Repositorio Común para Mediciones de Estaciones Meteorológicas Automatizadas

José Hipólito Moyano[1,2][0000−0002−9080−1850] and Karina Mabel Cenci[1,2]

[1] LiSiDi, Departamento de Ciencias e Ingeniería de la Computación, Universidad Nacional del Sur, Bahía Blanca, Argentina
[2] Laboratorio de I+D en Ing. de Software y Sistemas de Información (UNS-CIC Provincia de Buenos Aires)
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Bahía Blanca, Argentina
{jose.moyano,kmc}@cs.uns.edu.ar

Resumen Actualmente, la agrometeorología se encuentra fuertemente respaldada por Estaciones Meteorológicas Automatizadas (EMA). Estos dispositivos recopilan información meterológica que es publicada en servicios en internet y en ocasiones se encuentra disponible con una conexión local. Los servicios pertenecen a los fabricantes, quienes los ponen a disposición del propietario de la EMA, con funciones avanzadas sólo disponibles a cambio de una subscripción paga.
Las entidades que adquieren las EMA, cuentan con distintos modelos, en algunos casos de distintos fabricantes, que presentan la informatión en distintas modalidades y formatos.
La recolección e integración de la información son relevantes para la investigación científica, y las estimaciones de rinde y producción de la industria agropecuaria. Por las características mencionadas, el equipo de meteorología se ve obligado a resolver manualmente la recolección de datos de distintos servicios, y a consolidar esta información en un formato común que permita su procesamiento.
Este trabajo plantea responder a la pregunta si es factible automatizar dicho proceso utilizando conceptos de sistemas distribuidos e Internet de las Cosas (IoT), recolectando en un único repositorio.

Keywords: Agrometeorology · IoT · Distributed Systems ·

1. Introducción

Distintos organismos alrededor del mundo mantienen redes de estaciones meteorológicas propias como la red NOANN [6] del Observatorio Nacional de Atenas, o el sistema RAWS [11] que utiliza el Servicio Forestal de EEUU (USFS). En Argentina se pueden citar la Red de Estaciones Meteorológicas del Instituto Correntino del Agua y Medio Ambiente [1] o la Red de Estaciones Meteorológicas de San Luis [2].

El propósito de estos sistemas es recolectar la información de Estaciones Meteorológicas Automatizadas (EMA) dispersas en una región para mejorar sus estimaciones agrometeorológicas y también como servicio ofrecido al público general.

Su funcionamiento consiste en la recolección de las mediciones en un sitio centralizado, y a través de una interfaz, generalmente un *frontend* web, consultar las distintas variables en rango de fechas.

Esta información tiene tanto propósitos comerciales, como maximizar el rendimiento de la explotación agropecuaria; y objetivos ecológicos y sociales, buscando variaciones climáticas, y meterológicos, mejorando predicciones para implementar acciones preventivas y mitigando catástrofes naturales.

En el caso de las redes de ICAA y de San Luis, la información se encuentra disponible en Internet. Sin embargo, está almacenada en silos independientes por cada estación. Se pueden consultar las variables meteorológicas en cada estación de manera independiente.

Este trabajo plantea responder a la pregunta si es factible automatizar la recolección y consolidación de la información proveniente de distintas EMAs, servicios, y bases de datos, utilizando conceptos de sistemas distribuidos e Internet de las Cosas (IoT), integrando en un único repositorio las mediciones de los equipos, convirtiendo la información a un formato uniforme de manera automática.

A su vez, esta plataforma debe permitir la consulta, análisis, y procesamiento, ofreciendo interfaces y servicios que simplifiquen las tarea del meteorólogo. Resulta muy difícil predecir todas las necesidades que se van a manifestar sobre estos datos, por lo que entre sus propiedades se deben encontrar capacidad de extensión y facilidad de mantenimiento.

Se persigue mejorar la eficiencia y efectividad del analista y alcanzar a un público mayor.

A partir del primer planteo de la factibilidad de una solución, el trabajo se direcciona en responder a las siguientes preguntas: R1: Cómo diseñar una solución distribuida que permita la interoperabilidad entre los diversos artefactos y locaciones y R2: Cuáles son los desafíos relacionados para alcanzar una viable solución. La metodología *Investigación en Ciencias del Diseño* (Design Science Research (DSR))[4][9] es la utilizada para el desarrollo de este proyecto, ya que es apropiada para el desarrollo de un artefacto. En la sección 2, se describe la primera etapa que es el conocimiento del problema; en la sección 3, se sugiere un diseño y el comportamiento y desempeño esperado; en 4 se desarrolla el diseño con todas sus características; la sección 5 presenta el contexto de las pruebas y evalúa los resultados. Las sección 6 propone cómo continuar y refinar el proyecto antes de las conclusiones.

