Artículo: COMEII-17014

III CONGRESO NACIONAL
DE RIEGO Y DRENAJE COMEII 2017
Puebla, Pue., del 28 al 30 de noviembre de 2017

DISEÑO DE UNA ARQUITECTURA EN LA NUBE PARA LA PROGRAMACIÓN DEL RIEGO CON DISPOSITIVOS MÓVILES, APLICANDO APRENDIZAJE AUTOMÁTICO

Daniel Arturo Salinas Verduzco[1]*; Ernesto Sifuentes Ibarra[2]; Waldo Ojeda Bustamante[3]; Yobani Martínez Ramírez[1]

[1]Facultad de Ingeniería. Universidad Autónoma de Sinaloa. C.U. Fuente de Poseidón y Ángel Flores s/n, Col. Jiquilpan Módulo B2, C.P. 81226, Los Mochis, Sinaloa.

dsalinasv@uas.edu.mx - telf.: (668) 812 7641 (*Autor de correspondencia)

[2]Estudiante de Doctorado en Ciencias y Tecnología del Agua (Instituto Mexicano de Tecnología del Agua), Jiutepec, Morelos, México C.P. 62550 e investigador de INIFAP-Campo Experimental Valle del Fuerte, Juan José Ríos, Sinaloa, México C.P. 81110
[3]Coordinación de Riego y Drenaje. Instituto Mexicano de Tecnología del Agua.Paseo Cuauhnáhuac 8532, Progreso, Jiutepec, Morelos, C.P. 62550. México.

Resumen

Se presenta el diseño de una arquitectura que utiliza dispositivos móviles como terminales cliente usando la aplicación IrriMoist, esta cuenta con una base de datos local SQLite para almacenar los datos de las parcelas del productor, además interactúa con los servicios en la nube de FireBase, el cual forma parte de Google Cloud Platform y con algunos microservicios implementados en Microsoft Azure Machine Learning Studio. IrriMoist guarda la información de las texturas del suelo, el cultivo y la fecha de siembra de la parcela, con lo cual puede calcular el balance hídrico necesario en el pronóstico del riego. También se conecta a la base de datos FireBase para consultar el clima de la estación más cercana a la parcela. IrriMoist debe contar con la información anterior para calcular los grados día acumulados del cultivo sembrado en la parcela, con esto puede consultar en los microservicios de Microsoft Azure Machine Learning Studio el coeficiente de cultivo, la profundidad de la raíz y el factor de abatimiento para el tiempo que ha transcurrido a partir de la fecha de siembra. Por último, IrriMoist recibe la lectura del tipo de sensor de humedad elegido para calcular a los cuantos días es necesario regar.

Palabras clave adicionales: sistemas inteligentes, cómputo en la nube móvil, gestión del riego

Introducción

Las tecnologías de información son una herramienta de vital importancia para la innovación en el sector del agua, ya que pueden eficientizar los procesos para gestionar los recursos hídricos del país (Hernández, 2014).

En México existen varias propuestas para apoyar estos procesos tales como Spriter (Ojeda & Sifuentes Ibarra, 2003), CalRiego (García, 2005), IrriModel (Sifuentes Ibarra & Quintana Quiroz, 2013) e IrriNet (Valencia *et al.*, 2013), además hay otros trabajos con prototipos entre los que destacan el uso de dispositivos móviles y dispositivos del internet de las cosas (Espinosa, 2010; Espinosa & Nolasco, 2011; Rodríguez & Nolasco, 2014; Rodriguez, 2015).

También en otros países están desarrollando estas herramientas con tecnologías de punta de las cuales tenemos los servicios web, el computo en la nube, sistemas de información geográfica, sistemas de soporte a las decisiones y el modelado de procesos de negocio (Xu *et al.*, 2011; Urrestarazu *et al.*, 2012; Yu *et al.*, 2014).

