300 millones de dólares anuales. Algunas de estas empresas desarrolladoras de software han logrado obtener certificaciones de calidad bajo estándares internacionales como el CMMI, ISO9000, IT MARK, siendo interés de ApeSoft que todos sus afiliados cuenten con algún tipo de acreditación de calidad con la finalidad de mejorar la competitividad del software Peruano.

La asociación representa los intereses comunes de las empresas miembros desde tres áreas principales:

• Promoviendo simultáneamente objetivos nacionales e intereses profesionales y económicos de la industria.

• Representar al sector ante las instituciones públicas y privadas, nacionales y extranjeras para la obtención de beneficios directos e indirectos.

• Recogiendo, evaluando y distribuyendo información relevante para las compañías de software.

CITESOFTWARE

El Centro de Innovación Tecnológica de Software fue creado con el fin de promover el desarrollo y la innovación tecnológica de la industria peruana del software. Proporciona a las empresas de la cadena productiva de software servicios tecnológicos que le ayudan a fortalecer su competitividad y mejorar su productividad. Así mismo, servir de soporte para promover las soluciones tecnológicas y el uso de herramientas informáticas a las PYMES de otros sectores conformantes de la Red de Cites. Los servicios de este centro incluyen: Acreditación de su producto de software, Proyectos de I + D + i, capacitación, eventos, foros de difusión para Pymes y servicios de vigilancia tecnológica.

Este centro surge como una iniciativa de ApeSoft para contribuir en la inclusión digital de Perú.

CLÚSTER LIMATECH

Se trata de una iniciativa tipo Clúster formada por empresas e instituciones especializadas y complementarias en actividades relacionadas al software como: Consultoría, outsourcing de servicios, desarrollo, distribución y comercialización de software. Estas empresas, ubicadas en Lima, interactúan entre sí creando un clima de negocios en el que todos pueden mejorar su desempeño, competitividad y rentabilidad. Los ejes estratégicos de esta iniciativa son:

1. Consolidación institucional del Clúster.

2. Fortalecimiento de la Competitividad del Clúster.

3. Desarrollo de una oferta conjunta de productos y servicios innovadores.

4. Desarrollo de mercados y oportunidades comerciales

EXPO DATA GOB PERÚ

Se trata de un evento que tiene como objetivo estratégico reunir temas de TIC's que los funcionarios públicos peruanos deben incorporar en el análisis de toma de decisiones a niveles estratégico y operativo. La idea principal es proponer temas que permitan encontrar las causas de las deficiencias que limitan en la mejora de la competitividad.

Aborda 6 áreas principales: Diseño-Infraestructura, Networking, Energía, Seguridad, Virtualización, Cloud Computing.

LEY NO 30309

Al documentar el caso de la Industria Peruana del Software, resalta de manera importante la existencia de esta Ley que refleja lo importante que es para este país incentivar este sector. Dicha Ley promueve la Investigación Científica, Desarrollo Tecnológico e Innovación Tecnológica, busca que más empresas incursionen en proyectos de I + D + i que impacten en la competitividad de sus organizaciones. Esta ley concede una deducción tributaria sobre los gastos correspondientes a los proyectos. Los requisitos para acogerse a los incentivos son:

• Presentar su proyecto de I+D+i al CONCYTEC a través del sistema en línea.

• El equipo humano de la empresa que participará en el proyecto deberá registrarse en el Directorio DINA de CONCYTEC.

• Los contribuyentes deberán llevar en su contabilidad cuentas de control por cada proyecto, debiendo asignar los gastos de I+D+i durante el año debidamente sustentados.

• CONCYTEC emite resolución que califica y/o autoriza el proyecto; con esta resolución el contribuyente presenta su declaración de impuesto a la renta a la SUNAT con deducción por gasto en I+D+i.

• CONCYTEC fiscalizará la ejecución de los proyectos e informará los resultados de dicha fiscalización a la SUNAT.

CIFRAS IMPORTANTES

En el año 2012, el presidente de la Asociación Peruana de Productores de Software (ApeSoft) Rubén Caballero indicó que la Industria de Software representaba 250 millones de dólares al año de la cual se exportaban 25 millones de dólares, siendo el más exportado el software de administración de empresas.

Para este año, en entrevista, el presidente de ApeSoft, Juan José Miranda, nos informó que esta industria representa actualmente 300 millones de dólares al año de los cuales se exportan 23 millones oficialmente. También comenta que existen otros 23 millones en "zona gris", es decir, no oficialmente registrados como tal. El crecimiento ha sido lento pero constante de un 8 a 10% en los últimos años.

Por otro lado, un reto para esta área es la piratería ya que representa en Perú un 53%. El software como Servicio es una estrategia que se está empleando para darle movimiento a esta industria.

Desde el año 2007 se han realizado acciones en este país para exportar la industria de desarrollo de software ya que se detectó que la demanda por software era de 3,000 billones de dólares y solamente se tenía oferta por 2,000 billones.

Para este año, varias instituciones Educativas Peruanas están apostando a la Ingeniería de Software como carrera e incentivar el desarrollo de talentos.

CONCLUSIÓN

Como se puede observar, el crecimiento económico de este país ha sido detonante para apostar por la Industria del Software Peruana ya que se considera una industria detonante del crecimiento del país debido al bajo costo de inversión y gran poder de exportación. Es por ello que Perú tiene consideradas varias y fuertes estrategias para al menos hasta el 2018, año en el que seguramente el Consejo Nacional de Competitividad Peruano reevaluará los resultados obtenidos con la Agenda de Competitividad 2014-2018 y reestructurará las estrategias futuras de este sector. No perdamos de vista este país que seguramente nos estará dando sorpresas al cosechar los frutos de la fuerte inversión en este sector.

Referencias

[1] http://www.apesoft.org/

[2] http://www.novatronic.com/archives/2007/02/apesoft_prompex.php

[3] http://clusterlimatech.org/

[4] http://www4.congreso.gob.pe/pvp/leyes/ley30309.pdf

[5] http://www.cnc.gob.pe/images/upload/paginaweb/archivo/38/Ciencia_tecnologia_innovacion.pdf

[6] http://www.expodatagobperu.com/

[7] http://www.concytec.gob.pe

Ernesto Quiñones

Presidente de Lima Tech

● *Ernesto Quiñones es Presidente del Cluster Lima Tech, que es una iniciativa público-privada que busca fortalecer la Industria de Tecnologías de Información y Comunicación en Lima, Perú. El cluster Lima Tech está conformado por más de 35 empresas e instituciones que incluyen empresas de tecnología, instituciones académicas, instituciones públicas y grandes empresas.*

¿Cuáles consideras que son las áreas de especialidad de la industria de software en Perú?

Perú siempre se ha destacado por ser muy bueno en el nicho de factoría de software especializada, no la genérica; tenemos mucho expertise en verticales de negocios muy importantes como las finanzas, por ejemplo, sin embargo sabemos que el modelo de factory es bueno pero no lo más deseable.

¿Como industria, cuál consideras que es su principal reto para el año 2017?

El gran reto es llegar a consensuar la estrategia común del sector. Esto debe involucrar al estado, academia y empresa, se ve complicado pero no imposible, estoy seguro que lo lograremos.

¿Ha favorecido el teletrabajo en la atracción de proyectos y exportación de talento?

Sí, como mejora de la economía. Mucho dinero ingresa por este concepto gracias a la gran cantidad de talento individual que ahora desarrolla software para empresas como Google y Facebook, pero no como mejora estratégica del sector. Se ha generado una economía de freelancers que todos sabemos será muy complicado que escale para darle solidez a la industria.

Sobre el teletrabajo contratado localmente por empresas; ha mejorado los costos de las empresas muy poco, ya que se han transmitido las cargas sociales a estas contrataciones.

¿De dónde surge el talento? ¿Cuáles son las principales universidades que generan egresados para sector TI?

En Lima tenemos la suerte de contar con profesionales de todo el país, por lo cual tenemos muy bien identificadas la calidad de los profesionales que año a año egresan de las universidades. Por supuesto que podemos asegurar que el talento principal que más nos agrada proviene de nuestros afiliados y asociados académicos, entre ellos la Universidad San Martín, La Pontificia Universidad Católica, La Universidad Nacional de ingeniería, La Universidad Garcilazo de la Vega, el Instituto Tecnológico Iberotec y Avansys, pero obviamente a nivel nacional tenemos muchos más centros académicos de calidad con los cuales ya estamos empezando a trabajar.

¿Cuáles son los principales obstáculos para el crecimiento de la Industria de Software Peruana?

Tenemos varios frentes que atacar. Quizás los tres principales son: mejora de capacidades comerciales de los asociados; visión de largo plazo armonizada con acciones de corto plazo; aumentar y facilitar fuentes de financiamiento para proyectos de tecnología, refiriéndose a industria establecida no el ecosistema de subvenciones que ahora existe.

¿De qué forma ha impactado la Ley N° 30309 (Ley que promueve la investigación científica, desarrollo e innovación tecnológica) en el crecimiento de la industria local de software?

Aún no existen cifras oficiales del impacto. Pocas empresas se han acogido a los beneficios de esta ley, especialmente por la burocracia que tiene detrás para lograr acceder a los financiamientos, sin embargo sabemos que esto está en revisión para llegar a una sana simplificación.

A un año del surgimiento de la iniciativa "Cluster LimaTech", ¿qué impacto ha tenido en la Industria del Software Peruana?

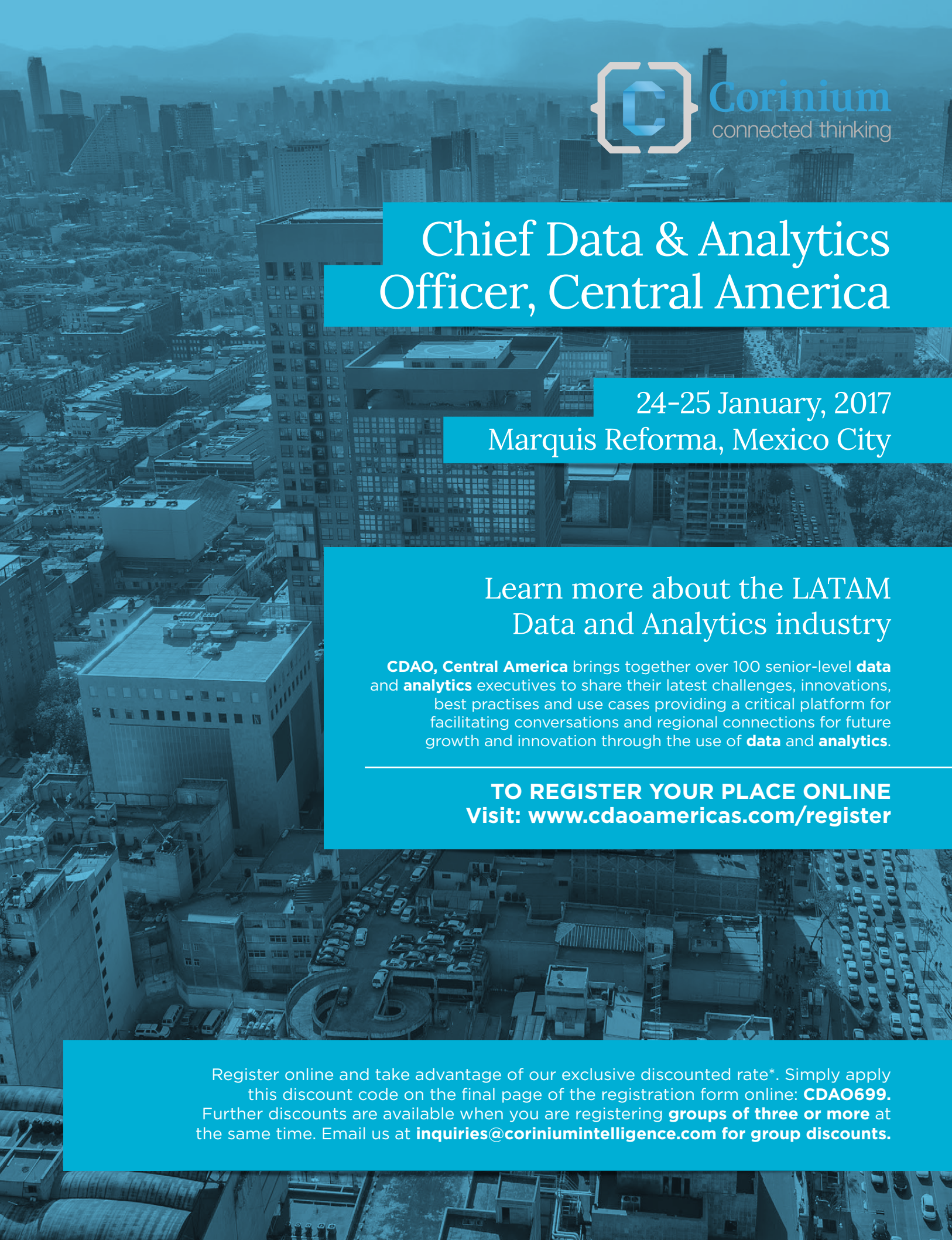
Debo precisar que aunque la iniciativa entró a funcionar oficialmente en diciembre del 2015, apenas en junio del 2016 nos encontrábamos institucionalizados, tanto que por fin pudimos ir a elecciones formalmente. El impacto en este corto tiempo ha sido muy positivo, hemos logrado articular esfuerzos con muchos actores de la industria, alinear intereses y empezar el desarrollo comercial del cluster, aún no podemos medir el impacto cuantitativamente, pero todo indica que es positivo.

¿Cómo están incentivando la colaboración entre empresas del cluster?

La colaboración es fundamental para nosotros, sin ello el cluster no tendría sentido. La fuente principal de incentivo para la colaboración son los negocios, una regla fundamental que tenemos es que al menos dos empresas unidas deben participar de las iniciativas comerciales, ello facilita que de forma natural la gente logre unirse.

¿Cuál sería la estrategia que el Cluster LimaTech debiera seguir para continuar consolidándose como institución y brindar más impulso a la Industria del Software en los próximos años?

Mi opinión personal es que debemos, y lo vamos a hacer, organizar la industria nacional para hacerla más competitiva a nivel internacional y empezar a conquistar nuevos mercados.



Corinium
connected thinking

Chief Data & Analytics Officer, Central America

24-25 January, 2017
Marquis Reforma, Mexico City

Learn more about the LATAM Data and Analytics industry

CDAO, Central America brings together over 100 senior-level **data** and **analytics** executives to share their latest challenges, innovations, best practises and use cases providing a critical platform for facilitating conversations and regional connections for future growth and innovation through the use of **data** and **analytics**.

TO REGISTER YOUR PLACE ONLINE
Visit: www.cdaoamericas.com/register

Register online and take advantage of our exclusive discounted rate*. Simply apply this discount code on the final page of the registration form online: **CDAO699.** Further discounts are available when you are registering **groups of three or more** at the same time. Email us at **inquiries@coriniumintelligence.com for group discounts.**

1 VISUAL STUDIO MOBILE CENTER

Durante su evento para desarrolladores Connect(); 2016, Microsoft realizó una gran cantidad de anuncios y lanzamientos. Algunos fueron sobre nuevas versiones de productos (C# 7, Visual Studio 2017), otros sobre la disponibilidad en otros sistemas operativos de herramientas tradicionalmente para Windows (ej. Visual Studio para Mac, SQL Server para Linux), y otros tantos sobre nuevas herramientas. De las nuevas herramientas, una que nos llama la atención es Visual Studio Mobile Center. Esta herramienta provee un conjunto de servicios en la nube para construir y gestionar aplicaciones móviles. Entre las capacidades que incluye están: construir (build) aplicaciones automáticamente con cada pull request, testing automatizado en dispositivos reales, distribución automática de nuevas versiones a usuarios beta, monitoreo de la aplicación en tiempo de ejecución para detectar fallas, analytics de uso de tus apps, backend as a service (BaaS) con servicios como autenticación de usuarios, almacenamiento y sincronización de datos offline. Visual Studio Mobile Center actualmente soporta apps iOS y Android hechas con Swift, Objective-C, Java, Xamarin y React Native.

https://www.visualstudio.com/vs/mobile-center

2 NETFLIX OSS

NETFLIX CONDUCTOR

Conductor es un orquestador de microservicios que Netflix recientemente ha hecho disponible como software libre. Conductor fue desarrollado internamente en Netflix y es utilizado para ejecutar procesos de negocio de forma automatizada y distribuida. Conductor gestiona la ejecución de flujos de trabajo (workflow) que son definidos utilizando un lenguaje de dominio específico (DSL) basado en JSON. Uno de los componentes centrales de Conductor es un servicio de máquina de estados llamado Decider, que opera una cola distribuida para gestionar las tareas. La comunicación se realiza por medio de APIs sobre HTTP. Conforme se van dando eventos en el flujo de trabajo, Decider combina la definición del workflow con el estado actual de ejecución para identificar el estado siguiente, programar la ejecución de la tarea y actualizar el estatus. Así que si estás considerando implementar una solución costosa de BPM para tu empresa, tal vez valga la pena echarle el ojo a Conductor.

https://netflix.github.io/conductor

3 ANDROID THINGS

Google ha lanzado Android Things, un sistema operativo para dispositivos IoT que permite a los desarrolladores aprovechar las herramientas y servicios del ecosistema Android para construir aplicaciones IoT. Además de las APIs de Android y librerías para acceder servicios de Google, el nuevo sistema operativo contiene una nueva librería llamada Things Support Library, la cual tiene dos APIs importantes: Peripheral I/O API para interactuar con sensores, y User Driver API para extender la funcionalidad de los dispositivos. Una de las ventajas principales de Android Things es la integración con el ecosistema Android, permitiendo utilizar los mismos lenguajes, APIs, herramientas y servicios que los desarrolladores ya utilizan para construir aplicaciones móviles Android. A la par del lanzamiento de Android Things, se han hecho mejoras a Google Weave, una plataforma y protocolo para comunicación entre dispositivos IoT.

https://developer.android.com/things

4 AWS GREENGRASS

Amazon Web Services lanzó Greengrass, una tecnología que permite que dispositivos IoT puedan operar localmente servicios de cómputo distribuido, mensajería y cacheo de datos de forma segura y sin estar conectados a internet. Greengrass opera bajo la idea de tener grupos de dispositivos IoT, dentro de los cuales hay un dispositivo central y múltiples dispositivos satélite. El dispositivo central ejecuta un componente de software llamado Greengrass Core, que es un motor de ejecución de AWS Lambda, y opera como proxy hacia los servicios en la nube de AWS para proveer capacidades como almacenamiento de datos a largo plazo y gestión remota de los dispositivos. Los dispositivos satélite utilizan el IoT Device SDK —un conjunto de librerías en C++—, para comunicarse con el dispositivo central e invocar sus servicios. Todo está diseñado para operar de manera offline, y solamente el dispositivo central necesitaría conectarse a Internet ocasionalmente para sincronizar. Greengrass se puede configurar para filtrar los datos recibidos de los dispositivos satélite y solamente transmitir a la nube aquellos que nos interesen. Greengrass gestiona la autenticación y toda la comunicación entre dispositivos y hacia la nube está cifrada. AWS Greengrass todavía no está disponible al público en general pero puedes solicitar acceso a la edición previa.

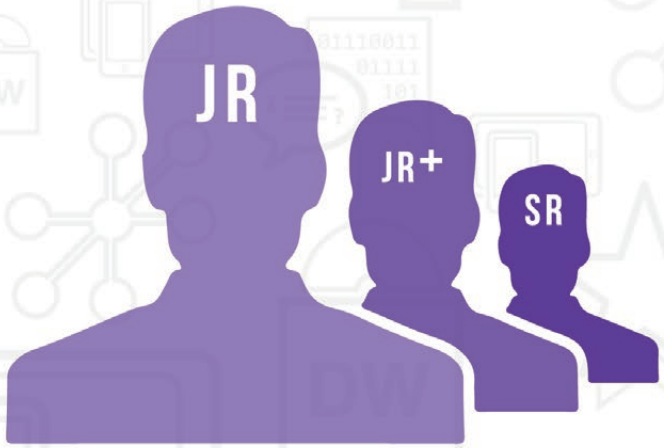
https://aws.amazon.com/greengrass

dw
a CMMI®-DEV company

EQUIPOS DE TRABAJO EXTENDIDO

Más que outsourcing de TI

En DW estructuramos un equipo a la medida de sus necesidades, garantizándole **eficacia** en el reclutamiento y **reducción de costos**



Perfiles disponibles; Analistas, Arquitectos, Desarrolladores, Testers

Experiencia en proyectos reales

Respaldo Senior

Un equipo de Desarrollo de Software orientado al modelo CMMI-DEV nivel 4

¡EXTIENDA SU EQUIPO DE TRABAJO CON NOSOTROS!

TRAINING Y CERTIFICACIONES

Una empresa de software que traslada su experiencia a través de cursos presenciales (en nivel básico, intermedio y avanzado).

- iOS-Swift 3
- Xamarin
- Angular 2
- Android
- Redes y Servidores
- Administrando Linux
- IoT Android Wear

Al inscribirte a cualquiera de nuestros cursos: **Cupón ¡10% DESCUENTO!**

cursos.dwsoftware.mx

CONTÁCTENOS

ventas@dwsoftware.mx

(33) 3030 7100

www.dwsoftware.mx

@DWSoftware

/DWSoftware

Desarrollo de Aplicaciones Móviles con NativeScript

—

Por Alejandro Mercado

El mundo móvil ha venido cambiando drásticamente, haciendo sumamente difícil para los desarrolladores crear y mantener sus aplicaciones móviles. Para alcanzar la mayor audiencia posible, las aplicaciones desarrolladas necesitan mantenerse en muchas plataformas y dispositivos. Para seguir al ritmo de este mundo cambiante, los desarrolladores comenzaron a ubicar en un lugar preponderante un solo código base para escribir y mantener las aplicaciones.

Mobile App Type	Framework
native	Android, iOS
hybrid	PhoneGap/Cordova
cross compiled	Xamarin
JIT compiled	NativeScript

Figura 1. Diferentes tipos de aplicaciones móviles y sus frameworks más populares.

En este contexto podemos incluir una larga lista de empresas y tecnologías alrededor de la creación y entrega de aplicaciones móviles de plataforma cruzada (cross-platform mobile apps), las cuales, en su mayoría plantean un solo código base para el desarrollo y entrega de aplicaciones móviles para diferentes plataformas: tecnologías como PhoneGap/Cordova, React Native, Appcelerator Titanium y Xamarin entre otras. Asimismo, podemos mencionar que el mundo del desarrollo móvil se sustenta en 4 grandes categorías: nativo, híbrido, compilación cruzada y justo a tiempo (just in time –JIT– por sus siglas en inglés,).

Las aplicaciones JIT o compiladas justo a tiempo son aquellas aplicaciones que son compiladas en tiempo de ejecución (runtime) en oposición a aquellas que son compiladas antes de la ejecución de la aplicación. Por ejemplo, en una aplicación JIT el código fuente no es compilado a código máquina nativo sino hasta el último minuto, o inmediatamente antes de ejecutar cada sentencia. NativeScript es JIT, compila justo a tiempo.

CONOCIENDO NATIVESCRIPT

NativeScript es un entorno de trabajo de código abierto (open-source mobile framework) para construir aplicaciones móviles para las plataformas iOS, Android —y próximamente Windows— usando JavaScript, CSS y XML. De manera opcional, podemos utilizar TypeScript y Angular 2 para obtener un mejor desempeño, manejo y escalamiento de nuestras aplicaciones.

NativeScript difiere de otros entornos de desarrollo de muchas maneras. La más importante es que se trata de un entorno de trabajo que verdaderamente permite escribir una sola vez y entregar a todas las plataformas. Con Nativescript podemos:

• Aprovechar el conocimiento existente (no tienes que saber Objective C, Swift o Java). NativeScript ha sido diseñado para ser aprovechado por desarrolladores con diferentes formaciones.

• Implementación de hojas de estilo CSS. Podemos cambiar la apariencia y estilo de vistas y elementos en una aplicación NativeScript de manera similar a como lo hacemos en una aplicación web, usando hojas de estilo o cambiando el estilo del objeto de los elementos en JavaScript. Sin embargo, sólo un subconjunto del lenguaje CSS es soportado.

• Acceso a las APIs nativas de la plataforma. Si el framework de Nativescript no expone una API nativa que necesitemos, podemos implementar plugins con NativeScript.

• Tu código se escribe una vez. En experiencias pasadas, trabajando con otros frameworks cross-platform, he visto que se requiere usar muchos shims (código de cuña) para que la aplicación funcione bien en las distintas plataformas; por ejemplo, puede que necesitemos agregar un poco de código para desplegar un botón sólo en la versión de Android o, tal vez, necesitamos escribir código adicional para hacer que un menú de lista se vea bien en iOS. Pero en NativeScript esto rara vez es necesario.

¿CÓMO FUNCIONA NATIVESCRIPT ?

Tal vez la característica más importante de NativeScript es su mecanismo para brindar acceso directo a las APIs nativas de cada plataforma. Pero, ¿cómo funciona?. Examinemos el siguiente código de una aplicación NativeScript para Android.

```
var time = new android.text.format.Time();
time.set( 12, 12, 2016 );
console.log( time.format( "%D" ) );
```

Te preguntarás: ¿es código JavaScript creando una instancia de un objeto Java? Sí, JavaScript crea una instancia del objeto, android.text.format.Time(), invoca su método set() y luego manda al log de la consola el valor de retorno de su método format, el cual es la cadena "12/12/16".