2. Conocimiento del problema

Las EMA publican su información a través de medios diferentes

1. Localmente: Ofrecen una consola o una conexión a una computadora junto con una aplicación que permite visualizar y recolectar la información.
2. Sitio del fabricante: El proveedor de la EMA cuenta con un dominio en internet, donde la estación envía los datos y estos pueden ser consultados por el usuario. Esta modalidad cuenta con funciones que dependen de una subscripción paga.
3. Repositorio a medida: Organizaciones gubernamentales y no gubernamentales construyen sus propias estaciones que entregan las medidas con un mecanismo a medida o *ad-hoc*.

Estos mecanismos existen para las EMA de forma heterogénea. Por ejemplo, las estaciones Davis ofrecen las interfaces 1 y 2, mientras que las estaciones Omixon sólo cuentan con 2. La Bolsa de Cereales de Bahía Blanca utiliza un desarrollo de EMA originado por CONICET y UNS, publicando sus datos con un formato a medida.

Esta información también presenta distintos formatos, estructuras de datos, y unidades; y distintas estaciones, incluso del mismo fabricante, tiene disponibilidad de un determinado conjunto de sensores, con distintas precisión y calidad. Finalmente, tampoco es posible controlar la frecuencia de toma de muestra, ni las unidades en las que se presenta la información. Las estaciones entregan datos continuamente y aceptan configuraciones de distintas medidas que no son consideradas.

La ciencia agrometeorológica integra estas fuentes de información dispares manualmente, convirtiendo unidades, con un proceso generalmente engorroso, difícil de repetir y propenso a errores, utilizando software de planillas de cálculo.

Es necesario encontrar mecanismos automáticos de integración de estas medidas en un repositorio universal que de soporte a consulta y formas de procesamiento automatizado de la información.

3. Sugerencia

La solución a la integración de datos de EMA debe contar con las siguientes propiedades

- Mantener activo el enlace de información de las EMA con los sitios de los fabricantes: Como parte de la intervención, no se debe perder la capacidad de la EMA de reportarse a su fabricante. Algunos fabricantes obligan a sus usuarios a acceder a la información a través de su sitio web y no ofrece información local. En otros casos, las EMA son compartidas con otros organismos y el mecanismo de acceso actual debe mantenerse vigente.
  Mantener intactos los accesos existentes en una estación permite también proponer a organismos con sus propias EMA incorporarse a una red manteniendo sus servicios actuales.
- Normalmente cuando las estaciones pierden conectividad con el destino, las medidas se pierden irremediablemente. En caso de existir problemas de conexión o disponibilidad del repositorio, deben realizarse todos los esfuerzos posibles para que las estaciones no pierdan datos.

- La información debe alojarse en un formato uniforme con las estructuras y los metadatos que soporten la consulta y el procesamiento.

Estos requisitos plantean como solución ideal incorporar un dispositivo, sistema embebido o sistema de IoT que, anexo a la estación, mantenga el enlace de comunicaciones con el sitio del fabricante y recolecte las medidas localmente para transferir al repositorio universal.

La recolección local permite almacenar los datos cuando el enlace de comunicaciones no está disponible, otorgando resiliencia a la estación frente a cortes de conectividad. Este dispositivo debe soportar distintos modelos y configuraciones de estaciones.

Como las estaciones suelen encontrarse en lugares estratégicos, en ocasiones aislados de centros urbanos, sería ideal que soporte actualizaciones *over-the-air* (OTA) para el mantenimiento de software.

Finalmente, resta abordar dos situaciones adicionales. Primero, el fabricante que almacena la información en su propio servicio como única forma de acceso; y aquellas organizaciones que tienen EMAs *custom*.

Además de visualizaciones web, los sitios de los fabricantes proveen una *Application Programming Interface* (API) de consulta. De este modo sus clientes pueden desarrollar sus propias aplicaciones que consulten los datos. En estos casos, es necesario construir un servicio o proceso que corriendo en un sistema realice las consultas, procese la información, y la añada al repositorio universal.

En el caso de las soluciones a medida, las organizaciones que utilicen la solución de repositorio universal no tendrán inconvenientes técnicos de incorporar a su infraestructura existente un servicio que, similar a la función del dispositivo junto a la estación, consulte los datos en la plataforma *custom* y los envíe al servicio centralizado.