Sin embargo, estos sistemas solo están pensados para usarse en una computadora local o a través de una página web, usan herramientas de desarrollo tradicionales que requieren bastante trabajo de mantenimiento porque hay que detallar que y como va a llevar a cabo las tareas el sistema.

Esto limita su adopción a larga escala y adaptación de cambios a corto plazo, por lo tanto, se propone una arquitectura basada en servicios de computo en la nube con escalabilidad automática y la aplicación de algoritmos de aprendizaje automático para automatizar la generación de modelos que predicen cuando y cuanto regar.

El presente trabajo es una propuesta tecnológica que servirá como base para implementar un sistema inteligente capaz de escalarse a la demanda de nuevos usuarios y permitirá agregar mejoras al sistema con menor esfuerzo usando las nuevas tecnologías de datos masivos y de aprendizaje automático.

Este tipo de tecnologías permiten diseñar sistemas que se adaptan a los datos ya que encuentran los patrones de datos conocidos con lo que se puede predecir con mucha exactitud los resultados de entradas de datos no conocidos (Awad & Khanna, 2015).

Materiales y métodos

A continuación, se describen las herramientas utilizadas para el diseño de la arquitectura:

Micosoft Azure Machine Learning Studio

Es una herramienta colaborativa, permite configurar módulos tan solo arrastrándolos y soltándolos en un lienzo, estos permiten construir, probar y desplegar soluciones de análisis predictivo aplicando algoritmos de aprendizaje automático en conjuntos de datos. Los modelos generados pueden ser convertidos en servicios web para que sean fácilmente consumidos por cualquier tipo de aplicación que soporte el consumo de servicios REST (Fontama *et al.*, 2015).

Firebase

Es una plataforma de desarrollo que ofrece diversos servicios, uno de ellos es la base de datos en tiempo real, esta provee un servicio de backend por medio de una API que permite sincronizar los datos de las aplicaciones del lado del cliente con los datos almacenados en la nube de Firebase. Además, permite asegurar los datos con reglas de seguridad y gestionar el inicio de sesión con múltiples proveedores. Después de integrar Firebase a la aplicación móvil nos olvidamos de escribir el código backend y de la gestión de la infraestructura (Fu, 2017).

Delphi

Es un entorno de desarrollo integrado para programar aplicaciones con el lenguaje Object Pascal. Incluye una extensa biblioteca de componentes para el desarrollo de aplicaciones nativas para múltiples plataformas con servicios flexibles en la nube y una amplia conectividad IoT. La clave de Delphi está en el desarrollo de aplicaciones móviles porque la app se diseña una vez desde la misma base de código y se compila a Android y iOS de manera nativa (Gtowacki, 2017).

SQLite

Es una base de datos relacional integrada que existe simbióticamente dentro de la aplicación a la que sirve como parte del programa que la hospeda. Gracias a que está dentro de la aplicación no se requiere configuración de red ni administración. Entre sus características están la configuración cero, la portabilidad, la compacidad, la simplicidad, la flexibilidad, una licencia liberal, la confiabilidad y la conveniencia (Owens, 2006).

Análisis y discusión de resultados

Arquitectura del sistema

La figura 1 muestra el diagrama de despliegue del sistema, está formado por tres nodos, el nodo Azure que es donde se suben los conjuntos de datos para entrenar los modelos aplicando algoritmos de aprendizaje automático, el nodo Firebase donde se encuentran los datos de temperatura y evapotranspiración de las estaciones agroclimáticas y el nodo IrriMoist que representa la aplicación móvil que guarda los datos de la parcela del usuario en la base de datos SQLite, estos datos almacenados en el dispositivo móvil permitirán consultar los otros dos nodos para predecir cuándo y cuánto regar.

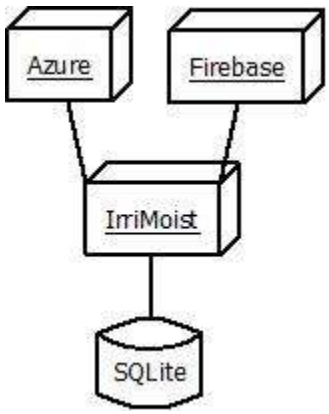

Figura 1. Diagrama de despliegue.