Veamos un ejemplo de una aplicación NativeScript para iOS.

```
var alert = new UIAlertView();
alert.message = "Hello world!";
alert.addButtonWithTitle( "OK" );
alert.show();
```

Alejandro Mercado Peña es Director de Tecnología en KMMX, socio de entrenamiento Progress-Telerik en México. amercado@kmmx.mx

Figura 2. Ejemplo de ventana de diálogo con NativeScript

Este código JavaScript crea una instancia de una clase de Objective-C llamada UIAlertView, establece su propiedad message y luego llama a sus métodos addButtonWithTitle() y show().

Vale la pena aclarar que estos casos en los que accedemos a las APIs nativas invocando objetos específicos de Android o iOS, son excepciones. NativeScript incluye muchos módulos para tareas comunes, como hacer una petición HTTP, construir componentes UI, etc. Sin embargo, a veces las aplicaciones necesitan acceso a las APIs, y el runtime de NativeScript hace este acceso muy sencillo cuando se requiere.

NATIVESCRIPT RUNTIME

El runtime de NativeScript es donde se hace la magia. Todo comienza con la máquina virtual de JavaScript que utiliza NativeScript para ejecutar comandos JavaScript. Específicamente NativeScript usa V8 en Android y JavaScriptCore en iOS. Así que todo el código que escribimos para tener acceso a las APIs nativas necesitan las construcciones y sintaxis de JavaScript que V8 y JavaScriptCore entienden.

Esta es la primera parte del proceso, regresemos a la primer línea de código del ejemplo que mostramos anteriormente para Android:

```
var time = new android.text.format.Time();
```

En el runtime de NativeScript en Android, este código es compilado justo a tiempo y ejecutado por V8. Pero, ¿cómo sabe V8 qué es android.text.format.Time()?

V8 sabe qué es Android porque el runtime de NativeScript lo inyecta, pues resulta que V8 tiene una tonelada de APIs que nos permiten configurar muchas cosas acerca del ambiente JavaScript. Podemos insertar nuestro propio código C++ para perfilar el uso del CPU de JavaScript, manejar la recolección de basura y cambiar cómo trabajan las variables internas, entre otras cosas.

Entre estas APIs existen algunas clases de "Contexto" que permiten manipular el alcance global, haciendo posible a NativeScript inyectar un objeto android global. Este es, de hecho, el mismo mecanismo que usa Node.js para hacer públicas sus APIs globales —por ejemplo, require( )— y NativeScript lo usa para inyectar APIs que permiten el acceso al código nativo.

JavaScriptCore tiene un mecanismo similar que hace la misma técnica posible para iOS.

Regresemos a nuestro ejemplo:

```
var time = new android.text.format.Time();
```

Sabemos que nuestro código corre en V8 y que éste sabe qué es android.text.format.Time() porque NativeScript inyecta los objetos necesarios en el alcance global. Ahora bien, ¿cómo sabe NativeScript qué APIs inyectar o qué hacer cuando la llamada a Time() sea realizada? NativeScript usa la técnica de reflexión (reflection) para construir la lista de APIs que están disponibles en la plataforma en que corre. La reflexión es la capacidad que tiene un programa para examinar su estructura y comportamiento en tiempo de ejecución, y es una técnica común en el lenguaje Java. NativeScript utiliza reflexión para construir una lista exhaustiva de las APIs de cada plataforma, incluyendo en este caso android.text.format.Time.

Generar esta información no es trivial desde una perspectiva de desempeño, es por eso que NativeScript lo hace antes de tiempo y embebe los metadatos generados previamente durante el paso de construcción en Android/iOS.

INVOCANDO CÓDIGO NATIVO

La respuesta a cómo NativeScript invoca código nativo nuevamente recae en las APIs de las máquinas virtuales JavaScript. En esta ocasión, veremos una serie de llamadas que nos permiten ejecutar código C++ en puntos determinados durante la ejecución de JavaScript.

Por ejemplo, el código `new android.text.format.Time()` invoca una función JavaScript, para la cual V8 tiene una devolución de la llamada o callback. Así es como V8 tiene un callback que permite que NativeScript intercepte la llamada a la función, ejecutar ciertas acciones en un código C++ personalizado y proveer un nuevo resultado.

En el caso de Android, el código del runtime de NativeScript en C++ no puede acceder directamente a las APIs de Java como android.text.format.Time. Sin embargo, Android JNI o Java Native Interface provee la habilidad para hacer el puente entre C++ y Java, así que NativeScript utiliza JNI para hacer el salto. En iOS este puente extra no es necesario ya que el código C++ puede invocar directamente las APIs de Objective-C.

Dicho lo anterior regresemos a nuestra línea de código.

```
var time = new android.text.format.Time();
```

Sabemos que este código corre en V8 y que éste conoce qué es android.text.format.Time porque NativeScript inyecta dicho objeto. Posteriormente, NativeScript corre un proceso para la generación de los metadatos para obtener esas APIs y sabemos que cuando Time() se ejecuta, ocurre lo siguiente:

1. La función callback de V8 se ejecuta.

2. El *runtime* de NativeScript usa los metadatos para saber qué significa Time() y qué necesita para crear una instancia del objeto android.text.format.Time.

3. El runtime de NativeScript usa JNI para crear una instancia del objeto android.text.format.Time y guarda un referencia a éste.

4. El *runtime* de NativeScript regresa un objeto JavaScript que provee una interfaz (*proxy*) al objeto Java Time.

6. El control regresa a JavaScript donde el objeto proxy se almacena como una variable local time.

El objeto proxy es la manera como NativeScript mantiene un mapeo entre los objetos JavaScript y los objetos nativos.

Veamos la siguientes líneas de código de nuestro ejemplo anterior:

```
var time = new android.text.format.Time();
time.set( 1, 0, 2015 );
```

Debido a los metadatos generados, NativeScript sabe todos los métodos que tiene que poner en el objeto proxy. En este caso, el código invoca el método del objeto Time set(). Cuando este método corre, V8 nuevamente invoca su función *callback*. NativeScript detecta que ésta es una llamada al método y luego NativeScript utiliza el Android JNI para hacer la correspondiente llamada al método en el objeto Time de Java.

Bastante bueno, ¿no? De manera general así trabaja NativeScript, porque convertir Objective-C y objetos Java en objetos JavaScript puede ser complejo, especialmente cuando consideramos los diferentes modelos de herencia que cada lenguaje usa.

NATIVESCRIPT CLI (COMMAND LINE INTERFACE).

La línea de comandos de NativeScript o CLI es el pegamento que mantiene todo el desarrollo unido. Esta línea de comandos está involucrada en todas las etapas del desarrollo, desde los andamios hasta la entrega de una aplicación en un emulador o en un dispositivo físico.

Para resumir y desde una vista a 10,000 pies: la CLI de NativeScript abstrae la complejidad de las herramientas nativas y SDKs, brindando al desarrollador un conjunto de comandos agnósticos a la plataforma para construir, mantener y entregar sus aplicaciones móviles.

TYPESCRIPT

Los desarrolladores NativeScript que prefieran la programación orientada a objetos pueden utilizar TypeScript.

TypeScript es un lenguaje originalmente creado por Microsoft pero posteriormente liberado como software libre. TypeScript se originó a partir de las deficiencias "percibidas" de JavaScript para el desarrollo de aplicaciones grandes, como entre los clientes externos. Los retos de lidiar con código JavaScript complejo conduce a una demanda de herramientas específicas para facilitar el desarrollo de componentes en el lenguaje. Cabe mencionar que Anders Hejlsberg arquitecto líder de C#, creador de Delphi y Turbo Pascal trabajó en el desarrollo.

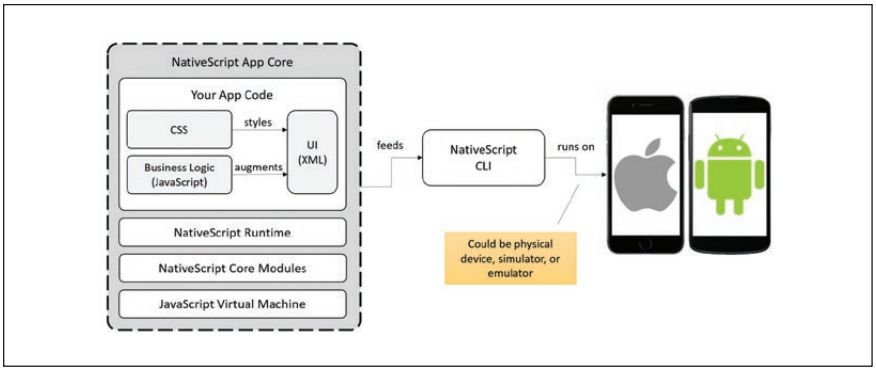


Figura 3. Componentes utilizados para producir una aplicación [2].

TypeScript actualmente es trans-compilado, es decir Typescript es un superconjunto tipado de JavaScript que compila a JavaScript plano, pero se espera que eventualmente no sea así a medida de que las máquinas virtuales JS y navegadores den soporte el estándar ES6.

ANGULAR 2

Angular 2 es la nueva versión del popular MV* framework de Google para construir aplicaciones complejas en el navegador (y más allá). Podemos decir que Angular 2 ha mutado de ser un framework a una plataforma y no simplemente una nueva versión. Angular 2 es una solución completa. Incluye rendering, compilación, vinculación, comunicación a servidor y pruebas unitarias, todo junto. Sin preocupaciones de escoger entre 20 librerías diferentes cuando solamente se necesita hacer una llamada HTTP.

Google liberó Angular 2 en septiembre de este año y ya tiene planeado liberar Angular 3 en marzo de 2017. Actualmente NativeScript se integra con Angular 2 y se espera una mayor integración y soporte para la nueva versión. Los equipos de desarrollo de ambas empresas han trabajado juntos para lograr esto. De tal forma que, si ya conoces Angular 2, estás listo para desarrollar aplicaciones móviles con NativeScript con un poder y desempeño 100% nativo.

COMPONENTES UI PARA NATIVESCRIPT

Se estima que un gran porcentaje del tiempo de un proyecto de desarrollo móvil se invierte en la creación de componentes de la interfaz de usuario y, en general, en la interfaz de usuario. De ahí que, en vez de desarrollar los componentes uno a uno, resulta mejor en términos de productividad utilizar componentes prediseñados. En este sentido, Telerik, la empresa creadora de NativeScript, provee Telerik UI que es un conjunto de componentes nativos de interfaz de usuario, que agrega muchas características avanzadas por encima de la capa de componentes por defecto que se entregan en el framework de NativeScript. Telerik tiene una larga historia desarrollando componentes UI para distintas plataformas (.Net, PHP, Node) que van desde listas, botones, formas de captura, y componentes de visualización de datos como gráficas, calendarios, etcétera.

ESTATUS ACTUAL Y FUTURO

Actualmente NativeScript se encuentra en la versión 2.4, la cual ha tenido importantes y significativas mejoras, además de más de 360 plugins, lo cual es un claro indicador de que la comunidad está produciendo mucho más código que el equipo que lleva el núcleo. Esto es un gran logro para cualquier proyecto de código abierto.

Algunas de las principales mejores en esta versión son: Flexbox layouts, web workers, soporte a Angular 2.2, Node 6 LTS, ECMAScript 2015 (ES6) y ES7. La versión 2.5 se planea estará lista para enero de 2017.

Si quieres aprender más acerca de NativeScript, el mejor lugar para empezar son las guías y entrenamientos oficiales que puedes encontrar en www.nativescript.org y en www.telerik.com

Si estás en la Ciudad de México, hay un meetup de NativeScript con reuniones mensuales a las que puedes asistir.

¡Feliz NativeScript-ing!

Referencias

[1] NativeScript. http://nativescript.org

[2] N. Branstein. What NativeScript means to Mobile Development. http://swgu.ru/s8

5 Puntos Clave Para Construir una Comunidad en Línea

Por Juan Lombana

Crear una comunidad en línea desde cero es todo un reto. La buena noticia es que si estás en la etapa del inicio puedes construirla de la manera correcta desde un principio.

Mi nombre es Juan Lombana y fui parte del equipo responsable de crear la comunidad en línea de Google Partners para varios países de Latinoamérica. Aunque esta tarea nos llevó mucho tiempo y esfuerzo, terminamos por construir una comunidad gigantesca.

Al usar el adjetivo "gigantesca" para referirme a la comunidad de Google Partners, no únicamente me refiero a su tamaño por número de miembros, sino que a las muchas interacciones entre estos miembros, que en mi opinión, son el punto más importante sobre una comunidad.

Así que después de esta grata experiencia comparto aquí los que considero son 5 puntos clave para construir una comunidad en línea.

DEFINE TU IDENTIDAD CON LA COMUNIDAD

El primer punto que necesitas entender es: ¿Quién serás tú en esa comunidad? Puedes ser identificado como una marca o usar un acercamiento más personal y escribir en nombre de ti mismo. Me parece que la primera opción es mucho mejor para grandes marcas y la segunda para expertos en determinados temas.

ENFÓCATE EN AYUDAR

La gran mayoría de las personas que están construyendo una comunidad en línea lo hacen con el fin de crecer su negocio, lo sé, lo sé. Sin embargo tienes que pensar en tu comunidad como una manera de ayudar a la gente, cuando los ayudes sin pedirles nada a cambio, algunos te comprarán y algunos no lo harán, pero lo importante es que estarás ahí para ellos de manera incondicional.

Las comunidades son para ayudar, no para vender. Si quieres hacer alguna actividad de marketing que esté orientada a ventas, el SEO o el SEM son una mejor opción para ti.

SE PROACTIVO, NO REACTIVO

Seguramente tú mismo sigues alguna página o eres miembro de alguna comunidad en línea. A cualquier persona le gusta seguir páginas que publican contenido interesante, responden rápido a los comentarios y usan cada aspecto de las redes como las encuestas, concursos y diferentes tipos de contenido.

Si quieres construir una comunidad que participe y esté altamente enganchada, tú tienes que ser el primero en poner ese ejemplo.

SÉ ORIGINAL

Cualquier persona, de verdad cualquiera, puede abrir una página de Facebook y compartir contenido que viene de otras fuentes, o usar Twitter únicamente para retwittear. Ese tipo de estrategia no te va a llevar muy lejos a la hora de crear una comunidad, no añade valor y te convierte en un ladrón de ideas.

Para realmente añadir valor tienes que compartir contenido que nadie más haya compartido nunca, contenido producido por ti, con esto no quiero decir que este contenido tiene que ser de compleja producción, simplemente original.

LA AGENDA ES LA REINA

Aunque la frase original es "El contenido es rey" realmente pienso que lo segundo más importante es tener un plan de contenidos y fechas. Al hacer esto siempre estás interactuando con tu comunidad y además lo haces de manera que las personas se acostumbran a la frecuencia con la que compartes contenido, ya sea una vez al día, a la semana o al mes...una agenda crea orden para ti y para todos los que son parte de la comunidad.

Necesitas tener un plan de al menos un mes de contenido antes de abrir tu comunidad, siempre es bueno tener un equipo con reservas.

CONCLUSIÓN

Si bien, estos cinco puntos te deberían ayudar bastante a la hora de crear tu comunidad en línea, me encantaría advertirte que esto requiere mucho más dedicación de lo que parece.

Tienes que estar muy comprometido con la meta y el motivo de esta comunidad por un largo tiempo y ser sumamente paciente, asegúrate de que tu agenda te permite este tiempo.

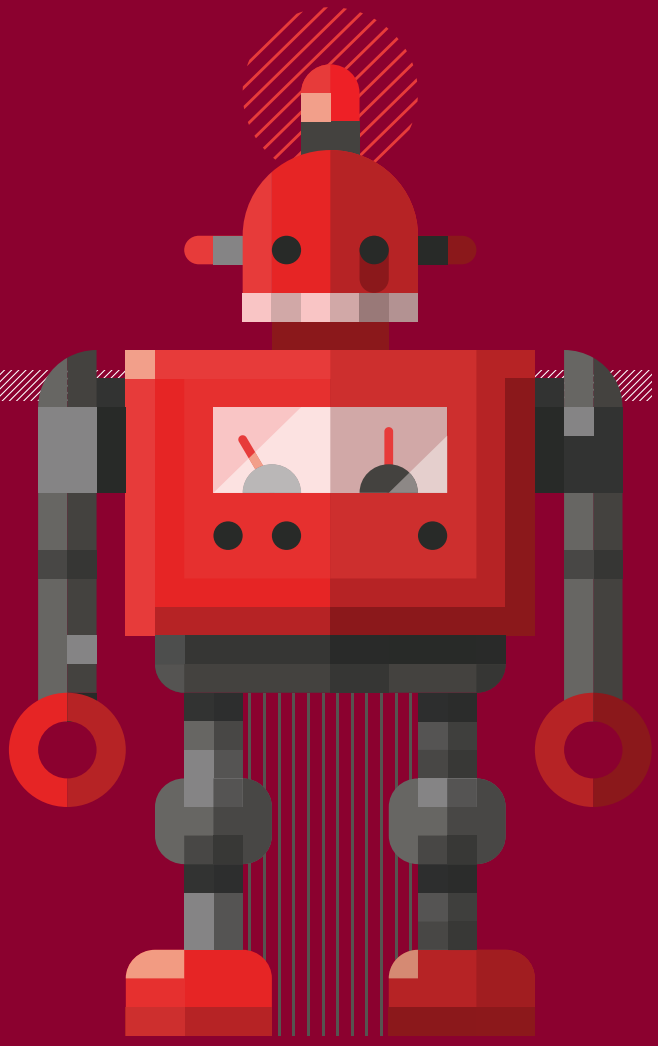
Por otro lado, si realmente estás comprometido, no dejes que nada ni nadie te detenga ya que cuando logras crear una comunidad enganchada en línea, el sentimiento es invaluable.

Si tienes alguna pregunta, por favor contáctame en @Juan_Lombana.

Juan Lombana (@juan_lombana) es Google Expert en Marketing y Google Regional Trainer para México y escribe en su blog sobre marketing digital Mercatitlán. www.mercatitlan.com

CHATBOTS:
CONVERSACIÓN NATURAL CON ROBOTS

Por Carlos Duchanoy



Los avances tecnológicos han generado un constante cambio en la manera en que interactuamos. Actualmente, las conversaciones en persona son menos comunes que por teléfono, chat o redes sociales. Esto ha generado que sea completamente normal hablar abiertamente con una máquina que llevará el mensaje a nuestro interlocutor y después nos comunicará su respuesta.

Ya que lo anterior lo encontramos cotidiano, no nos será ajeno comunicarle a la máquina nuestras necesidades y que las resuelva. Para ello se han desarrollado los bots conversacionales (chatbots), los cuales permiten interactuar con las personas ya sea como asistente personal, para atender un servicio, entretener mediante una conversación o en el área de Servicio al Cliente.

DE LAS REGLAS A LA INTELIGENCIA

La mayoría de los chatbots existentes están muy lejos de ser perfectos, por lo tanto, los usuarios prefieren evitarlos y tratar directamente con una persona. ¿A qué se debe esa preferencia? La respuesta a esto es simple, el robot no interactúa de la misma manera que lo hace un humano. No debería de sorprendernos que el cliente se disguste y exija ser atendido por un operador si el robot es incapaz de entender lo que le solicitan y sólo realiza un interrogatorio acerca de la problemática. Esto nos indica que el grado de éxito de un chatbot se encuentra directamente relacionado con el grado de inteligencia que éste tiene. Por esto es necesario comprender que no todos los chatbots son iguales y que estas diferencias pueden impactar en la experiencia de nuestro cliente.

Los chatbots basados en reglas siguen un flujo determinado de interacciones y generalmente se limitan al entendimiento de oraciones con construcciones o comandos específicos. Éste es el tipo más común de robots de servicio, eficientes para realizar tareas sencillas como indicar una hora, proveer información básica de un producto, proponer opciones de pago, capturar datos del cliente o promocionar un producto. No obstante, estos robots no entienden realmente al usuario, se limitan a seguir una serie de reglas, que en procesos complejos se vuelven demasiado complicadas de seguir tanto para usuarios como para programadores.

Todos nos hemos encontrado con este problema al llamar a un corporativo y encontrar una interminable serie de menús que buscan determinar a qué área dirigir nuestra llamada. Seguramente, la mayor parte de los usuarios preferirían que un operador tomara la llamada, entendiera su problema y lo comunicara con la persona adecuada para resolverlo. Esta interacción nos ha llevado a creer que los robots son poco inteligentes, deficientes, se confunden con facilidad y su funcionalidad es extremadamente limitada.

La interacción con chatbots debería ser muy similar a la que tendríamos con una persona para que sean exitosamente incorporados como agentes de servicio. Esto ha motivado la investigación para generar modelos que permitan al usuario interactuar con los chatbots de una manera más natural, que sean capaces de entender el lenguaje de la misma manera que lo haría una persona.

Lo anterior es investigado por el procesamiento de lenguaje natural (PLN), una rama de las Ciencias de la Computación, que analiza la forma de interactuar entre la computadora con el lenguaje humano. Es un área multidisciplinaria que engloba las ciencias de la computación y la ingeniería lingüística. Se ha estudiado desde los años 50; se realizaba a partir de complicadas reglas hechas a mano por expertos, pero no era capaz de contemplar todas las posibilidades del proceso de comunicación humana. En los años 80 mejoró considerablemente gracias a la incorporación de algoritmos de aprendizaje automático, desafortunadamente no logró la eficiencia esperada debido a las limitaciones físicas de procesamiento en las computadoras y a las escasas bases de conocimiento disponibles. Actualmente, esto ha dejado de ser un impedimento puesto que las computadoras ahora son capaces de procesar grandes volúmenes de información y se encuentran disponibles grandes volúmenes de conversaciones libres en internet.

Con lo antes expuesto, una motivación y la capacidad tecnológica para desarrollar mejores chatbots han sido las condiciones

adecuadas para fomentar la investigación y desarrollo en el área de lenguaje natural.

Así pues, los chatbots impulsados por inteligencia artificial (IA) tienen como objetivo realizar un verdadero entendimiento del lenguaje: son capaces de entender una petición del usuario a partir del contexto obtenido a lo largo de la conversación, comprender un mensaje sin importar que tenga errores ortográficos o cambios de palabras introducidos por el corrector ortográfico, y de permitir que el cliente se exprese libremente sin limitarlo a una construcción determinada.

COMPONENTES DE UN CHATBOT BASADO EN IA

Para entender mejor el funcionamiento de un chatbot basado en inteligencia artificial es necesario analizar a detalle los elementos que lo conforman. A continuación, se presenta el esquema de una de las configuraciones más usadas por investigadores y empresas reconocidas mundialmente. Algunos ejemplos se pueden encontrar en [1] y [2].

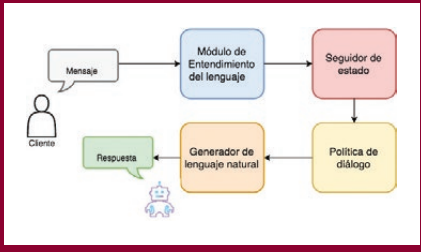


Figura 1: Esquema de un chatbot basado en inteligencia artificial

En un primer momento, el cliente interactúa con el chatbot mediante un mensaje de texto que llega al módulo de entendimiento del lenguaje, el cual procesa el texto para que el chatbot sea capaz de entender el significado. Después, el seguidor de estado se encarga de llevar el registro de la conversación, lo que permite al chatbot entender nuevos mensajes a partir del contexto de todo el diálogo. Luego, la política de diálogo permite la toma de decisiones acerca de las respuestas y acciones que debe seguir el chatbot. Finalmente, el módulo generador de lenguaje natural toma en cuenta el mensaje, el contexto de la conversación y la política de diálogo para generar la respuesta más adecuada.

Cada módulo involucra el uso de algoritmos de IA especializados para resolver el problema en cuestión. Éste es un campo con gran dinamismo en el que continuamente se están generando nuevas técnicas y herramientas.

En relación con el entendimiento del lenguaje se han desarrollado múltiples técnicas para analizar la composición, sintaxis y el significado de un mensaje. Uno de los grupos de investigadores que más ha contribuido en esta área es el Grupo de Procesamiento de Lenguaje Natural en Stanford [3], que ha hecho disponibles diversas publicaciones y herramientas bajo software libre para el procesamiento de lenguaje natural.

El módulo generador de lenguaje ha sido abordado por medio de algoritmos de inteligencia artificial que permiten obtener una respuesta para cada mensaje. El método más común es llamado secuencia a secuencia [4] (sequence to sequence) y primordialmente se ha usado para realizar labores de traducción.

El módulo para seguimiento de estado es uno de los mayores retos a resolver para poder implementar exitosamente un chatbot, y todavía no existen soluciones adecuadas a este problema. Por esta razón es necesario involucrar a la mayor cantidad de investigadores para obtener una solución. Ésta es la motivación de The Dialog State Tracking Challenge [5], el cual se encuentra impulsado por múltiples instituciones académicas y empresas para incentivar a investigadores alrededor del mundo a poner a prueba diferentes técnicas para realizar el seguimiento de una conversación.

PANORAMA EN NUESTRA REGIÓN

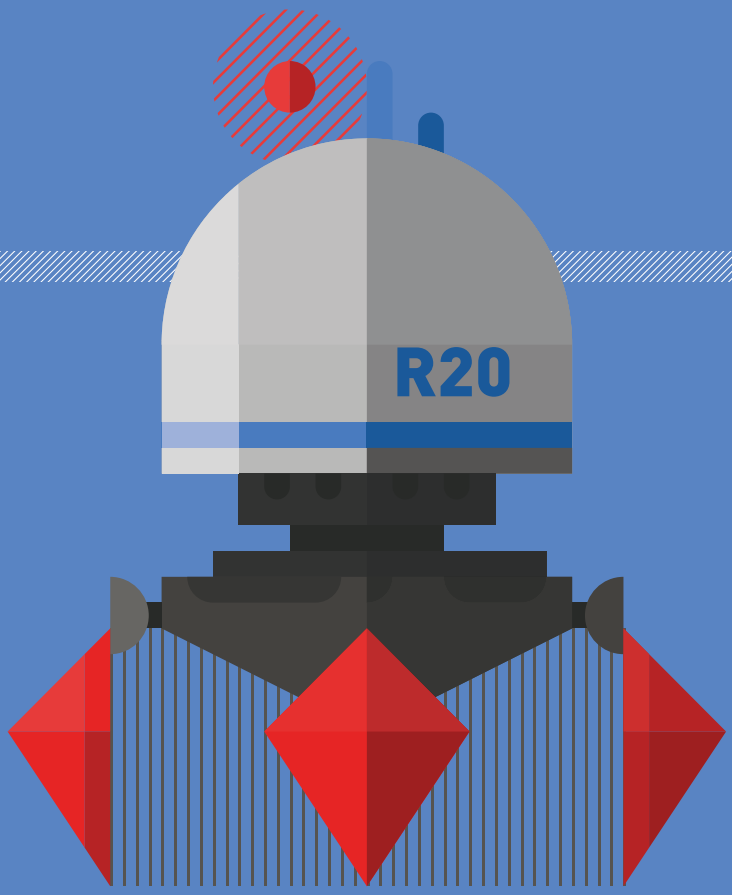
De este breve acercamiento al funcionamiento de los chatbots, pueden resaltarse dos puntos muy importantes: en primera instancia, no se ha logrado desarrollar un chatbot capaz de entender el lenguaje en su totalidad, a pesar de todos los esfuerzos invertidos; y, en segunda instancia, la mayor parte de toda la investigación realizada se ha enfocado en el idioma inglés. Esto nos hace pensar que aún falta mucho para contar con un chatbot inteligente centrado en el español. Sin embargo, esto no es del todo cierto, ya que académicos de múltiples países hispanoparlantes se encuentran realizando investigaciones para generar técnicas de inteligencia artificial adecuadas a nuestro idioma.

Sabemos que la investigación necesita la inversión de la industria privada, es por ello que hay industrias interesadas en invertir capital en el desarrollo de estas tecnologías. Existen grandes empresas multinacionales como Google o IBM que buscan desarrollar estos servicios en español, al tiempo que existen empresas mexicanas que también han observado este nicho de oportunidad y han apostado por el desarrollo tecnológico en nuestro país. Uno de los casos a notar es el de Gus Servicios Tecnológicos que, en su interés por mejorar la atención al cliente, desarrolla inteligencia artificial para su asistente personal Hola Gus [6], un chatbot que busca atender al cliente de una manera natural y personalizada. Esta área de investigación se encuentra en sus primeras etapas de desarrollo, así que seguramente empezaremos a ver cada vez más chatbots en nuestro país.

Referencias

[1] B. Dhingra, et al. End-to-End Reinforcement Learning of Dialogue Agents for Information Access. http://swgu.ru/rx
[2] T.-H Wen, et al. A Network-based End-to-End Trainable Task-oriented Dialogue System. http://swgu.ru/ry
[3] The Stanford Natural Language Processing Group. http://nlp.stanford.edu
[4] I. Sutskever, et al. Sequence to Sequence Learning with Neural Networks. Advances in Neural Information Processing Systems 2014. http://swgu.ru/rz
[5] J. Williams, et al. The Dialog State Tracking Challenge. http://swgu.ru/r-
[6] Gus Servicios Tecnológicos. https://www.holagus.com

BIO. Carlos A. Duchanoy (carlos.duchanoy@holagus.com) es Doctor en Ciencias de la Computación por parte del Centro de Investigación en Computación del IPN, y actualmente colabora en Gus Servicios Tecnológicos, a cargo del desarrollo e investigación de inteligencia artificial aplicada al desarrollo de chatbots.



DESARROLLO DE CHATBOTS
PARA FACEBOOK MESSENGER

Por Ernesto Riestra

El chat es uno de los primeros medios que surgieron en los inicios del internet para la comunicación. En la actualidad, un gran porcentaje de nuestras pláticas se da a través de esta herramienta conversacional; además la creciente tendencia a usar el celular ha generado innovaciones y mejoras en la misma.

Por su naturaleza, la conversación es altamente adaptable a las herramientas digitales y su aplicación para el aprendizaje. Los chats proveen de una comunicación escrita inmediata, ideal para resolver dudas o indagar en un tema específico. La ventaja de una conversación en un chat, en comparación con la experiencia cotidiana que tenemos de navegar el internet, es que ésta mantiene una secuencia que es más fácil de comprender.

Es muy distinto entrar a un sitio de comercio electrónico o un sistema de cursos en línea y tener que buscar entre muchas categorías lo que deseamos, a preguntarle a una persona y entablar una conversación para ubicar lo que más nos interesa o necesitamos con mayor agilidad.

¿QUÉ ES UN CHATBOT?

Las herramientas de inteligencia artificial han permitido la automatización de las conversaciones a través de lo que se conoce como chatbot. Un chatbot es simplemente un agente con cierto grado de inteligencia o adaptación que puede mantener una conversación con una persona dentro de un contexto delimitado.

Los chatbots han tenido históricamente un desarrollo, por un lado, desde los laboratorios de inteligencia artificial de reconocidas universidades. Por ejemplo Alicebot, un modelo de inteligencia artificial que construye herramientas de procesamiento lenguaje natural con base en un lenguaje de markup (AIML).Y en tiempos más recientes, en las redes sociales y sistemas de comunicación masiva como Messenger de Facebook, una de las principales plataformas de comunicación vía chat, que recientemente ha incorporado módulos para desarrolladores para integrar rápidamente chatbots.

En lo subsecuente nos enfocaremos en el desarrollo de chatbots con Facebook Messenger pues la ventaja de tener acceso a millones de potenciales usuarios permite desarrollar una diversidad de casos de aplicación a través de esta herramienta.

Los elementos clave que constituyen un chatbot son:

1. Un canal digital o chat.
2. Algún esquema basado en reglas, detección y construcción de patrones para poder determinar el flujo de la conversación.
3. La existencia de algún mecanismo de comprensión del lenguaje que facilite la comunicación.

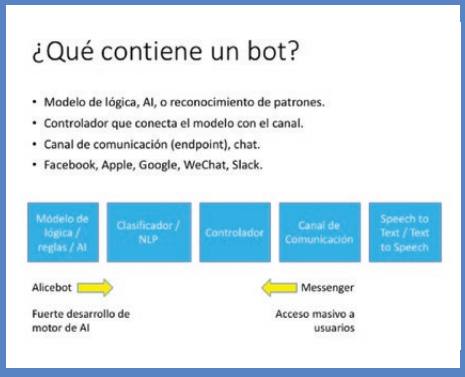


Figura 1. Componentes clave de un bot

CONVERSACIONES SIGNIFICATIVAS: FRECUENCIA VS. INVOLUCRAMIENTO

Antes de entrar de lleno a la parte más técnica del desarrollo de chatbots, es importante también acotar dos dimensiones importantes en el diseño de una conversación: la frecuencia y el nivel de involucramiento.

Lo primero se refiere a definir la frecuencia con la que se espera que un usuario consulte al chatbot o interactúe con él. Por ejemplo, en los chatbots que distribuyen noticias todos los días, se da en una u otra medida interacción con sus usuarios, mientras que en un chatbot para un juego o un cuento, una vez concluida la historia principal no es muy frecuente que los usuarios regresen.

En cuanto al nivel de involucramiento del usuario, regresando al caso del sitio de noticias el involucramiento no es muy alto, pues

la interacción puede limitarse a recibir un resumen noticioso, mientras que en un juego de roles, es posible que el usuario tenga una conversación con más involucramiento.

Un ejemplo de chatbot con baja frecuencia e involucramiento es un chatbot para un sitio de comercio electrónico: sus usuarios no suelen conversar a gran profundidad con dicho bot, y por otro lado una vez adquirido un producto no es muy probable que regresen a la misma conversación (aunque sí al sitio).

Todavía no existen muchos casos de alta frecuencia e involucramiento. Estas características suelen reservarse a las relaciones interpersonales, aunque el desarrollo de la inteligencia artificial puede dar lugar a chatbots con los que no sólo tengamos conversaciones regulares, sino un alto nivel de relacionamiento. Esta es la idea principal de asistentes de conversación como Cortana o Siri, proyectos en los que se hace una enorme inversión por parte de los corporativos detrás de su desarrollo.

DESARROLLANDO UN CHATBOT EN FACEBOOK MESSENGER

A continuación vamos a mostrar un breve tutorial para construir nuestro primer bot en Facebook. La referencia de este tutorial es el "Quick Start Guide" [1] en el sitio de desarrolladores de Facebook, pero aquí lo pondremos en español y con explicaciones en los puntos donde podrías tener problemas.

Este tutorial también está disponible por medio de un chatbot en Facebook messenger llamado YeiraBot. Así que si lo prefieres, puedes seguir este tutorial interactuando con un chatbot que te irá indicando el siguiente paso. La noción de un chatbot que nos explique cómo crear un chatbot tiene un buen grado de recursividad, que suele ser interesante en sí misma, y conduce a la reflexión. En metagraphos nos resultó interesante dejar a un lado las páginas convencionales y los sitios web de tutoriales, para crear este agente que lleve de la mano a los usuarios para crear su propio bot conversacional. Para acceder el tutorial de esta forma, ingresa a https://www.messenger.com/t/Yeirabot

La siguiente guía está basada en el funcionamiento de Yeirabot, un bot asociado a una plataforma educativa llamada Yeira. Fue desarrollado para ejemplificar cómo es posible que un chatbot sea una fuente para compartir conocimiento de una forma eficiente y novedosa. Su objetivo es explicar cómo construir un chatbot con Messenger de Facebook.

ELEMENTOS

Un chatbot en Facebook messenger consta de 3 grandes elementos:

- Página de Facebook. El bot utiliza una página de Facebook como su identidad. Esta típicamente es la página de una empresa, comunidad o causa. Cuando la gente chatea con tu bot, verán el nombre de la página y la imagen de su perfil. Puedes utilizar una página existente o crear una nueva.
- Aplicación con lógica del chatbot. Debes crear una aplicación server-side que implemente la lógica de tu chatbot. Dicha aplicación debe exponer un punto de acceso (webhook) a través del cual recibe y envía los mensajes.
- Aplicación de Facebook. Debes registrar una aplicación en Facebook (Facebook app) que contenga la configuración y permisos de tu bot, y sirve como enlace entre tu página de Facebook y la aplicación en tu servidor que implementa al bot.

CONFIGURAR SERVIDOR

Un punto importante es que la aplicación server-side con la lógica del chatbot debe estar accesible en internet por medio del protocolo https. Así que antes de continuar, configura un servidor y dominio con su certificado para https. Si no sabes como hacer esto, en la versión de este tutorial via chatbot se explica cómo configurar una instancia de Amazon EC2 que nos sirva para este propósito.

Una vez que tengamos listo nuestro servidor para la aplicación, procedamos a dar de alta nuestra aplicación de Facebook.

CREAR APLICACIÓN DE FACEBOOK

Para registrar la aplicación de Facebook visita https://developers.facebook.com/apps y da click en "Agregar una nueva aplicación". En la categoría indica que es una aplicación para Messenger (ver figura 2).

Figura 2. Registrar la App de Facebook.

La siguiente página nos preguntará detalles para configurar nuestro bot. En la sección de "Generación de Token" (ver figura 3) selecciona la página de Facebook que será la identidad de tu chatbot (necesitas tener permisos de administrador para dicha página). Al hacer esto, te aparecerá un token de acceso a la página. Cópialo.

Figura 3. Token de acceso a la página.

BIO. **Ernesto Riestra** es cofundador y líder de investigación y desarrollo en Metagraphos, firma dedicada a la creación de experiencias digitales para transformar la productividad y el conocimiento de personas y organizaciones. Es Ingeniero Mecánico Electricista egresado de la UNAM y tiene una Maestría en Mecatrónica por la Universidad Técnica de Hamburgo.

Ahora regresemos a la sección de configuración para obtener la clave secreta de nuestra aplicación. Tenemos que dar click en el botón "Mostrar" y copiar la cadena que se despliegue (ver figura 4).

Figura 4. Configuración de la aplicación de Facebook.

Con estos datos, estamos listos para configurar la aplicación server-side de nuestro chatbot.

IMPLEMENTAR LA LÓGICA

Para este tutorial vamos a utilizar una aplicación prehecha cuya funcionalidad simplemente consiste en recibir mensajes analizar su contenido, y enviar distintas respuestas dependiendo del contenido del mensaje recibido.

Descárgala en https://github.com/fbsamples/messenger-platform-samples

Dentro de la carpeta node encontrarás una aplicación en Node.js que implementa el chatbot. La aplicación es muy sencilla y toda la lógica es implementada en el programa app.js.

Para que nuestra aplicación funcione, debemos tener disponible un ambiente de Node.js. Si no lo tienes, instálalo.

Ya que tengas tu ambiente de Node.js, ejecuta el comando "npm install" para que se instalen los módulos que requiere nuestra aplicación.

Para que nuestra aplicación funcione, necesitamos modificar el archivo de configuración en config/default.json . Estos son los datos que debemos poner (ver figura 5):

- appSecret -> La clave secreta de nuestra aplicación de Facebook (la cadena de caracteres que aparece al dar click en el botón de "Mostrar", ver figura 4).
- pageAccessToken -> El token de acceso a la página que generamos cuando seleccionamos la página (ver figura 3)
- validationToken -> Esta es una contraseña que comparte la aplicación de Facebook con la aplicación en servidor. Usa cualquier cadena de texto que quieras.
- serverURL -> El url donde está disponible nuestra aplicación de servidor (ej: https://miserver.com/chatbot).

A estas alturas, ya podríamos arrancar nuestra aplicación ejecutando el comando "npm start". Sin embargo, el dilema es que por default la aplicación de Node.js escucha en el puerto 5000, pero la aplicación de Facebook le va a enviar peticiones al puerto 443 de https. Ante esto, tenemos dos opciones: la primer opción y que

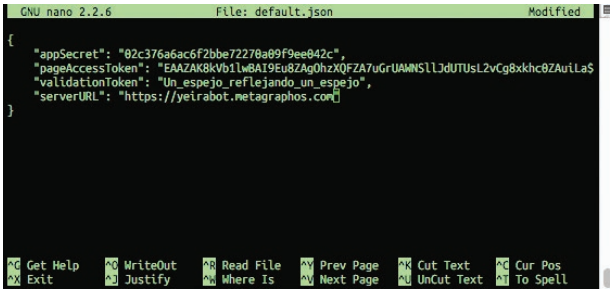


Figura 5. Configuración en la aplicación Node.js

es lo que normalmente se hace en ambientes de producción es configurar un reverse proxy en nuestro servidor web para que las peticiones recibidas en alguna dirección o carpeta sean redirigidas a nuestra aplicación de Node.js. La segunda opción sería que nuestra aplicación de Node.js escuche directamente en el puerto 443; para ello debemos editar el código de app.js para modificar el puerto donde se escucha y también para cargar el certificado SSL (ver figura 6 y 7).

```
/* jshint node: true, devel: true */
'use strict';

const
  bodyParser = require('body-parser'),
  config = require('config'),
  crypto = require('crypto'),
  express = require('express'),
  https = require('https'),
  request = require('request');

var fs = require('fs');
var app = express();
app.set('port', process.env.PORT || 443);
app.set('view engine', 'ejs');
app.use(bodyParser.json({ verify: verifyRequestSignature }));
app.use(express.static('public'));
```

Figura 6. Cambiar puerto y requerir módulo fs.

```
// Start server
// Webhooks must be available via SSL with a certificate signed by a valid
// certificate authority.
/*
app.listen(app.get('port'), function() {
  console.log('Node app is running on port', app.get('port'));
});
*/

var options = {
  key  : fs.readFileSync('/etc/letsencrypt/live/bot.metagraphos.com/privkey.pem'),
  cert : fs.readFileSync('/etc/letsencrypt/live/bot.metagraphos.com/cert.pem'),
  ca : fs.readFileSync('/etc/letsencrypt/live/bot.metagraphos.com/chain.pem')
};

https.createServer(options, app).listen(app.get('port'), function () {
  console.log('Node app is running on port', app.get('port'));
});

module.exports = app;
```

Figura 7. Cargar archivos de certificado y arrancar servidor.

Una vez que hayamos hecho el ajuste para escuchar en el puerto 443, podemos arrancar nuestra aplicación con "npm start" (o el manejador de aplicaciones de Node.js de tu preferencia) y podemos verificar que funcione bien ingresando a nuestro servidor (ver figura 8).

Una vez que el servidor está arriba y activo, es posible activar el webhook en la página de Facebook developer. Ve a la configuración de tu aplicación, y en la sección donde se configuras las opciones de Messenger da click en el botón de "Configurar

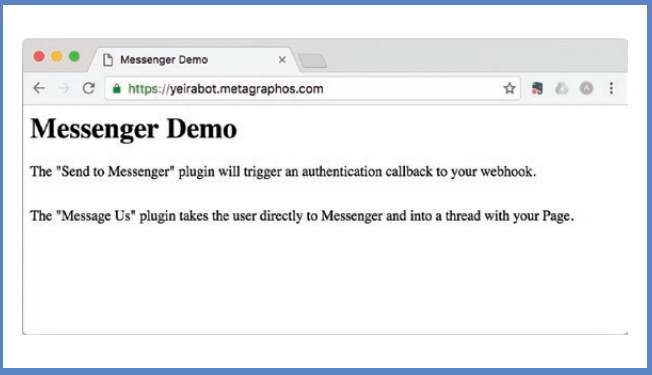


Figura 8. Página de inicio de la aplicación de servidor

Webhooks". Esto abrirá una ventana de diálogo (ver figura 9). Indica el url del webhook de tu aplicación (ej. https://server/aplicacion/webhook). Indica también la contraseña de validación (validationToken) que pusiste en tu aplicación en default.json, este es el texto arbitrario que escojiste. Activa las casillas messages y messaging_postbacks.

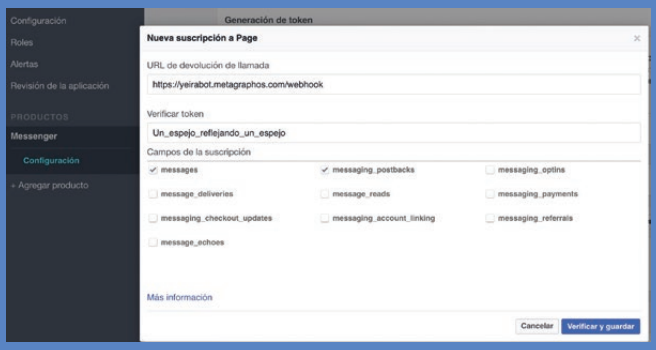


Figura 9. Configurar Webhook

Una vez que tu webhook es validado, puedes suscribir el webhook a los eventos generados en tu página de Facebook, selecciona la página en la lista de selección (ver figura 10). Esto hará que cada que tu página reciba un mensaje, se invocará el webhook.

Todo listo, ahora podrás conectarte con tu bot en la página de Messenger respectiva. En este caso: http://m.me/Yeirabot

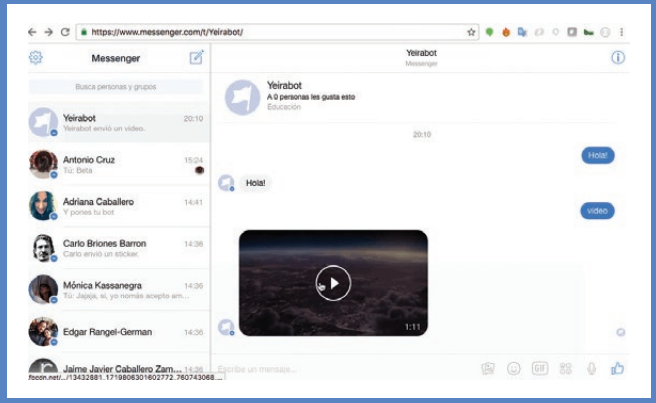


Figura 10. Chatbot en acción

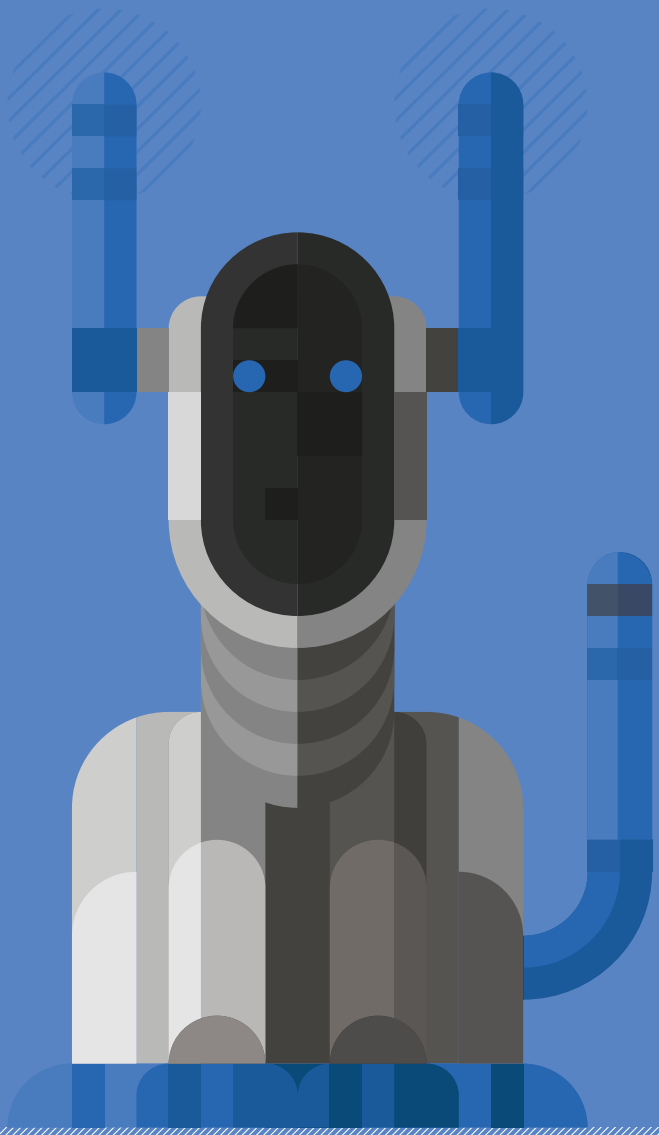
La aplicación escucha mensajes y te contesta con el mismo mensaje que recibió. Hay algunas excepciones para ejemplificar tipos de contenido especial. Por ejemplo, si escribes "image" la aplicación te responde con una imagen; si escribes "video", te responde con un video.

CONCLUSIÓN

En este artículo conocimos los pasos básicos para construir un chatbot para Facebook Messenger a partir de una aplicación prehecha. El siguiente paso para cualquiera que le interese continuar este camino es revisar la guía de desarrollador en Facebook [3] y el código fuente de la aplicación ejemplo para entender cómo es que recibe, interpreta y contesta los mensajes.

Referencias

[1] Facebook Messenger Platform Quickstart Guide. http://swgu.ru/s9
[2] ¿Qué es Yeirabot? http://swgu.ru/sa



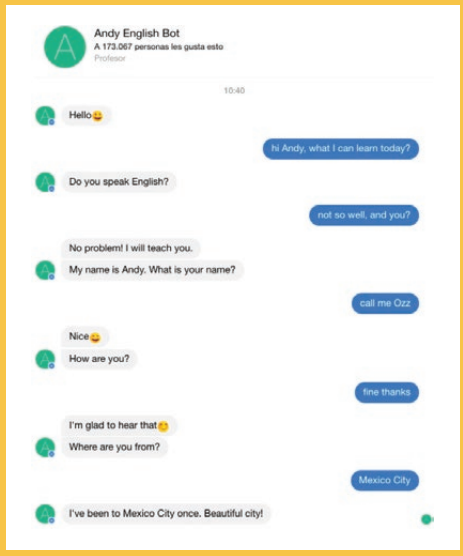
USO DE BOTS PARA INTEGRAR APLICACIONES

Por Oswaldo Herrera

Si has utilizado herramientas de mensajería como Skype, Slack o Facebook, habrás notado que existen usuarios que por lo general en su nombre terminan con la palabra bot. Si has enviado mensajes a alguno de estos, te habrás dado cuenta que ellos suelen responder a tus comandos casi de inmediato. Así es, los bots son programas que imitan el comportamiento humano (normalmente en chats) o responden a comandos específicos que ayudan automatizar tareas repetitivas y costosas en tiempo o esfuerzo.

Plataformas como Facebook Messenger o Slack proveen una base sobre la cual otras personas puedan integrar bots provistos por terceros o crear los suyos propios. Esto ha permitido que existan bots que integran otras aplicaciones. Por ejemplo, Andy English Bot (andychatbot.com) te ayuda con el aprendizaje del idioma directamente en el messenger de Facebook como si hablaras con otra persona; Meekan (meekan.com/slackhelp) te ayuda a organizar tu calendario desde el mensajer slack, creando eventos, invitando a otras personas, enviando una encuesta para que los invitados voten sobre la mejor fecha que les favorece a cada uno, agregar un punto de reunión, un enlace de streaming, recordatorios, te permite avisar que llegas tarde o posponer juntas; HealthTap (www.messenger.com/t/healthtap) provee de una conexión instantánea a más de 100,000 doctores y algunos mas con especialidades, 24 horas al día para proveer respuestas adecuadas o artículos documentales para tus preguntas sobre salud. También existen otros mas comerciales como Fynd (www.messenger.com/t/gofynd) que presume de comportarse de manera diferente para cada usuario gracias a que recuerda sus preferencias, descubre tendencias y te alerta de nuevos productos que llegaron.