Componentes

La figura 2 muestra los componentes del nodo Azure, tiene un componente experimento que lee el conjunto de datos con cálculos conocidos del coeficiente de cultivo, de la profundidad de raíz y del factor de abatimiento para una etapa de crecimiento de un cultivo dado a cierta cantidad de grados día acumulados. Por otro lado, tenemos un conjunto de datos con los tipos de sensores de humedad, la medición del sensor y la humedad. En el experimento se elige el algoritmo de aprendizaje automático para entrenar el modelo, este se almacena en el siguiente componente que después se convierte a un microservicio.

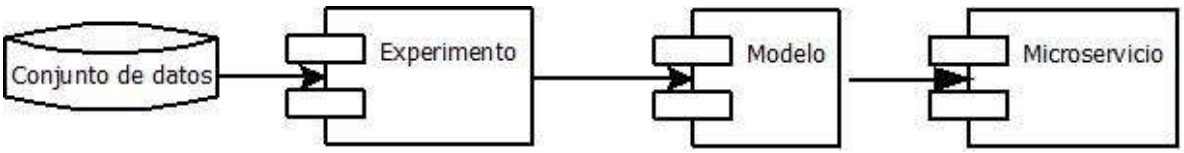

Figura 2. Componentes de Azure.

La figura 3 muestra los componentes del nodo Firebase, el componente autenticación ofrece el servicio de inicio de sesión a la aplicación móvil, una vez autenticada esta recibe un token para poder ejecutar consultas de las estaciones agroclimáticas mediante el backend que nos ofrece Firebase con su API REST.

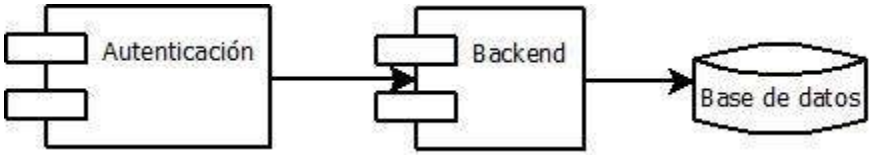

Figura 3. Componentes de Firebase.

La figura 4 muestra los componentes de la aplicación móvil IrriMoist desarrollada con Delphi, iniciamos con el componente menú que se encarga de darnos acceso a los catálogos de la aplicación móvil y al pronóstico de la parcela seleccionada. El componente catalogo permite guardar, leer, actualizar y eliminar los datos de las tablas de texturas, cultivos y parcelas en la base de datos local SQLite. También cuenta con un componente consultas que se encarga de abrir la conexión al servicio de Firebase donde solicita los datos agroclimáticos de la estación más cercana, estos datos permiten a la aplicación móvil calcular los grados día acumulados hasta la fecha actual a partir de la fecha de siembra que se capturo en la parcela elegida, los grados día acumulados se envían al componente cálculos para recibir el coeficiente de cultivo y profundidad de raíz que ha sido modelado en Azure ML Studio, estos coeficientes permiten solicitar al componente sensor la humedad actual del suelo en base a la lectura que haya hecho el usuario con el tipo de sensor elegido.