Sin embargo, algunos encontrarán que la integración con aplicaciones específicas no es siempre suficiente para proveer la funcionalidad completa que se desea. Los ChatBots son una buena alternativa en estos casos y cada vez son más populares en las redes sociales. Por ejemplo, Hubot (hubot.github.com) es uno de los más soportados gracias a GitHub que lo hizo Open Source y la comunidad que lo adoptó, tiene una muy buena integración con otros servicios, se puede instalar en la nube y utiliza un concepto de Adapters para usarlo desde otras plataformas como Facebook, Skype, Slack o incluso la línea de comandos. Cog (operable.io) y Yetibot (yetibot.com) son opciones que han causado revuelo porque permiten encadenar comandos con "pipes", similar a como se hace en la consola, la salida de un comando es la entrada de otro en un proceso, así los comandos se pueden anidar a diferentes niveles como se desee.



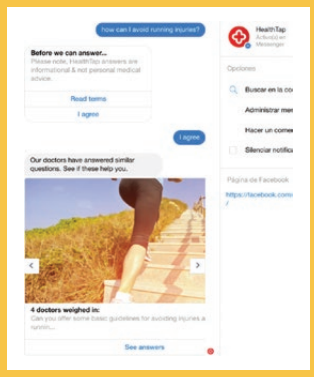
Andy English Bot

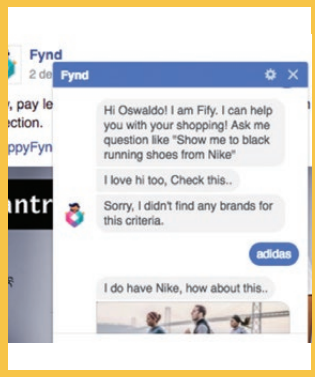
Los bots han ido ganando terreno porque proveen una mejor experiencia a consumidores, reducen tiempos de respuesta y economizan procesos donde se involucran más de una persona para atender a miles o millones de clientes; los bots agilizan los procesos de compra-venta, obtienen una retroalimentación del cliente mejor enfocada, identifican tendencias de mercado y por consecuencia generan clientes más felices.

La automatización en procesos para desarrollar un producto/servicio de manera que se facilite el acceso a la información de los diferentes flujos en IT hacia todo el staff involucrado en dicho desarrollo es una de las áreas donde se han integrados los chatbots por su habilidad de integrar otras aplicaciones que ayuden a todo el equipo encargado. En estos chats se promueve la colaboración entre desarrolladores y personas del negocio, sin importar si están distribuidos en distintas ciudades, porque se dispone de un canal donde todos pueden aportar soluciones a problemas, se generan lluvia de ideas, y si nos vamos a la parte técnica de lo que comúnmente DevOps desempeña, se puedan revisar logs, recibir alertas sobre ajustes en infraestructura, reportar datos con gráficas, responder a incidentes, crear o actualizar tickets de soporte, facilitar juntas de revisión, administrar eventos en el calendario para el staff, compartir documentos, tener control de la configuración, administrar los contenedores en la nube y la infraestructura orquestada, por nombrar algunos.

Mejor conocido como ChatOps, intentan llevar los flujos que ya se usan para administrar la aplicación y su infraestructura hacia el mismo canal primario de comunicación y colaboración que ya existe entre desarrolladores, gente de operaciones, del negocio y automation, el chat. Es ahí donde frecuentemente nuevas aplicaciones pasan de una idea

al MVP en semanas en lugar de meses debido a la habilidad que se tiene de rápidamente construir y desplegar aplicaciones durante todo su ciclo de vida. Un ChatOps además, ayuda a disminuir la curva de aprendizaje para nuevos miembros en el equipo y se reduce la necesidad de revisar documentación pues conocen los flujos de trabajo de primera mano. Hay que tomar en cuenta que ya existe un amplio mercado de integraciones entre aplicaciones que permiten a los equipos en ChatOps y usuarios de mensajería tener a la mano herramientas útiles con un mínimo esfuerzo. No obstante, cuando se requiere un nivel de integración más avanzado, más completo, con cierta personalización o con mayor flexibilidad es cuando se requiere un chatbot.



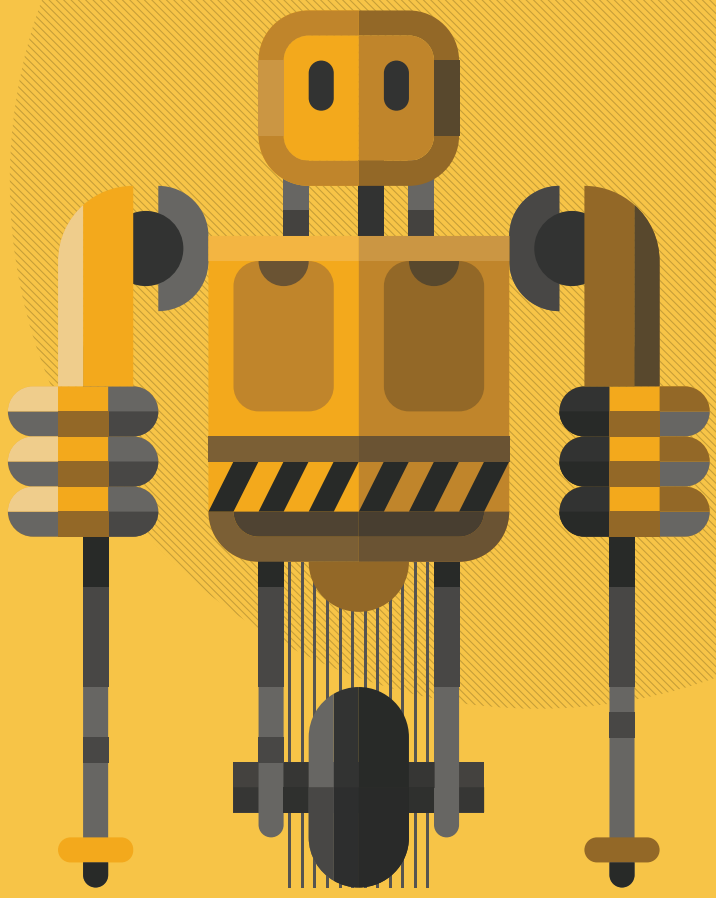


HealthTap, Fynd

Otras compañías de venta al por menor también experimentan con bots integrados a sus sistema de comercio en línea. Incluso existen tiendas de bots en linea como bots.kore.com que ofrecen algunos ya creados y la posibilidad de crear nuevos bots con un mínimo o nulo (chatfuel.com) conocimiento de programación y con la posibilidad de aprovechar la infraestructura que ya tienen en la nube.

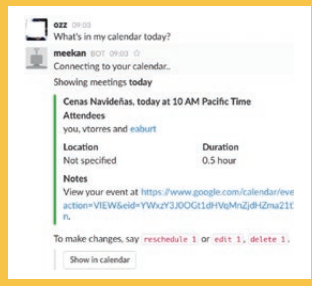
Al final, la tendencia es que los bots se integren más en las apps de uso diario con el fin de tener el control de ejecución y planeación de decenas de tareas desde un mismo punto de acceso. Para ello existen plataformas preparadas y en continuo proceso de mejora para facilitar el desarrollo de bots que integren otros servicios: Facebook Messenger, Kik, Slack, Hipchat, Skype, Telegram, por mencionar algunos. Los dos primeros son unas de las plataformas que ya tienen inclusive integrada un repositorio de bots para empezar a interactuar sin necesidad de programar nada.

No obstante muchos de los bots responden a una sintaxis específica para ejecutar comandos, al estilo línea de comando, usando banderas y variables que de indicarse en el orden incorrecto o por olvidar escribir una letra, causan error o ninguna respuesta. El reto inmediato es dotar al bot de la inteligencia necesaria para responder a malos comandos con sugerencias que ayuden al usuario a usarlo de forma correcta en un modo conversacional. Un desafío a futuro es lograr que esa conversación entre bot y usuario se dé sin la necesidad de ejecutar comandos con una sintaxis rígida, sino a través de conversaciones como entre humanos se da. Otro desafío está en la seguridad, considerando los múltiples usuarios interactuando con el bot, las diferentes integraciones



y accesos a servicios/datos sensibles junto con la capacidad del bot para aprender y ejecutar soluciones sin interacción humana. Evitemos casos como el del bot Tay creado por Microsoft que se hizo famoso porque en poco aprendió a enviar mensajes racistas por Twitter.

Por último, les comparto algunos enlaces para que puedan aprender más sobre varias de las plataformas mencionadas.
https://developers.facebook.com/products/messenger/
https://dev.kik.com
https://api.slack.com/bot-users
http://botlab.hipch.at/
https://www.skype.com/en/developer/
https://core.telegram.org/bots



Ozz

BIO. Oswaldo Herrera (@wdonet) es Ingeniero de Software en Nearsoft. Ha trabajado con tecnologías Java por más de 10 años, y es apasionado de las buenas prácticas en el desarrollo de tecnología, la automatización de procesos e integración de servicios para facilitar el trabajo de otros. Frecuentemente encontrarás a Oswaldo como mentor en eventos de aprendizaje para desarrolladores.

Cuatro Maneras en que la Inteligencia Artificial Cambiará Prácticamente Todo

Por Enrique Ortegón

La película Volver a Futuro predijo de manera acertada muchas cosas que el día de hoy son realidad. Y aunque nos faltan algunas, hoy podemos contar con autos que se conducen solos, robots, y bots que responden preguntas de manera humana, recomendaciones sobre cuando dejar un lugar para llegar a tiempo al otro tomando en cuenta el clima y el tráfico. Muchas de las marcas que utilizamos en nuestro día nos conocen perfectamente bien y nos dan recomendaciones proactivamente.

Todas esas tecnologías están impulsadas por la inteligencia artificial (IA): sistemas complejos que logran procesar y analizar todos los datos que nuestra sociedad genera y almacena. Estos sistemas son capaces de aprender, responder y predecir en base a la cantidad más grande de datos en la historia, y la cual solo seguirá creciendo.

No fue fácil llegar hasta aquí. Como lo comenta el artículo del New York Times "Artificial Intelligence Swarms Silicon Valley on Wings and Wheels" , el interés en la IA en los años 80 dio lugar a un pequeño boom. Sin embargo, debido a que carecía de un uso viable para los negocios, se produjo el "invierno de la IA". Hoy, la mayor cantidad de datos y el inmenso poder de procesamiento han hecho posible la IA —esta vez en serio.

Actualmente existen muchos usos para esta tecnología, pero creo que estamos solo rascando la superficie sobre el potencial impacto que puede tener en nuestras vidas. La combinación de la conectividad a internet y la propagación de dispositivos móviles resulta en miles de millones de aparatos conectados que se pueden beneficiar de la IA, al servicio de la humanidad.

Nuestra vida está a punto de cambiar de una manera muy rápida. Estas son cuatro maneras en que la IA lo cambiará todo:

1. EL INTERNET DE LAS COSAS

¿Has notado cómo las computadoras son cada vez más pequeñas y, a la vez, más inteligentes? También son más baratas. Hoy en día, hay una computadora dentro de cualquier cosa con un botón de encendido/apagado. Todos estos nuevos dispositivos inteligentes —desde tostadores hasta cepillos dentales, termostatos, focos y automóviles— se comunican entre sí, con las empresas y con los consumidores. ¿Por qué no habría tu auto de avisar a tu casa que estás por llegar para que tu casa le diga al horno que se precaliente a la temperatura adecuada para ese pescado que ya sabe que acabas de comprar desde tu teléfono? Esos miles de millones de cosas conectadas significan volúmenes gigantescos de datos de clientes. Las empresas necesitan ser inteligentes en cuanto a la manera en que recopilan, digieren y aplican esos datos.

2. DATOS Y ANÁLISIS INTELIGENTE

Se está produciendo una brecha enorme entre las compañías y los clientes. De todos los datos que los clientes crean, menos del 1% se analizan, de ahí que el 77% de los clientes afirmen que no sienten que las empresas con las que interactúan tienen una relación con ellos. Existen tantas maneras de leer los datos y tantas conclusiones a las cuales llegar sobre los hábitos y preferencias de los clientes, y, sin embargo, la mayoría de esas posibles perspectivas se quedan a medio camino porque las empresas no dan prioridad al análisis de esos datos. Las nuevas herramientas revelan perspectivas útiles sobre el cliente. Esas perspectivas existen a lo largo de un espectro de inteligencia: las herramientas más básicas requieren que extraigas información de ellas, mientras que las herramientas más inteligentes te entregan información y anticipan lo que quieres saber. Para éstas últimas utilizamos el aprendizaje automático.

3. APRENDIZAJE AUTOMÁTICO

Con el aprendizaje automático, los sistemas de cómputo pueden tomar todos esos datos del cliente y complementarlos, a fin de operar no sólo con lo que se ha programado, sino también para adaptarse a los cambios. Los algoritmos se adaptan a los datos, lo que les permite desarrollar conductas previamente no programadas. Aprender a leer y reconocer el contexto significa que un asistente digital podría escanear los mensajes de correo electrónico y extraer lo que necesitas saber. Inherente a ese aprendizaje es la capacidad de realizar predicciones sobre el comportamiento futuro, conocer al cliente de manera más íntima y ser no nada más proactivo, incluso prescriptivo.

4. PREDICCIÓN

Los datos masivos y el análisis producen patrones, y cuando las máquinas más inteligentes son capaces de leer patrones y aprender de ellos, pueden descifrar lo que sigue, deducir en lugar de asumir, y concluir en lugar de adivinar. La promesa de un "asistente digital" no es la voz robótica que responde tus preguntas sobre el clima y los horarios de cine, sino que sabe tus patrones, aprende de ellos y nos recuerda que salgamos en ese momento para romper nuestro récord de llegar dos minutos tarde el 80% del tiempo. El sistema necesita alimentarse, y de ahí partirán máquinas más inteligentes, dispositivos conectados y la capacidad de predecir nuestras necesidades y deseos. Entre más información de calidad proporcionemos al sistema, mayor será su inteligencia y capacidad de servirnos.

En todas esas áreas, la IA nos ayudará a trabajar de manera más eficiente, al igual que nos permite etiquetar a gente en una foto de Facebook. El "invierno de la IA" no volverá a repetirse, y las compañías de todos tamaños se beneficiarán de ella. Desde mi punto de vista, en el caso de la Inteligencia Artificial (IA) el invierno no está por llegar, sino que el invierno ya acabó.

Enrique Ortegón es director de la región Hispanoamérica y Caribe en Salesforce.

Primero el Internet de las Cosas ¿Y luego?

Por Amín Espinoza

Llevo ya un par de años hablando en todo foro que me es posible acerca del Internet de las Cosas (IoT) y he aprendido tres grandes lecciones.

La primera es que a todo el mundo le encanta hablar del tema, decir que es el futuro de procesos, que te permitirá ahorrar mucho al automatizar las cosas y todo eso. Pero la realidad es que pocas son las empresas que han demostrado un interés real y lo han llevado a la práctica. Una vez que lo han hecho se dieron cuenta de que el proceso es mucho más complejo de lo que se muestra en la teoría, pero al lograr implementar una solución adecuada sí hay un enorme margen de ganancia principalmente en tiempo y en incremento de productividad. De hecho, el mercado en este segmento en México es prácticamente nulo permitiendo una oportunidad fantástica oportunidad para abrir una nueva línea de negocio para muchas compañías.

La segunda lección que he aprendido es que debemos aprender a diferenciar conceptos. Es sumamente fácil diluir las palabras "maker", "domótica", "electrónica" e IoT. Cada disciplina tiene sus retos propios y un común denominador que te permitirá moverte entre todas ellas sin una gran complejidad. La diferencia radica en los fines, si tu intención es construir una máquina dedicada a una función exclusiva o si quieres ir más allá al recolectar información. En el caso del Internet de las Cosas hay dos grandes fines, la automatización de procesos y el monitoreo remoto.

La tercera y más importante lección es que el Internet de las cosas ha permitido la creación de un perfil híbrido de gran especialidad y con un conjunto de habilidades que permiten llegar al fin esperado. Cuando una persona desea especializarse en IoT debe aprender de electrónica, infraestructura y desarrollo de software. Estas tres áreas pueden implicar por ellas mismas una especialidad. Debes considerar esto para cualquier escenario que tengas en mente, primero cómo vas a hacer para unificarlas en un solo propósito para generar un perfil único, altamente especializado en este tipo de soluciones y que con cierta práctica puede implementar una mejora en virtualmente cada proceso de nuestras vidas, desde el simple hecho de despertar hasta la optimización de cada segundo de nuestro tiempo. Ya después extenderlo a un equipo único que te permita escalar tus proyectos a niveles mucho más altos.

Así que tomando como base estos tres grandes aprendizajes puedo sugerirte tener presente los siguientes datos:

- El Internet de las Cosas llegó para quedarse y eso es fácil de notar, no solo a nivel consumidores en donde ahora puedes medir tu ruta recorrida, tu peso, la calidad del funcionamiento de tu auto y muchas cosas más, sino que el mayor nicho de mercado es ver a otras compañías que apreciarán mucho las implementaciones que la tendencia puede dar para evitar pérdidas y acelerar el acceso a la información. La estimación apunta a que un 70% de los proyectos está orientado al modo de negocio "Bussiness to buiness".

- Para el 2020 se tiene estimado un mercado de 1 billón de dólares originados a partir de este tipo de soluciones entre retail y soluciones B2B.

- El internet de las cosas es solo la puerta de entrada para nuevas tendencias tecnológicas inspiradas en el enorme manejo de datos obtenidos, después de terminar con el proyecto IoT necesitarás agregar en tu equipo a un analista de Big Data que te permita aprovechar cada bit que tengas almacenado ¡Es sorprendente la cantidad de cosas que podrás controlar e incluso predecir gracias a la mezcla de IoT y Big Data!

Así que no te quedes con este artículo únicamente, comienza a buscar más del tema, nota su enorme crecimiento y popularización con muchas empresas y comienza a buscar ese perfil tan particular que te dará una ventaja descomunal a nivel profesional.

Amín Espinoza es Sr. Technical Evangelist en Microsoft México.

ESTUDIO DE SALARIOS SG 2016

Durante noviembre de 2016 realizamos la novena edición de la Encuesta de Salarios de Software Guru. Compartimos aquí los principales resultados.

FUENTE DE LOS DATOS Y CONTEXTO

Las cifras y estadísticas mostradas en el presente reportaje se generaron en base a una muestra de 1,623 respuestas obtenidas durante noviembre de 2016 en una encuesta abierta realizada por medio de internet.

Aunque la encuesta estaba abierta a participantes de cualquier país, los datos mostrados reflejan principalmente la situación en México, dado que 96% de los participantes en la encuesta indicaron radicar en este país.

Todos los datos se refieren a salario bruto mensual, expresado en pesos mexicanos.

En la mayoría de las tablas mostramos los siguientes datos: mediana, media (promedio), desviación estándar y tamaño de la muestra. Recuerda considerar todos estos datos antes de hacer tus propias conclusiones.

SALARIO MEDIO

De acuerdo a los datos recopilados este año, **el salario medio de un profesionista de software en México es de $29,140** pesos brutos mensuales. Ese dato se refiere a la mediana, que consideramos que es el valor que tiene más sentido.

	Mediana	Media	Des. Std.	Q1 (25%)	Q3 (75%)
Salario Mx (MXN)	$29,140	$32,087	$20,530	$18,000	$40,000

Tabla 1. Estadísticas descriptivas del salario en México

La tabla 1 muestra los principales datos estadísticos para el salario. Como podemos ver, está organizada por cuartiles. De acuerdo a esta tabla, el 25% de los profesionistas de software en México tiene un sueldo bruto mensual menor a 18 mil pesos, otro 25% tiene un sueldo mayor a 40 mil pesos, y el 50% restante gana entre 18 y 40 mil pesos. Comparando estos datos con los del año anterior, donde obtuvimos una mediana de $26,000 y media de $29,380, podemos entonces deducir que el salario promedio de los profesionistas de software en México aumentaron alrededor de un 10% durante 2016.

Estamos conscientes de que este dato es demasiado general, ya que la compensación en nuestra profesión es afectada por un gran número de variables. Así que no hay que tomarse este dato muy en serio. A continuación mostraremos los datos desglosados a través de distintas variables, que ayudarán a obtener mayor claridad al respecto.

La tabla 2 muestra estadísticas descriptivas del salario a través de distintos países. Solamente se incluyen aquellos países de donde obtuvimos al menos 5 resultados, lo cual es un número bastante bajo como para dar datos confiables, pero aún así es útil para brindar un panorama general de lo que sucede en cada país. Las cifras se refieren a salario bruto mensual en dólares. Como podemos ver, a través de distintos países de Latinoamérica el salario medio es relativamente consistente, rondando entre 1,250 y 1,450 dólares, mientras que en Estados Unidos esto es mucho mayor, y por ello que tantos desarrolladores de software de nuestra región han emigrado a este país en los últimos años.

País	Muestra	Mediana	Media	Des. Std.
Estados Unidos	9	$9,000	$8,452	$1,460
México	1558	$1,465	$1,607	$1,030
Colombia	17	$1,400	$1,440	$749
Chile	5	$1,400	$1,976	$1,206
Perú	6	$1,350	$1,250	$459
Argentina	6	$1,253	$1,263	$239

Tabla 2. Salario para profesionistas de software en distintos países.

ACTIVIDADES Y ROLES

Como sabemos, nuestra industria involucra una gran variedad de actividades y roles que realizan los profesionistas. La tabla 3 muestra algunos de los principales, y los salarios asociados a cada uno. Los participantes en la encuesta podían elegir hasta 3 roles que realizan, es por ello que la suma de los porcentajes es mayor a 100%.

Como ya es costumbre, los roles de preventa, venta y dirección perciben los salarios más altos. Vale la pena aclarar que dado el perfil de la audiencia que contesta esta encuesta, la mayoría de las personas que indican ser directivos se refiere a que son fundadores o socios en empresas pequeñas; no estamos hablando de altos directivos en corporativos, ya que ese perfil tiende a percibir compensaciones mayores a los 100 mil pesos mensuales y están fuera del alcance del presente estudio.

En los últimos años hemos notado en el estudio de salarios de SG como la proporción de participantes que indica dedicarse a la programación ha ido aumentando. El salario de estas personas también ha ido aumentando respecto a otros roles como los de consultoría y liderazgo. Hasta hace unos años, la programación era considerada una actividad de bajo valor y era claramente superada salarialmente por otras actividades más cercanas al negocio (ej. Implantación de ERP, business intelligence); pero conforme las organizaciones entran en iniciativas de transformación digital y buscan desarrollar productos digitales, la demanda y valor de los programadores ha aumentado.

Rol	Pct	Mediana	Media	Des. Std
Preventa	2%	$50,000	$63,502	$45,536
Venta	3%	$49,000	$54,902	$40,286
Dirección	10%	$46,350	$51,366	$29,713
Consultoría de negocio	8%	$38,000	$41,335	$25,164
Mejora de procesos	8%	$37,000	$40,182	$21,629
Arquitectura de sistemas	27%	$35,000	$36,854	$20,389
Project Management	21%	$35,000	$36,443	$16,935
Capacitación	5%	$31,000	$36,649	$25,478
Business Intelligence	5%	$30,000	$34,735	$22,131
Implantación de ERP	6%	$30,000	$30,067	$15,580
Programación Back-End	48%	$28,850	$30,576	$16,989
Programación Front-End	35%	$27,250	$28,747	$16,719
Análisis de requerimientos	27%	$27,000	$28,253	$14,681
Infraestructura	8%	$25,423	$29,049	$17,954
Aseguramiento de calidad	14%	$24,250	$26,340	$13,860
Seguridad informática	3%	$24,000	$28,990	$17,477
User experience design	5%	$22,000	$26,771	$15,849
Administración de datos	12%	$20,000	$23,555	$13,977
Soporte técnico	8%	$19,500	$22,291	$15,546
Docencia	3%	$17,125	$23,299	$17,875

Tabla 3. Salario por rol

CIUDADES

Tradicionalmente hemos agrupado los salarios de acuerdo a la entidad federativa (estado) donde vive el participante. Sin embargo, para mejorar el nivel de granularidad este año hemos decidido manejarlo a nivel de ciudad, no de estado. La tabla 4 muestra el salario medio de acuerdo a la ciudad donde radica la persona.

Como podrán ver, la novedad es que la ciudad de Colima aparece con el salario medio más alto. Aunque podemos argumentar que este es un fenómeno particular de la audiencia de SG, consideramos que vale la pena tomar nota de este caso. A diferencia de ciudades como Guadalajara donde ha habido un empuje orquestado por el gobierno, clusters y grandes empresas como Oracle, IBM e Intel, en Colima lo que se ha dado es muy distinto y mucho más orgánico. Siguiendo el ejemplo de la extinta Crowd Interactive (hoy Magma Labs), se han establecido varias boutiques que hacen desarrollo remoto para clientes norteamericanos. La filosofía de estas empresas es similar: tener equipos pequeños con personal de buen nivel, bien compensado y autodirigido, que requiere una capa mínima de gerentes.

Fuera del caso extraordinario de Colima, lo que encontramos es una continuación de lo que hemos visto en los últimos años: Guadalajara y México se mantienen como los principales polos, mientras que Monterrey parece seguir a la baja.