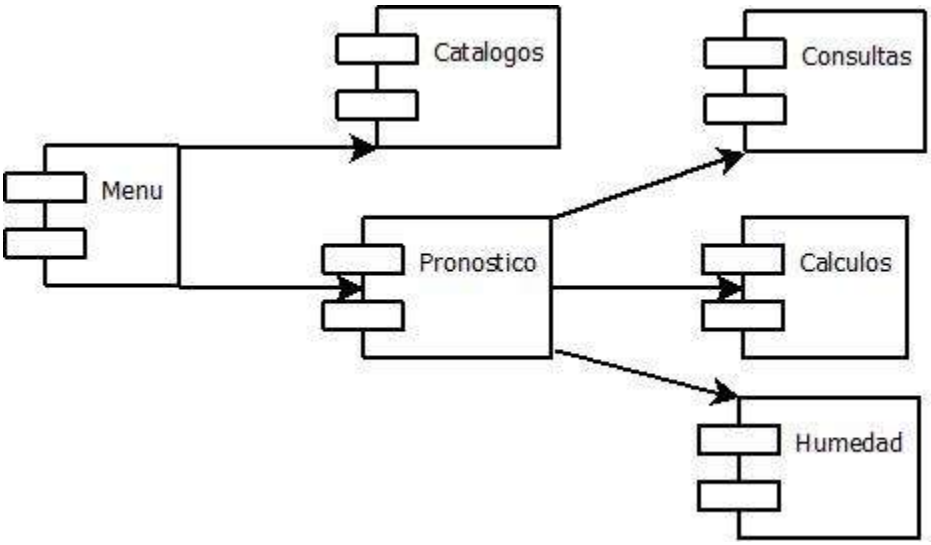

Figura 4. Componentes de IrriMoist.

Comunicación

Como se pudo ver en la figura 1 existe una comunicación entra la aplicación móvil IrriMoist y los servicios en la nube de Firebase y de Azure ML Studio, ya que ambos servicios ofrecen llamadas a servicios web de tipo REST, es por lo tanto, la mejor opción para la comunicación entre los nodos, además estos se llevan muy bien con las aplicaciones móviles porque son fáciles de invocar, producen una respuesta formateada y usualmente más fácil de analizar que otros formatos basados en arboles como XML (Christensen, 2009).

Aprendizaje automático

El conjunto de datos subido a Azure ML Studio incluye 9 cultivos de ciclos cortos, intermedios y largos con 5 diferentes sistemas de riego.

Los entrenamientos de los modelos de aprendizaje automático están basados en el trabajo “Desarrollo de una arquitectura basada en microservicios para pronosticar riegos aplicando técnicas de aprendizaje automático” solo que ahora en lugar de calcular los coeficientes de cultivo (Kc) y de profundidad de raíz (Pr) por etapa se hicieron por día (Salinas *et al.*, 2016).

Al final del entrenamiento el algoritmo que dio mejores resultados fue la red neuronal cambiando los parámetros número de redes ocultas configurado a 180, el número de iteraciones de aprendizaje configurado a 1000 y el tipo de normalizador Gaussiano.

Con esto se obtuvieron los siguientes resultados de R^2 tal y como se puede ver en el cuadro 1.

Cuadro 1. Coeficientes entrenados en Azure ML Studio.

Coeficiente	R^2
Kc	0.983802
Pr	0.973043

El entrenamiento del modelo de aprendizaje automático para los sensores de humedad sigue igual que el trabajo antes mencionado, la diferencia fue que se incluyeron otros 4 sensores y 3 texturas nuevas obteniendo una R^2 de 0.938778.

Una vez entrenados los modelos en la Azure ML Studio se procede a generar los servicios web que serán utilizados por la aplicación móvil IrriMoist. Aprovechando esta infraestructura que nos ofrece Azure es fácil diseñar sistemas con una arquitectura orientada a microservicios, estos son bloques de software que solo hacen una cosa pero bien hecha, con ello podemos lograr un desarrollo iterativo del sistema desplegando nuevas características por partes en base a su prioridad aplicando metodologías agiles como Scrum (Familiar, 2015).

Aplicación móvil

Por último, se presenta el diseño de las pantallas para la aplicación móvil IrriMoist iniciando con el menú en la figura 5, en esta pantalla se puede abrir los catálogos de texturas, cultivos o parcelas, también se puede entrar al pronóstico del riego.

La figura 6 muestra el listado de texturas, aquí se puede agregar nuevas texturas con el botón agregar de la esquina inferior derecha, la figura 7 muestra como se capturan el nombre de la textura y las cantidades de arcilla, arena y limo.