Estado	Muestra	Mediana	Media	Des. Std.
Colima	40	$40,000	$34,388	$17,251
Guadalajara	309	$35,000	$34,955	$15,428
Aguascalientes	16	$35,000	$33,656	$24,173
CDMX	571	$32,000	$36,605	$23,420
Hermosillo	70	$30,000	$32,466	$15,958
León	26	$27,500	$30,419	$13,713
Monterrey	127	$26,000	$30,949	$20,690
Estado de México	21	$26,000	$27,747	$15,005
Torreón	14	$24,750	$32,607	$22,407
Morelia	33	$24,000	$25,230	$16,436
Mérida	22	$23,500	$26,843	$18,063
Cuernavaca	11	$23,000	$37,827	$55,703
Chihuahua	17	$23,000	$24,994	$13,581
Ensenada	11	$22,900	$22,082	$8,813
Querétaro	62	$22,000	$24,797	$15,216
Puebla	19	$22,000	$24,311	$18,341
Cancún	11	$19,000	$25,396	$15,695
Veracruz	16	$18,917	$18,058	$9,053
Culiacán	27	$15,000	$19,085	$18,605

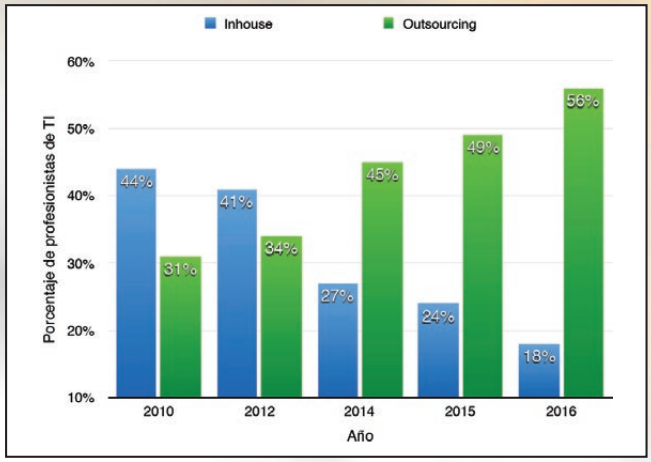
La tabla 5 muestra la descomposición de los participantes de acuerdo al tipo de organización en la que laboran. Los proveedores de servicios son típicamente empresas de outsourcing, los ISV son empresas que desarrollan software comercial, y las organizaciones usuarios son aquellas de otros sectores que no sean TI (ej. banca, retail, gobierno, etc).

La tendencia es la misma que hemos visto en los últimos años, con los startups y empresas de outsourcing ofreciendo los salarios más altos, aunque en este tipo de empresas (especialmente las proveedoras) rara vez se recibe el 100% de la compensación en un esquema de nómina. Más información al respecto en la tabla 6.

Tipo de empresa	Pct	Mediana	Media	Des. Std.
Startup	4.5%	$30,000	$33,262	$24,603
Proveedor de servicios TI	56.0%	$30,000	$33,132	$19,218
Org. usuaria	18.0%	$28,000	$32,231	$22,930
ISV	15.5%	$28,000	$31,999	$21,696
Capacitación	0.6%	$19,000	$23,120	$26,754
Institución educativa	3.7%	$18,000	$21,286	$13,925
Canal / distribuidor	1.7%	$16,000	$23,751	$18,585

Tabla 5. Salario por tipo de empresa

En ediciones anteriores hemos comentado sobre cómo está cambiando la proporción de personal de TI que labora en empresas de outsourcing en comparación de personas empleadas directamente por las organizaciones usuarias o clientes. La gráfica 1 muestra cómo se ha ido dando este cambio, y la tendencia es impresionante. Los datos son exclusivamente a partir de las encuestas de salarios de SG del 2010 al 2016 (en 2011 y 2013 no hicimos), y por lo tanto no podemos decir que son representativos de todos los profesionistas de TI de México, pero aún así muestran una tendencia muy clara.



Gráfica 1. Profesionistas de TI inhouse vs outsourcing

La tabla 6 muestra los distintos esquemas de contratación de los participantes en la encuesta. Aunque el esquema de nómina sigue siendo el más utilizado, el esquema híbrido (una parte en nómina y otra por honorarios) sigue aumentando su uso. De hecho, específicamente en las empresas proveedoras de servicios TI encontramos que el 47% de los colaboradores perciben su compensación bajo este esquema, comparado con un 46% que lo recibe por nómina al 100%.

Otro caso interesante es el de los freelancers. La diferencia de lo que puedes cobrar como freelancer trabajando para clientes nacionales versus internacionales es abismal. Obviamente, trabajar para clientes internacionales involucra tener dominio del inglés y normalmente involucra un mayor grado de experiencia y habilidad; pero como puedes ver, bien vale la pena prepararse para poder atender a este mercado.

Esquema	Pct	Mediana	Media	Des. Std.
Freelance internacional	0.5%	$67,500	$88,625	$72,687
Socio/accionista	2.7%	$45,000	$48,951	$34,891
Híbrido	31.8%	$32,000	$32,903	$15,127
Nómina	58.8%	$28,000	$31,549	$20,490
Honorarios	4.5%	$25,250	$26,260	$15,083
Freelance nacional	1.6%	$18,000	$21,813	$20,063

Tabla 6. Salario por esquema de contratación

GÉNERO

La tabla 7 muestra el salario medio entre hombres y mujeres. Desde la primera edición de esta encuesta hemos detectado una participación de alrededor de 15% de mujeres. Desgraciadamente, a pesar de los esfuerzos que se han estado realizando en años recientes a través de nuestra industria, esa cifra no ha cambiado significativamente.

Género	Pct	Mediana	Media	Des. Std.
Hombre	83.3%	$30,000	$33,764	$21,200
Mujer	16.7%	$21,000	$24,070	$14,959

Tabla 7. Representación y salario de hombres vs. mujeres

Más preocupante es que en nuestras cifras de este año, la brecha salarial entre hombres y mujeres es todavía mayor que de costumbre, teniendo una diferencia cercana al 40%. Realizamos un análisis adicional para detectar si esto cambiaba a través de distintos roles, pero la tendencia se mantiene; incluso acotándolo a actividades donde hay un porcentaje relativamente alto de mujeres tales como diseño UX, análisis de requerimientos y aseguramiento de calidad, el salario medio de los hombres es por lo menos un 25% mayor al de las mujeres.

HABILIDADES Y CONOCIMIENTOS

Las tablas 8a a 8e muestran el desglose de salarios a través de distintas categorías de herramientas y tecnologías utilizadas por los profesionistas de software.

Por último, en la tabla 8f tenemos el desglose a través de certificaciones. Sabemos que hay una gran variedad de certificaciones en nuestra industria, pero aquí hemos incluido a las más populares.

Plataforma	Muestra	Mediana	Media	Des. Std.
Mainframe	55	$35,000	$40,409	$27,571
iOS	255	$32,000	$36,693	$27,504
JVM	535	$32,000	$34,036	$20,885
Internet of Things	114	$30,000	$34,891	$25,043
Web	1045	$28,000	$30,960	$19,072
.NET	500	$28,000	$30,198	$17,670
Android	419	$25,000	$30,512	$21,886

Tabla 8a. Desglose por plataforma aplicativa.

Lenguaje	Muestra	Mediana	Media	Des. Std.
Elixir	13	$45,000	$42,923	$12,305
Go	12	$37,250	$33,902	$15,867
Groovy	16	$36,500	$39,250	$16,093
Bash	40	$35,500	$37,319	$19,567
R	4	$32,711	$48,855	$34,172
Objective-C	19	$32,000	$33,458	$12,753
ABAP	6	$30,500	$34,333	$13,140
Ruby	75	$30,000	$31,656	$16,559
Typescript	29	$30,000	$31,301	$10,105
Javascript	497	$30,000	$30,789	$17,786
Python	61	$30,000	$29,900	$15,823
Java	360	$29,537	$31,271	$17,863
Cobol	6	$29,500	$30,066	$14,004
Perl	4	$28,500	$37,575	$19,682
Scala	6	$28,500	$29,438	$12,421
PLSQL	212	$28,000	$29,827	$15,313
C#	263	$28,000	$29,535	$16,089
Swift	24	$26,500	$35,458	$38,265
VB	55	$25,900	$30,969	$18,748
C	49	$25,000	$27,982	$14,369
PHP	131	$20,000	$23,288	$14,602
Delphi	11	$17,898	$20,345	$12,223

Tabla 8b. Desglose por lenguaje de programación.

Tecnología Front-end	Muestra	Mediana	Media	Des. Std.
Ember	13	$40,000	$41,500	$16,086
React	49	$39,000	$36,041	$14,195
Nativescript	9	$35,000	$30,889	$16,632
Angular	272	$30,000	$31,230	$16,293
WPF	39	$29,200	$32,344	$14,454
Cocoa	14	$28,600	$29,657	$13,053
Unity	13	$28,600	$28,315	$14,327
Foundation	23	$28,000	$25,405	$12,089
JQuery	408	$26,500	$28,015	$14,762
Bootstrap	360	$26,000	$27,637	$15,971
MDL	5	$23,000	$26,660	$10,456
Ionic	21	$23,000	$25,324	$13,581
Cordova	21	$23,000	$25,138	$13,699
Polymer	9	$22,000	$27,367	$17,908
Sencha	7	$15,500	$24,071	$19,762

Tabla 8c. Desglose por tecnología front-end.

BD	Muestra	Mediana	Media	Des. Std.
Informix	9	$30,000	$29,103	$9,714
DB2	12	$27,500	$30,625	$12,324
Oracle	54	$27,000	$26,879	$13,735
MongoDB	18	$24,000	$28,972	$21,474
SQL Server	121	$21,000	$23,357	$12,223
PostgreSQL	38	$20,000	$24,179	$16,642
MySQL	115	$18,000	$21,991	$13,094
SQLite	18	$18,000	$24,044	$17,425
Access	25	$17,000	$19,848	$10,389

Tabla 8d. Desglose por manejador de base de datos.

Infraestructura	Muestra	Mediana	Media	Des. Std.
Openstack	8	$41,500	$40,431	$14,791
Heroku	12	$40,000	$42,125	$29,702
Docker	34	$35,295	$39,115	$21,604
AWS	53	$33,000	$35,398	$20,976
SAP	3	$31,000	$27,333	$10,017
CloudFoundry	2	$30,500	$30,500	$20,506
IBM	10	$30,000	$30,350	$11,629
JBoss	13	$29,074	$33,183	$15,300
Oracle	11	$27,810	$29,376	$14,485
Azure	20	$25,500	$32,200	$19,851
VMware	36	$21,000	$25,903	$15,332
IIS	32	$20,000	$24,835	$15,531

Tabla 8e. Desglose por plataforma de infraestructura.

Certificación	Muestra	Mediana	Media	Des. Std
Enterprise Architect	20	$50,500	$61,145	$36,182
Websphere	14	$41,500	$59,786	$50,780
PMP	78	$40,000	$43,565	$23,065
Microsoft Expert	79	$40,000	$42,294	$24,278
Business Analyst	32	$39,500	$43,784	$25,968
SAP	27	$39,000	$39,885	$18,893
Xamarin	13	$38,000	$30,856	$13,129
Scrum Master	161	$37,000	$39,491	$22,174
UML	27	$36,000	$36,833	$28,954
COBIT	19	$35,000	$46,395	$29,964
ITIL	93	$35,000	$39,563	$24,727
Seguridad (Ethical Hacker)	13	$35,000	$36,046	$19,836
Linux	33	$35,000	$34,162	$15,035
Java	201	$34,500	$34,707	$19,406
Microsoft Associate	85	$33,500	$35,402	$22,441
Six Sigma	28	$32,500	$35,846	$30,877
Oracle DBA	26	$32,500	$34,558	$13,250
Testing y SQA	61	$30,000	$32,982	$18,710
PSP Developer	34	$28,000	$26,442	$12,219
Oracle Applications	15	$25,000	$28,620	$15,731
PHP / Zend	12	$24,000	$28,824	$15,746
Android	50	$21,500	$22,071	$11,625
Cisco	60	$20,000	$25,095	$20,536

Tabla 8f. Desglose por certificación.

OTROS FACTORES

La tabla 9 muestra el desglose de salario de acuerdo al máximo nivel de estudios terminado. Aunque sí hay cierta influencia del nivel de estudios en el salario percibido, podemos ver que hay otros factores que tienen mayor impacto, tales como las certificaciones (8f), nivel de inglés (10) o años de experiencia (11).

Estudios	Pct	Mediana	Media	Des. Std.
Maestría	16.9%	$35,000	$41,627	$27,641
Posgrado	4.0%	$32,000	$34,507	$16,023
Universidad (titulado)	53.6%	$28,400	$30,793	$17,730
Doctorado	0.6%	$26,500	$35,100	$25,813
Técnica / Vocacional	1.3%	$26,000	$30,619	$25,743
Preparatoria	0.9%	$26,000	$24,933	$15,375
Universidad (sin titular)	22.5%	$25,000	$28,202	$19,069

Tabla 9. Desglose por nivel de estudios terminado.

Nivel de Inglés	Pct	Mediana	Media	Des. Std.
Nativo (ILR 5)	1.2%	$51,000	$76,694	$52,763
Avanzado (ILR 4)	17.6%	$39,600	$42,383	$24,198
Profesional (ILR 3)	40.8%	$32,000	$34,179	$19,000
Limitado (ILR 2)	28.9%	$23,000	$25,268	$13,668
Elemental (ILR 1)	10.4%	$20,000	$21,622	$12,756
Ninguno (ILR 0)	1.2%	$13,500	$21,511	$17,408

Tabla 10. Desglose por nivel de inglés.

Experiencia	Muestra	Mediana	Media	Des. Std.
Menos de 2 años	101	$10,000	$12,132	$9,034
2 a 4 años	359	$20,000	$21,432	$11,532
5 a 7 años	327	$30,000	$30,454	$15,443
8 a 10 años	280	$35,000	$35,937	$13,404
11 a 15 años	242	$37,250	$40,361	$22,778
16 a 20 años	143	$40,000	$45,917	$29,028
21 a 25 años	58	$37,500	$39,272	$18,755
26 a 30 años	29	$40,000	$52,674	$40,113
Más de 30 años	19	$38,000	$48,026	$26,600

Tabla 11. Desglose por años de experiencia.

¿Qué tendrá mayor impacto en el salario, la experiencia o el nivel de inglés? Esto lo podemos contestar con un análisis de correlación, y la tabla 12 muestra tal. Como podemos ver, el factor con mayor correlación son los años de experiencia, y de cerca le sigue el nivel de inglés. Un siguiente factor es la cantidad de personas a tu cargo. Por último, podemos ver que la antigüedad tiene un impacto prácticamente nulo en el salario. Así que afortunadamente, en nuestra profesión no dependemos de quedarnos en la misma empresa para mantener o mejorar nuestro nivel de compensación, nuestros conocimientos y experiencia hablan por sí solos.

Variable	Correlación con salario
Experiencia	0.42
Nivel de inglés	0.37
Numero de coordinados	0.27
Antigüedad	0.04

Tabla 12. Factores con mayor correlación

Antes de cerrar con el análisis de experiencia, hagamos un análisis "congelando" dos factores importantes: el nivel de inglés y la localidad. En este caso, la tabla 13 nos muestra un desglose de experiencia pero tomando en cuenta solamente a las personas que radican en la Ciudad de México y tienen un nivel de inglés profesional (ILR=3). Como podemos ver, los salarios en general son mayores a los de la tabla 11, debido al nivel de inglés y la localidad; sin embargo, la tendencia general es bastante similar a la que se da en la tabla 11.

COMENTARIO DE NUESTROS PATROCINADORES

Agradecemos el patrocinio de Unosquare (http://unosquare.com), que apoyó el fondeo para la realización del presente estudio. A continuación les compartimos un mensaje de Ignacio Miranda, COO en Unosquare:

Unosquare es una empresa que genera valor a través de su gente. Esta filosofía hace que nuestras metas como compañía estén alineadas a la búsqueda, retención y desarrollo de nuestro capital humano. Nuestra misión, lo que define el motivo de existencia de nuestra empresa y genera nuestra cultura laboral, ve hacia dentro, hacia nuestra gente. Existimos para lograr y mantener el bienestar y progreso de quienes trabajamos aquí. Por lo tanto, en Unosquare, nuestra misión es a 3 niveles:

- A nivel personal, generamos las condiciones para que nos guste venir a trabajar.

- A nivel profesional, tenemos pasión por lo que hacemos, somos autodidactas y somos emprendedores.

- A nivel comunidad, vivir bien si se puede y somos ejemplo de ello.

El participar y promover iniciativas como el Estudio de Salarios de Software Guru permite el recabar información y tendencias importantes en materia de sueldos y condiciones de empleo en diferentes ciudades del país. Por otro lado, guía al colaborador, ingeniero o futuro egresado de una Carrera relacionada con las tecnologías de información, a definir y diseñar su plan de carrera.

Experiencia	Muestra	Mediana	Media	Des. Std.
Menos de 2 años	5	$9,000	$8,600	$3,830
2 a 4 años	33	$25,000	$28,134	$11,403
5 a 7 años	44	$35,000	$34,120	$12,156
8 a 10 años	31	$40,000	$39,710	$10,817
11 a 15 años	30	$49,000	$53,238	$22,999
16 a 20 años	32	$45,000	$48,162	$20,378
21 a 25 años	15	$48,000	$46,926	$14,328
26 a 30 años	12	$40,000	$52,547	$29,854
Más de 30 años	6	$40,000	$54,500	$36,898

Tabla 13. Desglose por experiencia en Ciudad de México e inglés = 3.

Prestación	Pct.
Seguro de gastos médicos mayores	59%
Horario flexible	55%
Seguro de vida	44%
Trabajo remoto o desde casa	42%
Vales de despensa	29%
Préstamos / Caja de ahorro	29%
Bonos de desempeño	27%
Seguro de gastos médicos menores	27%
Servicio de comedor	17%
Telefono celular	15%
Ayuda para educación (ej. maestría)	10%
Acciones (equity)	6%
Gasolina	5%
Ayuda familiar (guardería, beca, etc)	3%
Automóvil	3%
Departamento / Vivienda	2%

Tabla 14. Prestaciones percibidas.

PRESTACIONES Y MOTIVACIÓN

La tabla 14 muestra las prestaciones que los participantes indicaron recibir. La tendencia para permitir trabajar remotamente sigue aumentando, lo cual es una gran ventaja en ciudades como la de México donde las personas pierden una cantidad significativa de tiempo en ir y venir de la oficina. Por otro lado, la cantidad de personas que tienen acceso a propiedad de la empresa (equity) es baja, pero parece que va aumentando.

Por otro lado, la tabla 15 muestra las principales motivaciones que indicaron los participantes para hacerlos cambiar de trabajo. Están agrupadas de acuerdo al rango de salarios percibido actualmente. Esto es porque sabemos que la principal prioridad es el salario, pero nos interesaba ver cómo van cambiando las prioridades de las personas conforme van obteniendo una mayor compensación salarial. Las columnas indican el salario actual (ej. Menos de 15 mil pesos, 15 a 30 mil pesos, etcétera).

Tabla 15. Prioridad de acuerdo al rango de sueldo actual.

Prioridad / Rango de salario	< 15k	15k a 30k	30k a 45k	45k a 60k	> 60k
Mejor sueldo	79%	81%	77%	70%	57%
Desarrollo profesional	57%	51%	46%	53%	42%
Mejores prestaciones	39%	43%	38%	38%	18%
Estabilidad	28%	23%	24%	13%	18%
Cambio de ciudad o país	14%	16%	20%	25%	24%
Equity	9%	9%	16%	21%	31%
Ambiente de trabajo	15%	12%	10%	8%	13%

Mejores Empresas para Trabajar 2016

En un mundo dirigido por software, contar con desarrolladores talentosos es una capacidad clave para las empresas. Ante esto, en los últimos años hemos atestiguado un boom de reclutadores especializados en TI. Las empresas están genuinamente interesadas en atraer al mejor talento tecnológico posible. En principio, esto se consigue ofreciendo una compensación económica atractiva, y para eso está el estudio de salarios de SG. Pero más allá del dinero, hay muchos otros factores que interesan a los profesionistas de software, y en base a los que evalúan una empresa donde colaborar.

Como ya es costumbre en estudio de salarios de Software Guru, cerramos con la lista de empresas mejor evaluadas por sus colaboradores. Esta lista se genera a partir de las respuestas de las personas que contestan la encuesta de salarios SG. Para la edición de cierre 2016 contamos con la participación de 1,623 personas, en su gran mayoría en México. Al contestar la encuesta, los participantes tienen la opción de indicar la empresa donde laboran, y evalúan su satisfacción laboral en cada uno de los siguientes rubros:

- *El trabajo que realizo es un buen reto intelectual.*
- *Trabajo con gente muy capaz, de la que aprendo.*
- *La empresa me apoya para capacitarme.*
- *La empresa ofrece facilidades para participar en eventos de la industria.*
- *Sé hacia donde voy con esta empresa, confío en que puedo crecer mucho en ella.*
- *Mi trabajo me ayuda a vincularme con personas fuera de mi empresa.*
- *Cuento con el equipo y herramientas (hardware y software) adecuado para realizar mi trabajo.*
- *La empresa tiene finanzas estables y paga puntualmente.*
- *Me agrada mi estación e inmobiliario de trabajo (oficina, escritorio, silla, etcétera).*
- *Estoy satisfecho(a) con la ubicación donde se encuentra la empresa/oficina.*
- *Me gustan las opciones que tengo para comer.*
- *Estoy satisfecho(a) con el horario de trabajo (o flexibilidad).*
- *Me siento valorado(a) por mis superiores.*
- *Laborar en esta empresa me facilita estar saludable.*

Con esto se obtiene una calificación total de parte de cada participante y luego se promedian las evaluaciones de participantes de la misma empresa para obtener la evaluación de cada empresa.

Cabe mencionar que previamente analizamos los datos para filtrar y descartar posibles casos de fraude. Adicionalmente, solamente tomamos en cuenta a empresas de donde obtuvimos respuestas de un mínimo de 5 participantes distintos. Así que si tu empresa no aparece, puede ser porque o no fue lo suficientemente bien evaluada, o no se recibieron suficientes evaluaciones de personal de esa empresa.

GANADORES

Presentamos la lista de las 20 mejores empresas para trabajar de acuerdo a los participantes en la encuesta de salarios SG 2016. La lista está ordenada de acuerdo al ranking obtenido.

1. Base22
2. Segundamano
3. Nearsoft
4. Michelada.io
5. Magmalabs
6. Tacit Knowledge
7. Unosquare
8. Tiempo Development
9. Globant
10. Oracle (Mexico Development Center)
11. IDS Comercial
12. DS Indigo
13. Scio Consulting
14. IBM
15. InnovaWeb
16. DW Software
17. Los Xamarinos
18. Extend Solutions
19. Softtek
20. Praxis

Reconocemos la labor de estas empresas para generar condiciones para que sus colaboradores estén satisfechos y puedan desarrollar su potencial al máximo.

Invitamos a todas las empresas que quieran aparecer en este ranking el próximo año a que se ocupen en mejorar la satisfacción de sus empleados y en noviembre de 2016 se aseguren de invitarlos a que contesten el estudio de salarios de SG.

El Caso de PHP y los Salarios

Por Basilio Briceño

Como programador uno se pregunta ¿en qué lenguaje debo invertir mi tiempo? es cierto que no existe el lenguaje perfecto y que la elección usualmente es basada en preferencias personales o en el tipo de problema a resolver, sin embargo, existe otro punto a tomar en cuenta: el financiero, o en términos simples, ¿en qué lenguaje debo especializarme para ser mejor pagado?

Y es donde nos preguntamos ¿por qué muchos empleadores pagan más a programadores especializados en ciertos lenguajes que en otros? ¿qué es más importante a ser tomado en cuenta a la hora de establecer tabuladores salariales: la experiencia del programador, lo complejo de la tarea a realizar, o el lenguaje de programación a usar? Si se contrata a un carpintero, no se le contrata debido a que utiliza martillos Stanley o destornilladores Craftsman, se le contrata porque sabe hacer bien su trabajo y da resultados. Si lo mismo aplícase a los programadores, entonces ¿por qué muchas empresas en México aún insisten en pagar tarifas basadas en el lenguaje de programación?

Y ¿qué tiene que ver PHP en lo anterior? pues resulta que es uno de los lenguajes con el que los programadores salen más afectados en términos financieros. PHP es un lenguaje interpretado fácil de aprender, flexible y se puede desarrollar con él en tiempos muy cortos, además es genial para utilizarse como pegamento entre aplicaciones. Siendo así, ¿por qué es prácticamente considerado inferior?

Técnicamente el lenguaje nació para desarrollo web, fue creciendo y adaptándose a las necesidades de los desarrolladores. Muchos desarrolladores expertos en otros lenguajes lo critican por haberse vuelto popular muy rápido sin tener una base tan sólida y reglas estrictas como en los lenguajes de uso general, solo que en su afán por establecer que PHP está mal olvidan no fue planeado como un lenguaje de uso general, sino para facilitar el desarrollo web.

Entonces ¿PHP es técnicamente inferior? No, el motivo por el que es criticado está principalmente basado en un círculo vicioso generado por los programadores novatos. ¿Cómo es esto?, cuando un novato generalmente no presta atención en la calidad de sus entregables, se enfoca únicamente en terminar lo que le piden sin añadir ningún valor agregado a su trabajo, ¿el resultado? programas en estilo código spaghetti, estructuras mal aplicadas, carencia de optimizacion, entradas de datos inseguras, y un sinfín de "características" más.

Lo anterior obviamente repercutirá en el desempeño del programa y en el costo de mantenerlo. Sin embargo, curiosamente el enfoque no es dirigido hacía el programador novato, sino sobre el lenguaje.

Así entonces mientras más novato el programador peor calidad de código, ante inferior calidad más desconfianza se genera en el lenguaje, al existir desconfianza en el lenguaje menor es la apuesta en términos salariales, a menores salarios los programadores con experiencia y enfoque en la calidad se interesan menos en el lenguaje, a menor cantidad de programadores con experiencia mayor contratación de programadores novatos, a mayor cantidad de programadores novatos peor calidad de código y de ese modo el ciclo va degradando el valor salarial sobre el uso del lenguaje.