En la figura 8 muestra el listado de cultivos, aquí se puede agregar nuevos cultivos con el botón agregar, en la figura 9 se muestra la captura del nombre del cultivo y las temperaturas umbrales mínima y máxima para calcular los grados día crecimiento.

La figura 10 igual que en anteriores catálogos muestra un listado, en este caso de las parcelas donde también se puede agregar nuevas igual que en los listados anteriores, en la figura 11 se captura el nombre de la parcela y se eligen la textura, el cultivo, la fecha de siembra y el tipo.

La figura 12 muestra el pronóstico de los días al riego donde se elige la parcela, el tipo de sensor de humedad y se captura la lectura del sensor para pronosticar.

Figura 5. Menú de IrriMoist.

Figura 6. Listado de texturas de IrriMoist.

Figura 7. Edición de la textura de IrriMoist.

Figura 8. Listado de cultivos de IrriMoist.

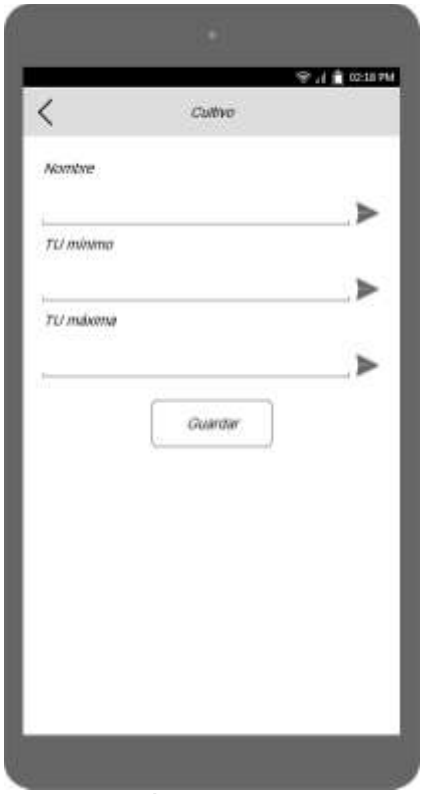

Figura 9. Edición del cultivo de IrriMoist.

Figura 10. Listado de parcelas de IrriMoist.

Figura 11. Edición de la parcela de IrriMoist.

Figura 12. Pronóstico de los días al riego de IrriMoist.

Conclusiones

Con este trabajo se ha logrado diseñar una arquitectura que es elástica, automatizada y distribuida gracias a plataformas en la nube existentes. Es elástica porque se puede escalar automáticamente replicando los microservicios en base a la demanda de los usuarios de forma transparente. Es automatizada porque permite agregar reglas que en base a eventos que ocurran se puede aumentar o disminuir los recursos usados por el sistema. Y es distribuida porque cada servicio web es independiente, esto permite aplicar metodologías agiles como Scrum para priorizar las funcionalidades y desarrollarlas por partes sin afectar lo que ya está desplegado.

Agradecimientos

Se agradece a la dirección de la Facultad de Ingeniería Mochis por el apoyo económico brindado para poder asistir al COMEII 2017.

Referencias Bibliográficas

Awad, M., & Khanna, R. (2015). Machine Learning and Knowledge Discovery BT - Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers (pp. 19–38). https://doi.org/10.1007/978-1-4302-5990-9_2

Christensen, J. H. (2009). Using RESTful web-services and cloud computing to create next generation mobile applications. *Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications*, 627–634. https://doi.org/10.1145/1639950.1639958

Espinosa, O. L. (2010). Automatización del riego con base en dispositivos móviles. Retrieved from http://www.biblio.colpos.mx:8080/jspui/handle/10521/179

Espinosa, O., & Nolasco, A. (2011). Prototipo para automatizar un sistema de riego multicultivo* prototype for automating a multicropping irrigation system. *Revista Mexicana de*. Retrieved from http://scielo.unam.mx/pdf/remexca/v2n5/v2n5a3.pdf