Por si fuera poco, el programador novato llega a creer que lo que hace es todo lo que se puede hacer con PHP, incluso hay quienes colocan en su currículo el adjetivo "experto en PHP" o "nivel avanzado en PHP", y al considerarse a sí mismos expertos no se esfuerzan en aprender más, ni en mejorar la calidad de su trabajo. Sin embargo más allá de sólo exponer el problema ¿cómo se pudiera convertir el círculo vicioso en uno virtuoso?

Mientras mayor calidad y valor agregado ofrezca el programador, mejores resultados verá el empleador, a mejores resultados el empleador debe ofrecer un mejor salario, con mejores salarios en el mercado, más programadores con experiencia y enfoque en calidad se verán interesados, así como los programadores novatos en mejorar la calidad de su trabajo para llegar obtener dichos salarios, a mayor cantidad de programadores con experiencia, mayor calidad en el código y mayor valor agregado, y de ese modo el ciclo va enriqueciendo la calidad de los entregables y el valor salarial.

Un buen punto de arranque para comenzar a cambiar el ciclo es reconocer en donde se encuentra actualmente el programador y fijar metas para mejorar. Basado en lo anterior surgió en el grupo de programadores de "PHP México" una iniciativa para definir un tabulador sencillo basado en los conocimientos y capacidades de los programadores, lo presento a continuación.

- **Novato.** Generalmente escribe PHP, SQL y HTML/JS/CSS en el mismo archivo y acostumbra a copiar y pegar cualquier cosa que se encuentra en internet que parecer hacer lo que se le pidió entregar y que según dicen los foros es la solución.

- **Aprendiz.** Ha aceptado que necesita mejorar y decide aprender mejores prácticas y aplicarlas. Consulta en foros no para buscar codigo para copiar y pegar, sino para analizarlo y pregunta después de haber investigado por cuenta propia.

- **Junior.** Sabe qué son los patrones de diseño y los utiliza, entiende qué es MVC, REST, CRUD, ORM, SQL Injection, XSS, I/O Sanitization, etcétera, y usa estas y otras técnicas, así como mejores prácticas; es excelente aprendiendo y usando APIs de terceros.

• **Senior.** Es capaz de desarrollar sus propias herramientas y APIs; es consciente de cómo funciona PHP internamente, está sumamente interesado en el desempeño de las aplicaciones más allá de su facilidad de desarrollo; no está satisfecho con el funcionamiento de todas las herramientas y APIs de terceros tal cual vienen y siempre busca innovar y crear cosas que mejoren el desempeño, faciliten su trabajo y la velocidad de sus entregables; contribuye con código en diferentes proyectos y comunidades; entiende la frase "PHP es el frontend de su backend".

• **Master.** Es capaz de desarrollar sus propias extensiones para PHP en C, contribuye a PECL y normalmente se dedica a mejorar el desempeño de aplicaciones de alto rendimiento. Conoce bien el lenguaje Hack, su origen y diferencias respecto a PHP.

• **Core.** Contribuye al desarrollo del core de PHP o el motor de Zend; contribuye en algunas extensiones y herramientas públicas; es un ponente frecuente en conferencias internacionales y es usual ver su nombre en PECL y en los créditos en cambios de versión de PHP.

Respecto de los salarios, estos pueden variar dependiendo la zona, los siguientes son sugeridos con base a los costos de vida de las principales ciudades de México en Diciembre de 2016:

- Novato - hasta 10 mil pesos.
- Aprendiz - de 10 a 18 mil pesos.
- Junior - de 18 a 30 mil pesos.
- Senior - de 30 a 50 mil pesos.
- Master - de 50 mil en adelante.

Ahora bien, no todo recae en el programador. El empleador debe ser consciente de este tabulador y aplicarlo o mejorarlo con base a los resultados de sus programadores, así como participar en la implementación de medidas que contribuyan a la mejora continua de las habilidades de sus programadores. De hacerse así es muy probable que se logren mejorar la calidad del software desarrollado con PHP y que se incrementen los beneficios mutuos.

Basilio ha contribuido por más de 15 años con diversas organizaciones como desarrollador, líder técnico, administrador de sistemas, líder de operaciones web, Jefe de Departamento, entre otras responsabilidades. Desde hace 4 años colabora con empresas de venta al detalle en Estados Unidos como GAP, Kohls, LVMH, entre otras automatizando procesos para reducir el tiempo de liberación al mercado de soluciones de comercio electrónico bajo un esquema DevOps utilizando diversas herramientas de integración y entrega continuas.

Cómo Estructurar el Contenido de tu Aplicación

Por Misael León

Sabemos que la media de las aplicaciones en el mercado pierde el 77% de sus usuarios después de tres días de la instalación. Dentro de 90 días sube a 95% [1]. Si tu aplicación tiene un alto índice de abandono es probable que la razón sea porque los usuarios:

1. No desarrollaron el hábito de usarla.
2. No encontraron lo que buscaban y la aplicación no es fácil de usar.
3. Nunca compraron algo o no vieron el valor de uso.
4. Nunca recibieron un recordatorio para volver a utilizarla.
5. No fueron dirigidos a ella desde la versión móvil de tu sitio.
6. Tuvieron problemas de desempeño con la app.
7. Recibieron demasiadas notificaciones y se hartaron.
8. Intuyeron problemas de seguridad con sus datos.
9. Experimentaron procesos de compra confusos.
10. Cambiaron de teléfono.

En esta ocasión abordaremos el problema de cuando los usuarios no encuentran lo que andan buscando dentro de la aplicación.

DEFINE EL PROBLEMA QUE EL PRODUCTO RESUELVE

Para que una aplicación provea valor real a las personas, es necesario primero descubrir cuál es ese "trabajo" que esperan realizar en nuestra aplicación. Esto comienza con la definición del producto que estamos construyendo. Debemos pensar en los problemas que estamos resolviendo antes de pensar en funcionalidades.

Para descubrir ese problema central puedes contestar las siguientes preguntas:

1. ¿Qué necesidad estás intentando resolver?
2. ¿Qué hacen las personas actualmente para satisfacer esta necesidad?
3. ¿Cuánto tiempo pasan satisfaciendo esta necesidad?
4. ¿Están buscando actualmente una solución para satisfacer esta necesidad?
5. ¿Tienen requisitos o necesidades especiales para adoptar el producto?
6. ¿Qué comportamientos / necesidades / objetivos predicen su uso?
7. ¿Cómo satisface este producto la necesidad?
8. ¿Cómo hace este producto el usuario mejor?
9. ¿Por qué seguirán utilizando en el transcurso de los próximos años?
10. ¿Cómo podemos medir cuando el producto ha satisfecho la necesidad?

Es probable que necesites realizar una serie de entrevistas con usuarios potenciales para validar y modificar tus supuestos. Lo importante es que conozcas los matices del problema que tu aplicación busca resolver. Una vez que lo tengas claro viene la tarea más compleja: idear una solución.

No estamos hablando aquí de un prototipo como tal, sino de la definición del producto. *"Vamos a hacer una aplicación que... encuentre promociones en restaurantes cercanos, que lleve el récord de gastos, que muestre recomendaciones sobre lugares para comprar ropa, etc."*

Piensa que la experiencia del usuario es definida por este binomio problema-solución. Las funcionalidades concretas del sistema sirven a esa estrategia central de UX pero no pueden reemplazarla. La idea de agregar features sin razón es darse un tiro en el pie. *"El Diseño de Interacción y Diseño Visual puede hacer que un producto sea hermoso, fácil de usar, encantador o que se destaque de la competencia, pero no puede hacer que el producto sea significativo para las personas." [2]*

EL MUNDO ESTÁ HECHO DE OBJETOS

Una vez definida la solución es momento de plasmar esa idea en un flujo funcional. Tal vez tu primera tarea sea la de prototipar algunas pantallas que reflejen ese *"vamos a hacer una app que..."*

Existe una metodología llamada *Object-Oriented UX* (OOUX) la cual consiste en estructurar el contenido de un sistema de manera que refleje objetos del mundo real. OOUX pone en el centro del proceso el modelo mental de los usuarios por encima de las acciones del sistema (búsqueda, filtrar, comparar, *sign-in, check-out, etc*) [3]

Pensemos el ejemplo de una aplicación para encontrar instrumentos musicales. Un usuario pensando en adquirir una guitarra eléctrica probablemente asocie conceptos como guitarras, cuerdas, afinadores, bocinas, tienda, órdenes, catálogos, accesorios, etc. Esto es porque el mundo está hecho de objetos que manipulamos y a los que aplicamos acciones como buscar, seleccionar, comparar, comprar, enviar, etc.

Lo anterior es cierto tanto para el mundo tangible como para el mundo virtual. En *Instagram* manipulamos fotografías, usuarios, comentarios, likes, seguidores. En *Uber* sucede lo mismo pero con autos, choferes, tarifas, destinos, etc. En eso consiste el enfoque orientado a objetos. OOUX es parte de la Arquitectura de Información y es una herramienta poderosa para crear el puente entre diseñadores y desarrolladores. Al momento de crear los componentes en el *back-end*, estos siguen siendo objetos e instancias a los que se aplican comportamientos específicos.

ESTRUCTURA LA ARQUITECTURA DE LA APLICACIÓN

A continuación describo un ejercicio para aplicar OOUX. Los únicos materiales que necesitas son un montón de post-it de cinco colores diferentes, marcadores negros y una pared o mesa amplia. El ideal es hacer esto con el equipo a cargo del producto.

Misael León (@misaello) es un UX/Product Designer que trabaja en Nearsoft investigando usuarios, desarrollando ideas de productos y diseñando prototipos. Su misión es la de crear herramientas intuitivas para que otros puedan realizar su trabajo. Le apasiona difundir las mejores prácticas de UX en las comunidades de desarrollo.

1. Encuentra los objetos del sistema
Comienza por tener a la mano la definición del producto que se va a construir. Subraya los sustantivos contenidos en los requerimientos de producto. Estos son generalmente los objetos que el usuario manipulará dentro de la aplicación. Ejemplo: "Una aplicación que permita a **músicos** encontrar y comprar **instrumentos musicale**s y con promociones exclusivas, en **tiendas** cercanas y con **reseñas** hechas por expertos"

Los sustantivos son: músicos, instrumentos musicales, promociones, tiendas, reseñas y expertos. Esos son los objetos del sistema, eso es lo que los usuarios llegarán buscando en la aplicación. Escribe cada uno de esos objetos en las notas azules y colócalas en fila lo largo de la pared.

2. Define el contenido esencial de cada objeto
Básicamente es identificar las características que componen el objeto dentro del sistema. Por ejemplo el nombre del producto, precio, la fotografía, descripción, etc. Escribe cada atributo en un nota amarilla y colócalas debajo de la notas azul correspondiente.

3. Añade la información secundaria del objeto (metadata)
Esta es la información que describe al objeto pero que si no estuviese presente no afectaría la percepción de los usuarios. Por ejemplo, el rating de popularidad otorgada por otros usuarios o tal vez el año de fabricación. Piensa que estos atributos son medios para filtrar y ordenar objetos en el sistema. Escribe cada metadata en una nota rosada y colócalas en la columna correspondiente.

La diferencia entre contenido esencial y *metadata* varía enormemente dependiendo del producto/servicio, de los juicios de los integrantes del equipo y de la audiencia a la cual nos dirigimos.

4. Encuentra la relación entre los objetos
Este paso consiste en ver cuál nota azul se relaciona con cuál. Por ejemplo el objeto "Reseña" se relaciona con el objeto "Experto" porque la primera es escrita por el segundo. También el objeto "Promoción" es asociado con el objeto instrumento y con el objeto "Tienda." Vuelve a escribir cada objeto en una nota azul y colócalas en la columna correspondiente.

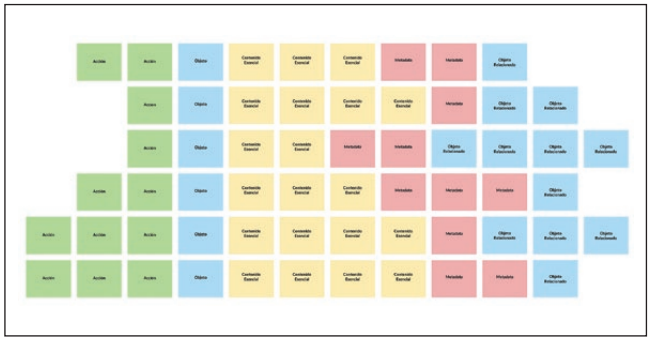


Figura 1. Mapa de sticky notes.

5. Decide las acciones viables para cada objeto
A cada objeto le corresponde una serie de acciones que los usuarios aplicarán a cada uno. Por ejemplo, una guitarra puede ser buscada, calificada, reseñada, agregada a carrito, etc. Las acciones están delimitadas por los objetivos de negocio, por las limitaciones técnicas y por las expectativas de los usuarios. Escribe cada acción en una nota verde y colócala por encima de cada objeto en nota azul en la fila principal que formamos.

El resultado final del ejercicio de OOUX es un mapa exhaustivo de tu aplicación como el que se ve en la figura 1. El mapa reflejará el modelo mental de las personas que utilizarán la aplicación. [4]

OOUX: ORDENANDO EL CAOS

El mapa de OOUX es una herramienta poderosa para lograr una visión de lo que se va a construir. El siguiente paso es hacer bosquejos de pantallas que reflejen esta estructura. Un punto fuerte de esta metodología es que permite desarrollar un enfoque de diseño modular. Es decir, crear plantillas de pantallas para mostrar diferentes instancias del mismo objeto.

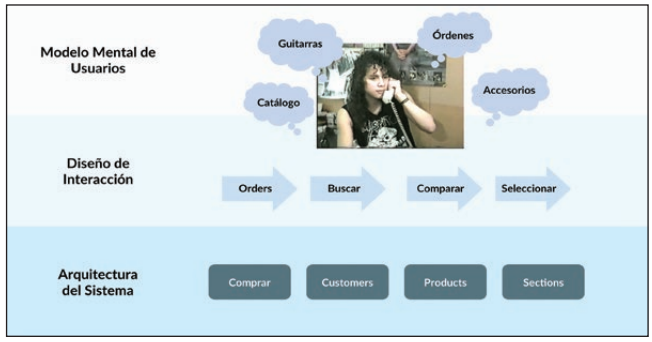


Figura 2. Mapa modelo mental/diseño de interacción/arquitectura de sistema.

A medida que tu producto evolucione irás añadiendo y eliminando objetos y elementos. Piensa que el mapa es una orientación para alinear el entendimiento del equipo sobre el producto que se construye.

El mundo digital no es todavía tan diferente al mundo real en el que interactuamos todos los días. Cada aplicación que construimos debe reflejar la manera en que las personas piensan. Si el contenido del sistema está ordenado de forma natural, aumentamos las probabilidades que los usuarios encuentren fácilmente lo que llegaron buscando.

Referencias
[1] A. Chen. "New data shows losing 80% of mobile users is normal, and why the best apps do better." http://swgu.ru/rj
[2] N. Blaase. "Why Product Thinking is the next big thing in UX Design". http://swgu.ru/ru
[3] S. Voychehovski. "Object-Oriented UX". A List Apart, Oct. 2015. http://swgu.ru/rv
[4] UX Team. How to Structure Your App's Content to Lower Abandonment Rate. Nearsoft, Inc. http://swgu.ru/rw

Special Purpose Languages

PARTE 5. LA LINGÜÍSTICA COMPUTACIONAL

—

Por Luis Vinicio León Carrillo

Luis Vinicio León Carrillo es Director General y co-fundador de e-Quallity. Fue profesor-investigador en la universidad ITESO. Realizó estudios de posgrado en Alemania, durante los cuales abordó temas relacionados con la prueba de software y los métodos y lenguajes formales.

En números anteriores hemos venido estudiando los lenguajes formales para tener herramientas que nos permitan hacer más eficiente el desarrollo de software. Permítanme en este número hacer una digresión para abordar brevemente el tema principal de este número, que considero podría serles de utilidad en el futuro próximo; prometo no entrar en detalle más allá de lo estrictamente necesario.

El tema "Lenguaje" es muy profundo y de muy largo alcance (y, debo confesar, es una de mis más profundas pasiones). "Lenguaje" está fuertemente vinculado con "Aprendizaje" e "Inteligencia" (por cierto, durante la Segunda Guerra Mundial se utilizaron exitosamente exámenes de lenguaje para seleccionar posibles oficiales entre soldados rasos). Hay quien ha llegado a escribir "... el universo es lenguaje, y el lenguaje es universo.", una frase de profundo alcance.

En un número anterior comentamos que sólo hasta que aprendió sobre funciones recursivas, el científico C.A.R. Hoare pudo expresar su método de ordenamiento llamado "QuickSort". Eso evidencía que los constructos de un lenguaje de computación pueden limitarnos para expresar lo que tenemos en mente. Pero, ¿acaso no ocurre lo mismo con nuestras habilidades lingüísticas, iniciando con nuestra lengua materna?

A lo largo del tiempo se han enunciado diversas "promesas" sobre las capacidades de una computadora para procesar lenguaje natural. Algunas capacidades sencillas ya pueden verse hoy en día comercialmente: hay asistentes personales que nos pueden ayudar a realizar reservaciones en un Hotel, o "chatbots" para dar soporte. Otras capacidades más sofisticadas han quedado pendientes. Quizás estemos cerca de ver algunas de ellas comercialmente. Hace algunas semanas asistí a un evento de IBM relacionado con "computación cognitiva", donde pude ver cómo un robot humanoide con varios grados de libertad obedecía instrucciones en Inglés a través de una conexión con Watson. También vi un producto para dar soporte de primer nivel, para el que, luego de recolectar 3,000 preguntas, se diseñaron 300 como "intended questions", que se utilizan como llaves primarias en una base de datos para encontrar las respuestas correspondientes; si el usuario comenta que no recibió la respuesta adecuada, entonces pasa a comunicarse con un humano; se mencionó una "tasa de éxito" es de entre 30% y 70%.

Pero, hablando de aplicaciones que implican una interacción en lenguaje natural muy sofisiticada, ¿a qué nos referimos cuando decimos que una computadora "entiende" el Español o a que un dispositivo sea "inteligente"?

Como una primera aproximación para responder esas preguntas, ya en 1950 Alan Turing sugirió reemplazar la pregunta "¿Pueden las máquinas pensar?" por "¿Puede una máquina tener un buen desempeño en el 'Juego de la Imitación'?", que consistía en que un humano interrogaba de manera remota a un hombre y a una mujer que se encontraban en otro cuarto para determinar cuál respuesta había sido dada por el hombre y cuál por la mujer.

Si las computadoras pudieran entender nuestros idiomas tan bien como lo hace una persona, podrían comprender muchísimo de lo que hay escrito en internet, así que podríamos hablar con ellas sobre una inmensa cantidad de temas, y sería difícil distinguir si estamos "hablando" con una máquina o con un ser humano muy cultivado. El que las computadoras entiendan el lenguaje natural puede tener muchísimas aplicaciones, no solo relativamente "sencillas" como las que mencionamos, sino otras de mucho mayor impacto. En los sistemas e-Learning, por ejemplo, podríamos tener "instructores" que jamás se desesperarían y que podrían enseñar a muchísimas personas, democratizando el conocimiento, y generando más oportunidades para todos; especialmente para los principiantes.

Tenemos 50 años diseñando lenguajes de programación para comunicarnos con las computadoras. Cabe preguntarse: ¿por qué no desarrollar también lenguajes naturales para comunicarnos con ellas? Ya hay antecedentes de ese trabajo; echémosle un ojo.

LENGUAJES "NATURALES ARTIFICIALES"

Llamamos "lenguajes naturales" a aquellos que son o fueron utilizados por una comunidad de seres humanos para comunicarse (como el Español, Inglés, Latín, Arameo, etc.). Dentro de los lenguajes naturales, hay algunos que fueron desarrollados artificialmente, a estos los llamamos "lenguajes planificados".

Se considera que la historia de los Lenguajes Planificados comienza en el siglo XVII, con los trabajos de Descartes y de Leibniz. Estos tipos de lenguajes parten de una clasificación de conceptos y construyen sistemas interconectados ordenados alrededor de conceptos fundamentales. Después de esos trabajos se desarrollaron otros lenguajes planificados, en relativamente diferentes momentos y países, y partiendo de principios distintos:

- Los llamados lenguajes "a priori", "esquemáticos" o "meta-lenguajes", que no incorporan características/construcciones de lenguajes naturales (al menos no intencionalmente), y que tienen una estructura altamente esquematizada, regular y lógica. Este tipo de lenguajes fueron dominantes durante varios siglos, pero en las primeras décadas del siglo pasado sufrieron un rápido declive.

- Los llamados lenguajes "a posteriori" o "naturalistas" que decididamente toman características de los lenguajes naturales (por ejemplo, la irregularidad en declinaciones y conjugaciones), y que renuncian a esquemas y clasificaciones dadas a priori. Una clase particular de estos lenguajes son los llamados "pseudo-lenguajes" que mezclan elementos de lenguajes naturales como raíces y declinaciones con elementos desarrollados artificialmente.

- Los lenguajes naturales simplificados ("sean" vivos como el Español, el Ruso, Chino o "muertos" como el Latín y el Arameo), en los que los mecanismos de "construcción" del lenguaje en cuestión (conjugaciones, declinaciones, constitución de frases, etc.) son simplificados sistemáticamente (por ejemplo, haciendo regulares verbos que son originalmente irregulares).

Dos de los lenguajes planificados más relevantes son el Volapük y el Esperanto. A continuación comento más sobre ellos.

El lenguaje Volapük fue desarrollado en Alemania por Johann Martin Schleyer, publicado en 1879, pensado para que lo usara la élite educada. Fue el primer lenguaje planificado en pasar de la teoría (sin una comunidad que usara el lenguaje) a la práctica (una comunidad amplia que lo usara y enriqueciera). Luego de 10 años de su publicación, ya existían 283 Organizaciones de "Volapukistas", 25 diarios en ese idioma, libros de enseñanza traducidos a 25 lenguajes; así como una Academia de la Lengua, que por cierto no era muy abierta a aceptar reformas al lenguaje, lo cual, aunado a que el autor veía al lenguaje como algo de su propiedad, originó un gran debilitamiento del "Volapukismo".

Esperanto significa "un esperanzado" o "alguien que espera", lo que da idea inicial sobre su autor, quien realizó las primeras publicaciones sobre el lenguaje bajo el seudónimo de "Dr. Esperanto". Su verdadero nombre era Lazarus Ludvig Zamenhof, de origen judío y nacido en 1859 en Bialystok (hoy Polonia), en donde confluían rusos, polacos, alemanes y judíos. Zamenhof hablaba Ruso en casa y Polaco en la calle; en la preparatoria se distinguió en Alemán y Francés, así como en Latín y Griego; con su padre profundizó en el Hebreo. Estudió medicina en Varsovia, y luego se especializó en Oftalmología. Era un humanista idealista, que, viviendo en Europa antes de la Primera Guerra Mundial, quería con su "Lenguaje Universal" unir a la humanidad. A diferencia de Schleyer con Volapük, Zamenhof no se apropió del Esperanto. Esperaba que el lenguaje contribuyera a una democratización del saber y la cultura, y a conectar las masas a nivel internacional sin necesidad de la élite educada. Sabía que su "Lenguaje Universal" podía generar un movimiento social, y realizó también trabajos para desarrollar una "Religión Universal".

Zamenhof diseñó Esperanto buscando que: fuera tan fácil de asimilar, que cualquiera pudiera aprenderlo jugando; pudiera ser usado rápidamente por un aprendiz, gracias a su estructura simple y lógica; fuera usado por las masas. Estos objetivos fueron alcanzados gracias a que en Esperanto: el vocabulario fue tomado de varios lenguajes naturales vivos; la derivación y la flexión son regulares y estructuradas; a cada letra le corresponde un solo sonido, y el vocalismo se limita a las 5 vocales; las raíces de las palabras se tomaron en un 75% de las lenguas romances –especialmente el Latín (del que fueron escogidas por su internacionalidad, de manera que un ruso entendiera inmediatamente un 40%) y del Francés–, 20% de las lenguas germánicas, y el 5% restante del Griego, Hebreo, el Árabe, el Japonés, y las lenguas Eslavas; se buscó una pronunciación con máxima internacionalidad.

Todo lo anterior originó que el Esperanto ganara mucha aceptación: la primera de las muchas publicaciones sobre el lenguaje fue "Linguo Internatia" en 1887; en 1905, en Francia, 668 personas de 20 países participaron en el primer Congreso Mundial; hasta antes de 1914 se realizaron unos 8 congresos. Sin embargo, en ese desarrollo se cruzaron las dos grandes guerras, y todas las consecuencias de las mismas debilitaron mucho ese movimiento.

Estos son algunos de lenguajes planificados creados a lo largo de la historia:

- Lenguajes a priori o meta-lenguajes: Beobi (1920, Rusia), Suma (1943, EEUU), Unilingua (1965, Francia).
- Lenguajes "a posteriori" o "naturalistas": Semi-Latin (1910, Austria), Romanid (1956, Hungría).
- Lenguajes "a posteriori" "pseudo-lenguajes": Volapük (1879, Alemania), Esperanto (1887, Polonia), Dilpok (1898, Francia), Novesperanto (1910, Suiza), Intal (1956, Alemania), Malfalsito (1963, Francia).
- Lenguajes naturales simplificados: Tutonish (1902, EEUU), Simplo (1911, Italia), Anglic (1930, Suecia), Basic (1935, Gran Bretaña).

LINGÜÍSTICA COMPUTACIONAL

Aún cuando la idea de crear un lenguaje planificado que tenga la estructura adecuada como para hacer práctica la comunicación de un humano con un sistema de cómputo puede parecer buena y viable en principio, eso no es lo que finalmente se busca; entre otras cosas porque los humanos tendríamos que aprender un nuevo lenguaje natural. El objetivo es que las computadoras puedan entender nuestras propias lenguas. Eso nos empuja a ver las cosas desde una perspectiva más amplia: la lingüística computacional. La lingüística computacional es un campo interdisciplinario que comprende secciones de lingüística tradicional y teórica, lexicografía, psicología del lenguaje, filosofía analítica y lógica, procesamiento de texto, bases de datos, así como procesamiento de lenguaje hablado y escrito.

La computación numérica ha sido el enfoque predominante desde el nacimiento de las computadoras. En él, la comprensión del lenguaje natural ha estado restringida fundamentalmente al usuario: cuando uno quiere encontrar fragmentos relevantes en un texto largo relacionados con "problemas con autos rojos durante el periodo entre 1990 y 2000", la palabra "rojos" es tratada sólo como un signo por la computadora, no con el significado que tiene para nosotros; si la computadora entendiera no solo el significado de "rojos", sino de la frase completa, nos mostraría también textos que incluyeran las frases "década de los 90s", y "automóviles/coches tintos", entre otras. Técnicamente dicho: en los sistema de cómputo actuales, el query anterior tiene una alta precisión (un alto porcentaje de los textos encontrados son relevantes) pero bajo *recall* (los textos encontrados representan un bajo porcental de los posibles).

La lingüística computacional se basa en la llamada "Nouvelle Artificial Intelligece", que no hace el típico procesamiento simbólico de la inteligencia artificial tradicional usando computadoras numéricas, sino que se enfoca al procesamiento de significado usando "Agentes Autónomos". El objetivo de la lingüística computacional es desarrollar componentes tecnológicos, que llamaremos "agentes cognitivos autónomos" (robots), con los que el humano pueda comunicarse libremente en lenguaje natural. Esto requiere del modelado de procedimientos y estados cognitivos, para lo cual es necesario tanto un modelo del hablante como del oyente.

Para efectos del desarrollo de un agente cognitivo autónomo, uno podría considerar como "prototipo básico de comunicación" al discurso que se presenta entre dos partes, cara a cara, hablando de objetos concretos, y en su ambiente inmediato. Los componentes de comunicación básico podrían ser:

- La especificación del medioambiente de la tarea a realizar (también llamado "contexto").
- La estructura del agente cognitivo, que representa su "cognición"; los datos que recibe (por ejemplo de sensores), que constituyen su "percepción"; y la clasificación de los mismos que se considera su "reconocimiento".
- La especificación del lenguaje en el cual se lleva a cabo la comunicación.

La especificación del lenguaje debería considerar los siguientes elementos:

- Fonética: para facilitar la automatización de los procesos de articulación, acústicos y auditivos del habla.
- Léxico: en el cual tengamos por ejemplo, para cada concepto, sus hipónimos e hiperónimos, que nos permitan entender mejor las entradas (*input*) recibidas por el agente para generar salidas (*output*) de mejor calidad.
- Morfología: que nos facilite el análisis de las palabras de entrada y obtener sus constituyentes para eventualmente expandirlas antes de generar la salida.
- Sintaxis: que nos permita detectar si las estructuras en la entrada son correctas y generar las equivalentes en la salida.
- Semántica: que nos permita por ejemplo entender frases con negaciones y detectar frases que tienen significados equivalentes.
- Pragmática: que nos permita entender el significado de frases genéricas como "aspectos revisados en la última junta" en contextos particulares.

Para ilustrar algunos de los elementos anteriores tomemos el caso de la traducción automática. A lo largo del tiempo se han desarrollado diferentes enfoques:

- Traducción directa (50's - 60's): utilizando diccionarios, prácticamente se traducía palabra por palabra (o cuando mucho, utilizando su morfología), evadiendo el análisis del significado, pero buscando preservar los significados y proporcionar traducciones sintácticamente aceptables.

- El enfoque "de Transferencia", en el que se realiza un análisis léxico, sintáctico y morfológico del texto y se construye una representación del mismo; luego, utilizado esa representación y un diccionario que incluya aspectos sintácticos y morfológicos, se hace la traducción. Es un mejor enfoque que el anterior, pero aún no se entiende el significado del input, lo que generará problemas cuando un verbo en el lenguaje fuente ("to be" en Inglés) tiene dos traducciones en el lenguaje destino ("ser" y "estar" en Español), o un pronombre en el lenguaje fuente ("they" en Inglés) tiene dos traducciones en el lenguaje destino ("ellas" y "ellos" en Español).

- El enfoque 'Interlingua', en el cual, además del análisis léxico, sintáctico y morfológico, se realizan también el semántico y el pragmático, y se construye una "representación interlingüística" (por ejemplo, en Esperanto) rica en elementos para permitir generar una traducción de alta calidad.

Se han desarrollado también soluciones parciales como los "Restricted Languages", que son lenguajes naturales simplificados altamente esquematizados y enfocados a un dominio específico, permitiendo con ello realizar traducciones de muy alta calidad.

EL ASPECTO FORMAL

Aún cuando en nuestra primera columna mencionamos un enfoque sobre lenguajes formales llamado Sistemas Lindenmayer, dado nuestro interés en encontrar mecanismos para eficientar el desarrollo de software en números posteriores estudiamos los lenguajes formales utilizando principalmente los trabajos sobre gramáticas del lingüista Noam Chomsky, las cuales se conocen como Gramáticas Estructuradas por Frase, y en un número reciente presentamos una jerarquía de las mismas. Se presume que, en esa jerarquía, los lenguajes naturales se encuentran en la clase de los Lenguajes Sensibles al Contexto. En realidad, las Gramáticas Estructuradas por Frase son un tipo particular de las llamadas Gramáticas Generativas, que son sistemas ("recursivos") de reglas provenientes del trabajo en la Lógica Matemática.

Algunos de los enfoques más relevantes que han sido utilizados para abordar las Gramáticas Generativas son:

- Gramáticas Categóricas ("C-Grammars"): desarrolladas inicialmente en 1929 por el polaco Lesniesvsky. Aplicadas al los natural lenguajes por primera vez por Bar-Hillel en 1953.

- Gramáticas Estructuradas por Frase (PS-Grammars): basadas en los "Rewrite Systems" de Emil Post desarrollados en 1936, que fueron aplicadas a los lenguajes naturales por primera vez por Chomsky en 1957.

- Gramáticas Asociativas-por-la-Izquierda (Left-Assotiative-Grammars): desarrolladas en 1985 por Roland Hausser especialmente para procesar lenguajes naturales.

Los dos primeros enfoques presentan el problema de que los recursos requeridos (tiempo y memoria) para hacer el procesamiento de lenguajes naturales son demasiado grandes como para llevarlos a la práctica. El tercero muestra algunas bondades al respecto.

Formalmente, una Gramática-LA (del Inglés Left-Associative Grammar, LAG) es una séptupla de la forma:

$$\langle W, C, LX, CO, RP, S_I, S_F \rangle \quad \text{donde}$$

W es un conjunto finito de llamadas Word surfaces;
C es un conjunto finito de Category segments;
L $\supset$ ($W \times C^+$) es un conjunto finito conteniendo el léxico;
CO = ($co_1, \ldots, co_{n-1}$) es una secuencia finita de funciones recursivas totales (entiéndase "programas") de ($C^* \times C^+$) a ($C^* \cup \{\perp\}$) llamadas Categorial Operations;
RP = ($rp_1, \ldots, rp_{n-1}$) es una secuencia finita (igualmente larga que CO) de subconjuntos de $\underline{n}$ (donde $\underline{n} = \{ i \mid 0 \leq i < n \}$) llamada Rule Packages;
S_I = { (c_{s1} rp_{s1}) ... (c_{sk} rp_{sk}) } es un conjunto finito de eStados Iniciales, donde cada $c \in C^+$ y cada rp es un subconjunto de $\underline{n}$ llamado paquete de reglas de inicio;
S_F = { (c_{f1} rp_{f1}) ... (c_{fk} rp_{fk}) } es un conjunto finito de eStados Finales , donde cada $c \in C^*$ y cada $rp \in RP$.

Así como con las Gramáticas Estructuradas por Frase de Chomsky que estudiamos, en el conjunto de lenguajes generados por Gramáticas-LA también tenemos subclases: las Gramáticas-LA generales, o A-LAGs, corresponden a la definición de arriba; las Gramáticas-LA limitadas (*bounded*) linealmente, o B-LAGs , son la subclase de las A-LAGs en las que la longitud de las categorías está limitada linealmente con respecto a la longitud de la cadena de entrada; las Gramáticas-LA limitadas por una Constante, o C-LAGs, son la subclase de las B-LAGs en las que la cantidad de cálculos requerida por operaciones categóricas individuales está limitada por una constante. Dentro de estas C-LAGs tenemos las subclases de C3-LAGs , C2-LAGs , y C1-LAGs .

A continuación describo la relación de las LAGs con la Jerarquía de Chomsky (incluyo algunos aspectos de rendimiento, que eventualmente abordaremos en números posteriores).

- Las A-LAGs aceptan y generan todos los lenguajes recursivos (o "Decidibles"); algo relevante, pues en la Jerarquía de Chomsky no hay un tipo de Gramática que definiera este tipo de lenguajes.

- Las B-LAGs aceptan y generan todos los lenguajes sensibles al contexto (LSC).

- Las C-LAGs, constan de 3 subclases:
  - C3-LAGs contiene los Lenguajes Libres de Contexto (LLC) más complejos y algunos LSC que son "NP-Complete" (la clase de problemas para los cuales no se han descubierto algoritmos eficientes que los resuelvan, y se piensan que no existen). Para procesar estos lenguajes se requiere tanto tiempo que los vuelve prohibitivos en la práctica (técnicamente requieren "tiempo polinomial", lo que significa que pueden procesarse en un tiempo expresado como un polinomio de grado ≥ 2 en función de la cantidad de datos de entrada);
  - C2-LAGs contiene varios LLC "no-deterministas" (su procesamiento implica backtracking) y varios LSC simples. Para procesarlos también se requiere tiempo polinomial;
  - C1-LAGs contiene todos los LLC "deterministas" (no requieren backtracking para procesarlos) y varios LSC simples. Su procesamiento sólo requiere tiempo lineal, lo que los vuelve muy útiles en la práctica.

Se piensa que en esta jerarquía, los lenguajes naturales están dentro de la clase C1-LAG, lo cual significaría que su procesamiento puede realizarse de manera práctica, aún con altos volúmenes de input.

Ojalá y tengamos pronto una C1-LAG para el Español, que nos permita desarrollar verdaderos sistemas cognitivos y entendernos con las computadoras en nuestra lengua materna!

¿Ven ahora por qué decimos que la Lingüística Computacional es de largo alcance? :-)

Sistemas Biométricos
Un panorama general

Por Juan Manuel Reyes

Cada vez es más común solicitar información biométrica para realizar trámites. En este artículo explicaremos por qué nos solicitan nuestra biometría y cómo se utiliza dentro de los sistemas.

Definamos primero qué es una biometría: una biometría se define como una característica física o conductual intrínseca que permite la identidad única de un humano. Ejemplos de una biometría son: las huellas dactilares, el iris, los rasgos faciales del rostro, la venas de las manos, la palma de la mano, la firma y la voz.

Veamos algunos casos reales de porqué se tienen que incluir las biometrías dentro de algunos trámites. Pongamos primero el ejemplo de la credencial de elector para votar en México. Anteriormente el Instituto confiaba plenamente en el ciudadano que decía la verdad al llevar sus documento probatorios para obtener su credencial para votar. Después de un tiempo, se empezó a ver que una misma persona obtenía varias credenciales utilizando documentos probatorios apócrifos que le permiten obtenerla derivado que no era posible validar los documentos con la institución que los emitió. Se pierde la confianza en el ciudadano y es necesario tener otro mecanismo en donde se pueda identificar que el ciudadano dice ser quien es.

Otro ejemplo es el sistema de pensiones mexicano que está estrenando sistema biométrico; hace tiempo algunos de los empleados de las AFOREs, quienes se encargan de atraer a los trabajadores para tener su ahorro hacia su AFORE, utilizaban la documentación del trabajador que le permitía cambiarlo de AFORE sin su consentimiento. La forma de resolver este problema es que si el trabajador va a realizar algún trámite como retiro de ahorros, lo haga de forma confiable utilizando sus biometrías para identificar que es la persona correcta y el empleado de la AFORE tendrá que identificarse con su biometría permitiendo realizar ese trámite para el trabajador.

¿Ahora bien, cómo se utilizan las biometrías dentro los sistemas? Comencemos por definir sistema de enrolamiento. Un sistema de enrolamiento es un sistema completo se que se encarga de capturar las biometrías, validarlas, transformarlas y guardarlas para verificaciones posteriores. Este sistema puede variar su arquitectura pero de forma general tiene los siguientes elementos:

- Dispositivos de captura.
- Formato de la biometría.
- Sistema de captura.
- Formato de envío.
- Middleware.
- Motor biométrico.
- Sistema de adjudicación.

A continuación los describo.

DISPOSITIVOS DE CAPTURA

Como su nombre lo indica, los dispositivos de captura son aparatos electrónicos que pueden capturar alguna de las biometrías de una persona. Por ejemplo, la mayoría conocemos los escaners para huellas digitales. Existe una gran diversidad de dispositivos para atender distintas necesidades y presupuestos. Existen dispositivos para tomar iris, huellas, rasgos faciales y cada uno de ellos captura la biometría y la convierte en un determinado formato binario. Típicamente estos dispositivos exponen APIs que nos permite consumir la información capturada en nuestros sistemas.

FORMATO DE LA BIOMETRÍA

Como ya mencionamos, los dispositivo capturan la biometría y la convierten en un formato binario. Por ejemplo, el formato de las huellas más usado es el WSQ (Wavelet Scalar Quantization) que fue desarrollado por el FBI [1] y el NIST (National Institute of Standards and Technology).

SISTEMA DE CAPTURA

Es una aplicación de software que permite a un operador registrar la información biométrica capturada. Este elemento es de los más importantes dentro del sistema de enrolamiento ya que en esta aplicación definimos reglas de validación que podemos aplicar a las biometrías. Esto es necesario porque si queremos que nuestro sistema sea confiable, entonces las biometrías registradas deben tener la calidad suficiente para poder distinguirlas de las de otra persona. La única oportunidad de tomarle las biometrías a una persona es cuando la tienes presente, entonces el operador debe aprovechar al máximo que tiene a la persona frente a él y con la ayuda del sistema de captura debe tratar de tomar lo mejor posible la biometría.

Como ejemplo de esto veamos el caso de la validación de las huellas dactilares, que es una de las biometrías más utilizadas al dia de hoy en diversos sistemas biométricos. Existen diversos modelos de captores de huella que pueden capturar desde una huella, dos huellas o hasta capturar 4 huellas de una sola toma. Todos estos dispositivos tienen la cualidad que están estandarizados con algo que se denomina FAP (*Fingerprint Acquisition Profile*), que son perfiles de adquisición de huellas. Estos perfiles, determinan diferentes características de la toma de la huella como son: resolución de la imagen, área de captura del dispositivo, el tipo de dispositivo, algoritmo de compresión, nivel de compresión, etc. La tabla 1 muestra valores para distintos niveles de FAP.

Característica	FAP 10	FAP 20	FAP 30	FAP 40	FAP 45	FAP 50	FAP 60
Resolución de la imagen	500 PPP ± 10 PPP (±2%)	500 PPP ± 10 PPP (±2%)	500 PPP ± 10 PPP (±2%)	500 PPP ± 10 PPP (±2%)	500 PPP ± 5 PPP (±1%)	500 PPP ± 5 PPP (±1%)	500 PPP ± 5 PPP (±1%)
Área mínima de captura (pulgadas)	0.5 de ancho por 0.65 de alto	0.6 de ancho por 0.8 de alto	0.8 de ancho por 1.0 de alto	1.6 de ancho por 1.5 de alto	1.6 de ancho por 1.5 de alto	3.2 de ancho por 2.0 de alto	3.2 de ancho por 3 de alto
Tipo de dispositivo (referencia)	Unidactilar	Unidactilar	Unidactilar	Bidactilar	Bidactilar	Bidactilar Decadactilar	Decadactilar
Algoritmo de compresión	WSQ 2.0 o superior	WSQ 2.0 o superior	WSQ 2.0 o superior	WSQ 2.0 o superior	WSQ 2.0 o superior	WSQ 3.1 o superior	WSQ 3.1 o superior
Nivel de compresión	10:1	10:1	10:1	15:1	15:1	15:1	15:1
Certificación del dispositivo	PIV	PIV	PIV	PIV	Apéndice F	Apéndice F	Apéndice F

Tabla 1. Valores para distintos valores de FAP.

Si queremos que nuestro sistema de enrolamiento biométrico de huellas sea los mas fiable posible, la recomendación de la mayoría de los proveedores de motores biométricos, es que se utilice en dispositivos el estándar del apéndice F (el cual es el estándar FBI IAFIS IQS CJIS-RS-0010: V7). Si checamos en la tabla los dispositivos con FAP 45 o superior son los que cumplen con este estándar. Como breviario cultural, el INE utiliza dispositivos decadactilares (4-4-2) que cumplen con este estándar y toman las diez huellas en 3 pasos tomando primero 4 huellas de la mano derecha, 4 dedos de la mano derecha y al ultimo los dos pulgares.

Entonces con el simple hecho de utilizar un dispositivo de buena calidad, ya podremos tener una buena toma de una huella, pero aun no es suficiente. Hay más parámetros que podemos mover en los dispositivos para que la huella contenga más elementos biométricos, tales como las minucias o la calidad de la huella (NFIQ) que nos permitan asegurar más la unicidad de las huellas. En este punto todo parece que con comprar buenos dispositivos ya todo nuestro sistema de enrolamiento va a funcionar, ¡qué gran error!. Por ahi supe de un proyecto de renombre que tenían super captores de biometrías, pero no tenían la idea de como hacer un sistema de enrolamiento biométrico.

En este orden de ideas, el sistema de captura es el encargado de interactuar con los dispositivos modificando los diversos parámetros para obtener la mejor toma posible de la biometría. Estos sistemas de captura pueden ser desde aplicaciones de escritorio, hasta aplicaciones móviles.

Para finalizar el sistema de captura, hay una recomendación más que es muy, pero muy importante. Puedes tener un buen dispositivo y un buen software de captura, pero nada eso funciona sin la capacitación del operador. Es importante capacitar al operador con las mejores prácticas para que aprenda a tomar las biometrías, ya que en ocasiones las personas a las que se les tomará la biometría no saben como poner, por ejemplo, los dedos en un captor de huella. Una mala toma de la biometría produce que se tenga que hacer regresar a la persona y eso en ocasiones es muy costoso.

FORMATO DE ENVÍO

Cuando se captura la biometría, los datos biográficos y de negocio en un sistema de enrolamiento es muy común que se tengan que enviar a algún sistema remoto para su procesamiento. Esto es, porque el verdadero procesamiento de la biometría se encuentra en un lugar central o distribuido; dependiendo la arquitectura derivada que se tiene que validar / cotejar con las millones de biometrías que se tienen capturadas de las demás personas. Para el envío de los datos, es necesario tener un formato que nos permita enviar las biometrías con sus datos intrínsecos y además los datos de negocio. Si el requerimiento de nuestro sistema biométrico no necesita la capacidad de comunicarse con algún otro sistema biométrico, nosotros bien podemos crear nuestro propio formato de envío que puede ir desde XML a JSON. Pero en el caso que tengamos que seguir los estándares de los sistemas biométricos o queramos interactuar con otros sistemas, es necesario ajustarse al formato ANSI/NIST-ITL 1-2011: Update 2015 que está regido por el NIST. Este formato es en realidad un XML que tiene todo el soporte para el envío de cualquier biometría con sus datos intrínsecos y es lo suficientemente flexible para que podamos enviar información propia de negocio.

MIDDLEWARE

Este elemento es en realidad un sistema que se encarga de procesar las peticiones de enrolamiento que envía el sistema de captura; sus tareas son almacenar la información, validar las reglas de negocio y llevar la auditoría de todo lo que pasa con la petición de enrolamiento para determinar si este último procede o no y enviar la respuesta a quien corresponda. Este sistema interactúa de forma directa con el motor biométrico de tal forma que, lo que determine el motor biométrico determinará si procede o no el enrolamiento.

MOTOR BIOMÉTRICO

Este elemento es la piedra angular de nuestro sistema de enrolamiento y es un sistema que se encarga de validar y convertir las biometrías para poder procesarlas y poder realizar operaciones sobre ellas. Si nuestro requerimiento es que debemos tener personas no repetidas, y las biometrías serán nuestro elemento para identificarlas, entonces están de acuerdo en que tenemos que tener una base de datos con la cual podamos comparar cuando llegue una biometría nueva para que no se inserte una repetida o evitar que alguien haga fraude.

Existen diversos proveedores de motores biométricos, los cuales tienen sus algoritmos secretos para tratar alguna de las biometrías y darnos la seguridad que el resultado de la búsqueda sea lo más certero y que nuestra base de biometrías esté libre de duplicados. Entre las empresas líderes en este segmento están NEC, 3M, Morpho y Neurotechnology.

Juan Manuel Reyes Medina es actualmente CIO de SynergyJ, empresa dedicada a la capacitación de tecnologías JVM. Es desarrollador de software desde hace 18 años, emprendedor desde hace algunos años e instructor de tecnologías y frameworks de JVM. Ha participado en proyectos en el sector privado y gobierno y dirigió dos proyectos exitosos de integración de sistemas biométricos que se encuentran actualmente en operación.

Cada una tiene ventajas y desventajas con respecto al software que utiliza y la infraestructura que requiere pero al final digamos que hacen lo mismo; aquí mucho depende de la inversión que se quiera hacer y los requerimientos, ya que no son nada baratos y el precio depende mucho de qué tan rápido se requiere que responda y la cantidad de biometrías sobre las cuales tiene que trabajar.

El INE tiene aproximadamente unas 80 millones de personas con sus huellas (10 huellas) y su captura facial. Anteriormente era una de las bases biométricas más grandes del mundo, aunque ahora India nos ha superado.

Estos motores biométricos están basados fuertemente en infraestructura ya que, por ejemplo para hacer una comparación con 50 millones de huellas se requieren unos 30 servidores con cientos de GB de memoria RAM para que la búsqueda responda en menos de 3 minutos. Por su parte, 3M tiene una propuesta interesante en donde reduce dramáticamente el número de servidores con unas tarjetas especiales que tienen el propósito de realizar las comparaciones en un solo servidor. Los motores típicamente convierten la biometría a un formato propietario (algo que le denominan template) el cual construyen a partir de las características intrínsecas de la biometría, por ejemplo, en el caso de las huellas se toman las minucias como base, suben a memoria los templates distribuidos en los diferentes servidores y con un software realizan las comparaciones de forma paralela en todos los servidores. Es importante mencionar que las comparaciones se basan mucho en estadística de qué tanto se parecen las biometrías y existen parámetros que se pueden calibrar para tener un equilibrio entre lo que se rechaza y lo que se acepta.

Los dos parámetros más importantes son el FAR (False Acceptance Rate) y el FRR (False Rejection Rate). El FAR mide la posibilidad de que el sistema acepte por error un acceso por un usuario no autorizado. En contraparte, el FRR mide la posibilidad de que un sistema rechace por error un acceso por un usuario autorizado.

Estos dos parámetros definen el umbral con el cual se rechazará o aceptará la biometría dependiendo de la similitud. Calibrar estos parámetros es una de las actividades más importantes y complicadas al implementar sistemas biométricos. Debe haber un equilibrio entre ambas ya que si le das más peso a una podrías estar rechazando los enrolamientos, derivado a que es muy alto el valor de precisión o en su defecto podrías estar aceptando enrolamientos duplicados (ver figura 1).

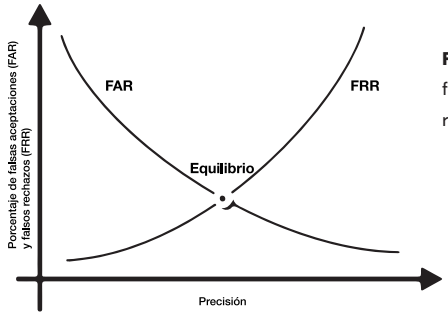


Figura 1. Equilibrio de falsas aceptaciones y rechazos.

Afortunadamente para nosotros, los proveedores de los servicios nos proponen los valores para estos parámetros, pero obviamente son referencia ya que esos valores son tomados de proyectos en los que han participado y no necesariamente son proyectos nacionales, entonces al final siempre tenemos que tropicalizarlos y hacerles el ajuste (tuning) correspondientes. Este ajuste es muy difícil de hacer cuando vas a empezar a enrolar desde cero y no tienes huellas con que probar, ya que para eso requieres un universo de huellas considerable (100,000 aproximadamente) que te permita probar y determinar una tasa baja de rechazos. Es uno de los problemas con los que te encuentras porque, ¿de donde sacas 100,000 huellas para probar?. Existen bases de datos de huellas públicas pero solo son 5,000 entonces solo puede probar con ese universo.

Estos motores biométricos por lo general exponen servicios que son independientes de la tecnología y la infraestructura para poder integrarlos con el middleware. Se pueden integrar a través de web services de tipo SOAP y REST o alguna API propietario en algún lenguaje específico.