Familiar, B. (2015). *Microservices, IoT, and Azure*. https://doi.org/10.1007/978-1-4842-1275-2

Fontama, V., Barga, R., & Tok, W. H. (2015). *Predictive Analytics with Microsoft Azure Machine Learning 2nd Edition*. https://doi.org/10.1017/CBO9781107415324.004

Fu, C. (2017). *Build Mobile Apps with Ionic 2 and Firebase: Hybrid Mobile App Development*. Retrieved from http://gen.lib.rus.ec/book/index.php?md5=a3bf84b1f1031bdd966bfe8d90e167fb

García, A. R. (2005). Sistema computacional para la calendarización del riego basado en internet. Retrieved from http://www.sidalc.net/cgi-bin/wxis.exe/?IsisScript=UACHBC.xis&method=post&formato=2&cantidad=1&expresion=mfn=102297

Gtowacki, P. (2017). *Expert Delphi*.

Hernández González, G. (2014). *Fuerzas impulsoras de las Tecnologías de Información y Comunicación en México: una aproximación desde el sector agua*.

Ojeda, W., & Sifuentes Ibarra, E. (2003). Sistema de pronóstico del riego en tiempo real. *Universidad de México*, 128–134. Retrieved from http://www.revistadelauniversidad.unam.mx/ojs_rum/index.php/rum/article/view/15943

Owens, M. (2006). *The Definitive Guide to SQLite*.

Rodríguez, G. A., & Nolasco, A. Q. (2014). Servicio de riego mediante internet y dispositivos móviles en la zona del Colegio de Postgraduados en Ciencias Agrícolas. *Revista Mexicana de*. Retrieved from http://www.scielo.org.mx/scielo.php?pid=S2007-09342014000300001&script=sci_arttext&tlng=pt

Rodriguez, G. E. (2015). Diseño y desarrollo de un prototipo de riego automático controlado con Raspberry Pi y Arduino. Retrieved from https://upcommons.upc.edu/handle/2099.1/25074

Salinas Verduzco, D. A., Sifuentes Ibarra, E., & Ojeda Bustamante, W. (2016). Desarrollo de una arquitectura basada en microservicios para pronosticar riegos aplicando técnicas de aprendizaje automático, 1–13.

Sifuentes Ibarra, E., & Quintana Quiroz, J. (2013). " IRRIMODEL" Programación Integral y Gestión del Riego a Través de Internet. Manual del Usuario. Version 1.0. Retrieved from http://biblioteca.inifap.gob.mx:8080/xmlui/handle/123456789/3970

Urrestarazu, L., Díaz, J., & Poyato, E. (2012). Development of an integrated computational tool to improve performance of irrigation districts. *Journal of*. Retrieved from http://search.ebscohost.com/login.aspx?direct=true&profile=ehost&scope=site&authtype=crawler&jrnl=14647141&AN=77977557&h=Dsbl1vfcKqXxSN9%2FjDd5B0GC60tZwjJcuiQh2e7h%2B6F5Cz7KG3EQX7bJ3YUzA%2BQteElQfoYMB4dcNxvExyGWFQ%3D%3D&crl=c

Valencia, E., Castorena, M., Ibarra, M., & López, A. (2013). IRRINET: Sistema en línea para el pronóstico del riego en tiempo real en coahuila. *AGROFAZ*. Retrieved from http://www.agrofaz.mx/wp-content/uploads/articulos/2013132IV_1.pdf

Xu, L., Chen, L., Chen, T., & Gao, Y. (2011). SOA-based precision irrigation decision support system. *Mathematical and Computer Modelling*, *54*(3–4), 944–949. https://doi.org/10.1016/j.mcm.2010.11.020

Yu, P., Yang, T., Kuo, C., & Chen, S. (2014). Development of an integrated computational tool to assess climate change impacts on water supply–demand and flood inundation. *Journal of Hydroinformatics*. Retrieved from http://jh.iwaponline.com/content/16/3/710.abstract