Ahora bien, para poder enrolar a una persona con sus biometrías es necesario que le asignemos un identificador de negocio que nos permita saber desde nuestro ámbito de negocio a quién pertenecen esas biometrías. Una vez definido este identificador podemos pasar a explicar los servicios que exponen los motores biométricos. Cada proveedor varía el conjunto de servicios que expone, pero estos son los más comunes:

- **Servicio de enrolamiento.** Se encarga de insertar las biometrías en la base de datos. Primero hace una búsqueda sobre todas las biometrías existentes, si no hay coincidencias las inserta y las asocia con el identificador de negocio. En caso de que encuentre coincidencias rechaza la inserción y regresa los identificadores de las personas con las cuales hizo coincidencia.
- **Servicio de búsqueda.** Hace una búsqueda sobre todas las biometrías existentes y regresa los identificadores de las personas con las cuales hizo coincidencia.
- **Servicio de inserción.** Inserta las biometrías dentro de la base de datos sin ninguna validación.
- **Servicio de verificación.** Se encarga de verificar que las biometrías de una persona coincidan con las biometrías que se enrolaron previamente. En este servicio se hace otra toma de las biometrías de una persona y se compara contra las huellas que se enrolaron previamente; el servicio nos regresa solamente un MATCH o NO MATCH que nos dice si la persona es realmente o no la que dice ser.
- **Servicio de borrado:** Se encarga de eliminar una biometría de la base de datos a partir del identificador de negocio.

La respuesta de estos servicios puede ser síncrona o asíncrona. Los servicios de enrolamiento y búsqueda son los que más frecuentemente son asíncronos por el tiempo que demoran. Este es un punto importante porque debemos considerarlo cuando se hace la integración con el middleware.

Ya tenemos la descripción completa de nuestro sistema de enrolamiento, ahora veamos cómo se utiliza en la realidad. El primer paso dentro del flujo del negocio de las empresas o entidades gubernamentales es generar una base de datos de identidades, en donde tengamos a una persona asociada a una identidad de negocio con cada una de sus biometrías, esto con la finalidad de poder identificarla en un segundo momento con una nueva toma de su biometría. Por ejemplo, cuando vamos a solicitar una nueva credencial de elector o hacemos un movimiento de nuestra Afore nos pedirán nuestras biometrías para realizar un enrolamiento de las mismas.

El sistema de enrolamiento será capaz de encontrar si alguna persona intenta volver a enrolarse con alguna identidad apócrifa apoyado por la búsqueda en la base de datos biométrica. En cuanto se tenga un enrolamiento válido será posible identificar a una persona por sus biometrías en un segundo paso. Por ejemplo, si un trabajador quiere hacer algún trámite con su afore solo tiene que poner alguno de sus dedos para poder verificar su identidad, esto lo hace el servicio de verificación del motor biométrico que se describió anteriormente, como ya se tiene el identificador de negocio del trabajador y la toma de la huella, se comparara con las huellas que se tienen de su enrolamiento previo y el motor biométrico determinará si es la persona que dice ser.

Hasta este punto hemos descrito el camino feliz de un enrolamiento, ahora veamos las diferentes situaciones de casos excepcionales que son comunes en un sistema de identificación biométrica.

¿Qué pasa si las personas que vamos a enrolar no tienen las biometrías con las cuales se está enrolando? Por ejemplo, si se están enrolando las huellas y la persona no tiene manos, ¿qué se hace?. O si las huellas de la persona están tan desgastadas que no pueden ser leídas por el lector de huellas. O si una persona tiene lesionada las dos manos y no pueden ser leídas tampoco. Este tipo de situaciones se deben de considerar dentro de nuestro sistema desde el principio para ver cómo se deben manejar. En el caso de la persona que tiene las manos lesionadas, algun dia le tendrán que sanar, entonces se deberá de considerar un servicio de actualización de huellas que permita actualizar las de esta persona cuando esté sana para que se pueda identificar con su huella posteriormente.

Como ya había comentado la comparación biométrica no es del todo exacta, se basa mucho en estadística, es altamente probable que cuando una persona envía sus biometrías y se busquen en toda la base se encuentren coincidencias con otras personas que no necesariamente son la misma persona. A esto se le denomina falso positivo y se incrementa fuertemente cuanto más grande es la base de datos biométrica.

SISTEMA DE ADJUDICACIÓN

¿Cómo determinamos si esas coincidencias son un intento de fraude o realmente si es otra persona con las biometrías muy parecidas?. Bueno, aquí es donde entra en acción el último elemento del sistema de enrolamiento: el sistema de adjudicación. Este se encarga de resolver casos donde un operador compara la información de la persona que se quiere enrolar contra la información de las personas que se encontraron como coincidencias.

Aquí mucho depende de qué información se tiene, pero anticipándonos a esta situación se recomienda evaluar desde un principio si es necesario incluir distintos datos biométricos en nuestro sistema de enrolamiento, con la finalidad de tener más elementos que permitan al operador tomar una decisión. Esta decisión afecta significativamente el presupuesto del proyecto, ya que por cada tipo de información biométrica necesitamos adquirir e integrar los dispositivos al sistema. Además, se necesita tener un motor biométrico por cada tipo de biometría para poder obtener los resultados numéricos de similitud de cada una y unificar los resultados para tomar un decisión.

Un sistema de adjudicación puede ser tan simple o tan complejo dependiendo de la implicación legal que tiene su resultado. Podemos tener un sistema que solo muestre las huellas que se capturaron y que un perito determine si las huellas son las mismas o no. O podemos tener por ejemplo el caso de la credencial de elector en donde el INE toma las huellas, la foto y la firma; entonces su sistema de adjudicación puede tener hasta 3 biometrías para que un operador pueda tener la decisión de adjudicación.

Una vez que se tiene la respuesta del sistema de adjudicación, si es un intento de fraude o si es una persona diferente, es necesario que el sistema de enrolamiento tenga un flujo alterno para estos casos y como la primera vez que lo intentó enrolar no fue posible, debe hacerlo de otro modo. Recordemos que la primera vez usó el servicio de enrol que hace la búsqueda en toda la base biométrica, luego entonces no puede usar este servicio, en su lugar usa el servicio de inserción que introduce las biometrías sin realizar la búsqueda.

CONCLUSIÓN

El presente artículo ha buscado brindar un panorama general sobre los principales componentes y aspectos a considerar en el desarrollo de un sistema de información para gestionar información biométrica. Esperamos les pueda aportar valor a la hora de implementar alguno.

Referencias

[1] FBI Biometrics Specifications. https://www.fbibiospecs.cjis.gov

Inseguridades de IoT

Por Gunnar Wolf

Gunnar Wolf es administrador de sistemas para el Instituto de Investigaciones Económicas de la UNAM y desarrollador del proyecto Debian GNU/Linux.
http://gwolf.org

Comienzo con la escritura de esta columna el 22 de octubre del 2016. Para quienes trabajamos en el campo de la seguridad informática el día de ayer fue particularmente intenso: Por un lado, nos enfrentamos con un tremendo ataque de denegación de servicio distribuido (DDoS) que dejó fuera de la red al proveedor de resolución de nombres Dyn DNS — con ello, a sitios del calibre de Twitter, Amazon, Whatsapp o Reddit. Y, por el otro, muchos administradores de sistema estuvimos atareados parchando nuestros sistemas por culpa de la "vaca sucia", conocida formalmente como CVE-2016-5195, una vulnerabilidad de escalación de privilegios en el núcleo de Linux.

Si bien no hay más relación entre estos dos hechos que la casualidad de su fecha de publicación, es muy probable que su impacto a futuro se mantenga entrelazado.

Se ha hablado del internet de las cosas en muchos contextos, mayormente positivos, especulando con anticipación acerca de la forma que tomará el futuro de nuestro campo (tanto hacia adentro, a cómo los profesionales lo afrontamos; como hacia afuera, cómo será el impacto hacia los usuarios y la sociedad en general). Sin ir más lejos, el número 51 de Software Gurú fue dedicado íntegro a este tema.

Como en todo tema que especula acerca del futuro, siempre hay algún pesimista que le ve el lado tenebroso. En este caso, traigo a referencia un par de artículos recientemente escritos por Matthew Garrett:

En el primero de estos artículos [1], Matthew describe cómo –por mera curiosidad por ver lo que había conectado a un cable Ethernet en un hotel– pudo asumir el control de las lámparas y cortinas de todos los cuartos del hotel. El segundo [2] describe algunos ataques que lanzó contra focos inteligentes que compró para fines de mera investigación... pero que podrían volver locos a muchos usuarios menos involucrados en el tema que él.

¿A QUÉ VIENE ESTO AL CASO?

El principal factor característico del ataque del 21 de octubre fue que provino de una botnet (red de equipos vulnerados, controlados por un mismo atacante) generada mediante el programa atacante Mirai, palabra que significa futuro en japonés. Mirai es un programa que busca y explota vulnerabilidades conocidas en las configuraciones por default de diversos sistemas del rango del Internet de las cosas — principalmente, cuentas abiertas por omisión. De hecho, el código fuente de Mirai está disponible en github [3].

Cabe mencionar que Mirai (cuyo código se hizo público varios días antes del ataque en cuestión), fue empleado también en el ataque al sitio Web del investigador en temas de seguridad informática Paul Krebs [4]. En ambos casos, la cantidad de tráfico generado rondaba los 600Gbps.

La forma en que Mirai obtiene control de los dispositivos es bastante sencilla y poco sofisticada. No se centra en aprovechar vulnerabilidades de validación de entradas, como la mayor parte de los gusanos; tampoco aprovecha mega-agujeros comunes, para los cuales se ha convertido en moda una publicación con todo y nombre y logotipo (piensen en Heartbleed para OpenSSL, Badlock para Windows y Samba, y el ya mencionado Dirty CoW para Linux). No, lo que hace Mirai es mucho menos sofisticado, ya que el grueso del ataque se centra en intentar autenticarse como administrador usando contraseñas comunes o predeterminadas (ej. admin/admin). La figura 1 muestra un pedazo del código de bot/scanner.c donde se aprecia esto.

```
add_auth_entry("\x50\x4D\x40\x56", "\x43\x46\x4F\x48\x4C", 8);                  // root     admin
add_auth_entry("\x43\x46\x4F\x48\x4C", "\x43\x46\x4F\x48\x4C", 7);             // admin    admin
add_auth_entry("\x50\x4D\x40\x56", "\x1A\x1A\x1A\x1A\x1A\x1A", 6);             // root     888888
add_auth_entry("\x50\x4D\x40\x56", "\x54\x4F\x4A\x46\x48\x52\x41", 5);         // root     xmhdipc
add_auth_entry("\x50\x4D\x40\x56", "\x46\x47\x44\x43\x57\x4E\x56", 5);         // root     default
add_auth_entry("\x50\x4D\x40\x56", "\x48\x57\x43\x4C\x56\x47\x41\x4A", 5);     // root     juantech
add_auth_entry("\x50\x4D\x40\x56", "\x13\x10\x11\x16\x17\x14", 5);             // root     123456
add_auth_entry("\x50\x4D\x40\x56", "\x17\x16\x11\x10\x13", 5);                 // root     54321
add_auth_entry("\x51\x57\x52\x52\x4D\x50\x56", "\x51\x57\x52\x52\x4D\x50\x56", 5); // support  support
add_auth_entry("\x50\x4D\x40\x56", "", 4);                                     // root     (none)
add_auth_entry("\x43\x46\x4F\x48\x4C", "\x52\x43\x51\x51\x55\x4D\x50\x46", 4); // admin    password
add_auth_entry("\x50\x4D\x40\x56", "\x50\x4D\x4D\x56", 4);                     // root     root
add_auth_entry("\x50\x4D\x40\x56", "\x13\x10\x11\x16\x17", 4);                 // root     12345
add_auth_entry("\x57\x51\x47\x50", "\x57\x51\x47\x50", 3);                     // user     user
add_auth_entry("\x43\x46\x4F\x48\x4C", "", 3);                                 // admin    (none)
add_auth_entry("\x50\x4D\x40\x56", "\x52\x43\x51\x51", 3);                     // root     pass
add_auth_entry("\x43\x46\x4F\x48\x4C", "\x43\x46\x4F\x48\x4C\x13\x10\x11\x16", 3); // admin    admin1234
add_auth_entry("\x50\x4D\x40\x56", "\x13\x13\x13\x13", 3);                     // root     1111
add_auth_entry("\x43\x46\x4F\x48\x4C", "\x51\x4F\x41\x43\x46\x4F\x4B\x4C", 3); // admin    smcadmin
add_auth_entry("\x43\x46\x4F\x48\x4C", "\x13\x13\x13\x13", 2);                 // admin    1111
add_auth_entry("\x50\x4D\x40\x56", "\x14\x14\x14\x14\x14\x14", 2);             // root     666666
add_auth_entry("\x50\x4D\x40\x56", "\x52\x43\x51\x51\x55\x4D\x50\x46", 2);     // root     password
add_auth_entry("\x50\x4D\x40\x56", "\x13\x10\x11\x16", 2);                     // root     1234
```

Figura 1. Ejemplo de contraseñas que intenta usar Mirai.

Y es aquí donde realizamos la conexión con el maravilloso mundo del IoT...

Al iniciar el diplomado de Linux en sistemas embebidos de la Facultad de Ingeniería en la UNAM, uno de los primeros temas que abordamos es, naturalmente, qué es un sistema embebido. Y, sin entrar en detalles y disquisiciones, uno de los principales criterios es que es un sistema en el que, independientemente del tamaño físico, la potencia

de cálculo u otros factores; la principal función (esto es, la principal característica que se presenta al usuario) no es la de una computadora de propósito general y en su uso diario podemos incluso olvidarnos de la computadora que incluye. Posiblemente el usuario deba pasar por una fase de configuración, pero el dispositivo en su uso diario pasa inadvertido como parte de la funcionalidad provista por un aparato completamente distinto.

De este modo, por ejemplo, podríamos clasificar de sistema embebido a:

- La compleja red de sensores y actuadores que hay dentro de un automóvil vendido en los últimos veinte años.

- Se impone el ejemplo eterno de un refrigerador inteligente que facilite nuestra vida prediciendo lo que requerimos comprar o tirar (si bien la cantidad de estos dispositivos se mantiene en lo risible). Los refrigeradores inteligentes que he visto, sin embargo, se acercan al límite de esta definición, pues incorporan una pantalla con una instalación de un entorno Android completo.

- Las cámaras de vigilancia y sistemas de portero automático controlables desde un teléfono celular (incluso si éste no está en la casa en cuestión).
  Los sensores biométricos y asistentes de salud vestibles.

- Los teléfonos VoIP, que ya se han convertido en la norma en entornos corporativos.

- Los ya mencionados focos inteligentes y demás soluciones de automatización y centralización de mando en un edificio.

Y, claro, un universo de dispositivos que va creciendo imparablemente, y ya superan ampliamente a las computadoras de uso frontal. En esta lista no entrarían los dispositivos de cómputo móvil, como las tabletas y los celulares, pues para éstos, su principal función (muy por encima de la que hasta hace diez años era la primaria, mantener una conversación de voz entre dos participantes) es indefectiblemente proveer una interfaz de cómputo.

Una diferencia fundamental, de cierto modo implícita en la definición, radica en lo relativo a la flexibilidad del sistema: dado que no se trata de dispositivos de uso genérico, el conjunto de programas que ejecutan podría mantenerse estable a largo plazo. Los diseñadores del software que va a manejarlo buscan ofrecer un entorno estable y duradero... y no siempre consideran brindar funcionalidad para la actualización, ya sea por el espacio adicional que esto requeriría o por considerarlo como un factor que complica la vida del público objetivo.

Esto, claro está, no es nuevo. El mismo problema se presentaba hace diez o veinte años — Pero si los programas que controlan el funcionamiento de mi cámara digital (producida en 2005), proyector de video (del 2008) o impresora (una reliquia del 2001, todavía perfectamente funcional) tienen vulnerabilidades que los hacen susceptibles a que un atacante se apodere de su funcionamiento... ¿Cuál es el riesgo real? Dado que son equipos que no cuentan con ningún tipo de conexión a red, tan bajo que resulta despreciable.

Los dispositivos IoT dependen de una conexión a Internet, no para su función principal, pero sí para funcionalidad altamente deseable para el usuario. Cómo actualizar a estos dispositivos ante una vulnerabilidad es, por decir lo menos, un problema complejo — Si se hace de forma automática, los problemas en que incurriría su compañía en caso de que una actualización no funcionara correctamente no serían asunto nada menor.

Y... toca volver a las contraseñas. A pesar de que el usuario que busca expresamente equipos IoT probablemente pase al menos algunos minutos familiarizándose con su configuración, cambiando contraseñas y ajustando funcionalidad, la experiencia demuestra que una amplia proporción de usuarios simplemente enchufan el dispositivo y se olvidan de él o lo dejan con las configuraciones predeterminadas en aras de no tener que buscar los manuales a futuro.

Como profesionales del desarrollo de software, muy probablemente nos tocará diseñar sistemas que caigan en la definición del Internet de las Cosas. No podemos olvidar nuestra responsabilidad ante el mundo. Desplegar masivamente un sistema con contraseñas fijas (o incluso generadas algorítmicamente alrededor de un valor visible desde fuera, como la dirección MAC) es una invitación a que nuestros equipos sean víctimas de mal uso. No proveer interfaces confiables y bien probadas para la actualización es un pecado igualmente grave.

La usabilidad de Internet a futuro depende de cada uno de nosotros.

Referencias

[1] M. Garrett. "I stayed in a hotel with Android light switches and it was just as bad as you'd imagine". https://mjg59.dreamwidth.org/40505.html
[2] M. Garrett. "I've bought some more awful IoT stuff". https://mjg59.dreamwidth.org/43486.html
[3] https://github.com/jgamblin/Mirai-Source-Code/
[4] "Source for IoT botnet mirai released". Krebs on Security. https://krebsonsecurity.com/2016/10/source-code-for-iot-botnet-mirai-released/

1

DOBOT M1

Hasta ahora, contar con brazos robóticos es un lujo que solamente se pueden dar grandes empresas con amplios recursos económicos, ya que típicamente cada brazo tiene un costo superior a los 40 mil dólares. La empresa Dobot está decidida a cambiar esto con su robot M1, dirigido a pequeñas empresas y entusiastas, que se espera pronto esté disponible comercialmente con un precio de alrededor de 2 mil dólares. El M1 es un brazo robótico tipo SCARA que destaca no solo por su precio sino por su versatilidad. Brinda una precisión de .02 mm, velocidad de 200 grados por segundo, soporta cargas de hasta 1.5 kg y tiene un alcance de hasta 4 metros. Se le pueden poner distintas cabezas para que realice distintas tareas como: tomar y mover objetos, soldar, hacer pirograbado láser, e incluso impresión 3D. El robot es programable de distintas formas. Por ejemplo, se puede poner en modo de aprendizaje y guiar al robot manualmente para que aprenda los movimientos que debe hacer y replicarlos. También cuenta con un ambiente de programación visual, y con SDKs para distintos lenguajes como Python, Java y C++. Dobot actualmente tiene una campaña en kickstarter para el M1 y planea entregar las unidades en mayo del 2017. Esperamos que antes de que termine el año ya esté disponible para el público en general. Más información en http://www.dobot.cc

2

BOLA LIMPIA PISOS MOCORO

Desde hace tiempo las principales empresas de electrodomésticos nos han ofrecido aspiradoras robot sofisticadas con precios superiores a los 400 dólares, que son ideales para casas familiares con alfombra. Pero si tu vives en un pequeño apartamento con pisos laminados o de madera, probablemente una aspiradora de esas sería demasiado. Afortunadamente, nuestros amigos japoneses han creado Mocoro, una linda bola de peluche con un costo cercano a los 40 dólares que rueda libremente por tu piso mientras captura el polvo. Mocoro se mueve por sí sola detectando paredes y obstáculos. Después de 15 minutos se detiene automáticamente para ahorrar energía, pero basta con que la patees ligeramente para que reinicie su actividad. Su cubierta es removible y lavable. Y como podrás imaginarte, puede brindar mucho entretenimiento a tus gatos.

3

PATINETAS ELÉCTRICAS BLINK

Para quienes alguna vez han andado en patineta, una fantasía común es adaptar algún motor para hacer girar las ruedas, lo cual sería especialmente útil en subidas. Bueno, pues la empresa Arcon ha lanzado la línea Blink de patinetas eléctricas que son justamente eso: patinetas con una batería recargable empotrada que habilita unos pequeños motores que están insertados dentro de las ruedas. Para acelerar y frenar utilizas un pequeño control remoto que traes en tu mano, así que puedes mantener tu postura natural para balancearte y girar. Existen distintos modelos que varían en términos de velocidad, duración de batería y precio; yendo desde la Blink Lite, que pesa poco más de 3 kg y da una velocidad de 15 km/h con un rango de 8 km por un costo cercano a los 250 dólares; hasta la Blink Quattro que alcanza los 37 km/h con un rango de más de 30 km por un precio de 1,700 dólares. Más información en http://actonglobal.com

4

OCCIPITAL BRIDGE

Bridge es un nuevo visor de realidad virtual aumentada creado por la empresa Occipital que permite superponer objetos virtuales con el mundo real. Este visor es compatible con iPhone (6, 6s y 7) pero a diferencia de otros visores de realidad virtual (ej. Samsung Gear VR) el Bridge cuenta con un sensor estructural que hace un mapeo 3D del lugar donde te encuentras y sobre éste coloca una capa de realidad virtual. Bridge está pensado principalmente para entretenimiento pero en el futuro se le podrían agregar más aplicaciones. Uno de los juegos preinstalados consiste en una mascota virtual que interactúa contigo. Por ejemplo, puedes jugar a lanzar la pelota y este robot esquivará mesas, cajas, sillas o lo que se le cruce en el camino para ir por ella. Bridge cuenta con una plataforma de desarrollo que permite construir tus propias aplicaciones. Bridge ya está disponible en cantidades limitadas a un precio de $499 dólares y se espera que en el segundo trimestre del 2017 se amplié su disponibilidad y su precio baje a 399 dólares. Más información en https://bridge.occipital.com

5

ARDUINO MKRZERO

La fundación Arduino lanzó la tarjeta MKRZero, la cual combina las capacidades de procesamiento de la tarjeta Zero con el minúsculo tamaño de la serie MKR por un precio cercano a los 22 dólares. La MKRZero usa un microcontrolador Atmel SAMD21 que tiene un procesador ARM Cortex M0 de 32 bits. La MKRZero es ideal como herramienta educativa para makers principiantes, ya que contiene en un paquete pequeño y económico todo lo que necesitas para soportar un microcontrolador. Entre sus ventajas destaca su pequeño tamaño (65 x 25 mm), bajo consumo de energía y soporte integrado para tarjetas SD.

Humor

Original de Manu Cornet. Publicado en bonkersworld.net

Pasar de Profesional de **TI** a **CDO**

Por Arístides Palma

El avance actual de la tecnología ha obligado a las empresas y a sus directivos a dejar atrás la percepción de que la tecnología de información queda acotada a un área o departamento. Es gracias a esto que muchas empresas ofrecen salarios llamativos a los profesionistas de TI, pero a la vez la formación y el perfil que se solicita tiende a ser más global e involucrar distintas disciplinas.

Tradicionalmente, la máxima aspiración para un profesionista de TI dentro de un corporativo ha sido llegar a ser el director de sistemas (Chief Information Officer, CIO). Sin embargo, ante la ola de transformación digital que están viviendo la mayoría de los corporativos, para el profesional de TI las oportunidades económicas y por tanto laborales ya lo ubican más allá de aspirar a ser un CIO; la tendencia ahora es convertirlo en CDO (Chief Digital Officer), quien ahora tiene un perfil más amplio, no solamente informático.

"EL PERFIL DEL PROFESIONISTA DE TECNOLOGÍAS DE INFORMACIÓN YA NO ES 100% TECNOLÓGICO."

Por ejemplo, las organizaciones en la actualidad deben hacer uso del big data, convirtiéndolo en el activo más importante de la empresa, siendo usado por la gente de ventas, mercadotecnia y comunicación, ya sea para dar seguimiento a sus clientes, captar nuevos prospectos de compradores, analizar los mensajes que hablan de las marcas y evitar una crisis de reputación. Asimismo, mediante estos datos las empresas de manufactura pueden analizar de mejor manera las épocas de mayor venta de determinados productos y así tener el control de existencias suficientes en stock, para solicitar la materia prima necesaria.

Para encontrar a un CDO, en la mayoría de los casos las empresas no necesitan echar mano de gente externa, ya que lo más común es que las personas más adecuadas ya estén dentro de sus organizaciones, pues son quienes conocen las entrañas de la empresa. Las empresas deben considerar a los recursos humanos con que cuentan, sin importar el área de la que vengan; lo que es importante, es que sean capaces de entender los datos y la información que arrojan sus diversos sistemas y puedan explotarlos para generar mayor rentabilidad en la empresa.

En nuestra empresa hemos notado que el perfil 100% tecnológico ha ido cambiando, pues cada vez es más común encontrar profesionales de TI capaces de vender, planear y operar estrategias de marketing basadas en análisis de big data, o analizar datos de producción para identificar oportunidades de mejora. En general estos profesionales tienen una mayor apertura a trabajar con otras áreas, teniendo una amplia cultura que puede abarcar distintos dominios: contabilidad, finanzas, mercadotecnia, logística, ingeniería industrial; con una visión más estratégica dentro de la organización, convirtiéndolo en un activo muy importante para la empresa que a su vez termina por mejorar su remuneración económica.

La visión de estos profesionistas debe estar fuertemente enfocada a resultados, y de acuerdo con las necesidades que vaya encontrando en su camino irá creando o adecuando la tecnología que reditúe en beneficio del negocio, así como capacitar a cada uno de los integrantes de la organización, desde las áreas de ventas, administrativas, mercadotecnia y TI para que se logren cada una de las metas del equipo.

Ese es el perfil del profesional que buscan las empresas. ¿Parece difícil? Puede ser, pero en la actualidad tanto el experto como la empresa deben adecuarse a esas necesidades para crecer de manera conjunta y sobresalir.

El Ingeniero Arístides Palma es fundador y Director General de Zafiro Software, empresa dedicada a diseñar, desarrollar e implementar avanzadas soluciones de administración y planeación de recursos empresariales, análisis de datos y business intelligence para una amplia variedad de empresas de diversos giros y tamaños, ayudándolos a alcanzar su máximo potencial de negocios.