3.1. Desafíos relacionados al desempeño

Es posible predecir que el almacenamiento de la información desde las estaciones hacia el repositorio donde se recolectan los datos no demandará una carga alta de procesamiento. Los mensajes son pequeñas cadenas de caracteres UTF-8 representando los distintos campos de una medición.

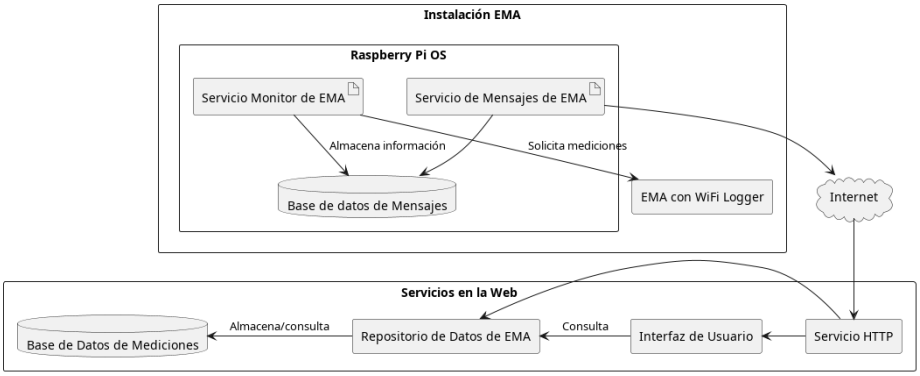
Sin embargo, con el objetivo ambicioso de consultar todas las medidas que provee una EMA con una frecuencia de una vez por minuto, la cantidad de entradas en la base de datos será muy grande. Para generar los informes, habrá una cantidad de formatos de consultas difícil de predecir y la base de datos requerirá índices a medida para soportar las consultas.

En el análisis preliminar de la bibliografía existente, el equipo de trabajo llegó a la conclusión que una base de datos SQL será lo más adecuado [7][5].

En el proceso de Crear, Leer, Actualizar y Borrar (CRUD, del inglés *Create*, *Retrieve*, *Update and Delete*), la mayor parte de la información procesada serán las mediciones de las estaciones, que serán creadas una única vez, y difícilmente sean actualizadas o borradas. La mayor cantidad de las operaciones serán de consulta.

Estas consultas obedecerán a distintos parámetros de filtro, por lo que se hará uso extensivo de los índices concatenados [10] y sus funciones más complejas. Operaciones de consulta sobre índices complejos tienen mejor rendimiento en bases de datos SQL.

4. Desarrollo



El dispositivo que se incorpora a las estaciones, a partir de este punto el *gateway*, siguiendo la terminología IoT, será el responsable de recolectar las mediciones y almacenarlas localmente y transmitirlas al repositorio.

El *gateway* intentará enviar las mediciones al repositorio remoto de datos. En caso de no poder transmitir la información, se conservará localmente todo el tiempo que sea posible, junto con los resultados de los envíos de información.

Las responsabilidades del repositorio serán las de autenticar y autorizar al *gateway* y recibir sus mensajes. Dado que el repositorio remoto funcionará sobre un sistema operativo moderno con recursos muy superiores al del sistema embebido, se delega al repositorio la responsabilidad de procesar la información de la estación y generar mediciones útiles a las consultas de los usuarios.

El repositorio se encargará también de almacenar cualquier dato relevante sobre la estación, la chacra y los datos, como son modelo y marca, latitud y longitud, etc.

Finalmente, un *frontend* web se encontrará a disposición de los usuarios. Este *frontend* será el responsable de consultar la información contenida en el repositorio y presentar visualizaciones acordes a lo solicitado por los responsables del MDA.

4.1. Conexión de la estación con el repositorio

Las estaciones Davis cuentan con una entrada USB que permite conectar el dispositivo WiFi Logger. Este dispositivo permite a la estación conectarse a

Internet y transferir las mediciones al sitio del fabricante. Además WiFi Logger ofrece un *endpoint* HTTP donde solicitar las mediciones actuales de la EMA. Este *endpoint* puede ser utilizado por el servicio de recolección en el *gateway* para recuperar la información.

4.2. *Gateways*

Para resolver el *gateway*, el equipo de trabajo incorpora como parte de la configuración de la EMA, una mini computadora Raspberry Pi 4 (RPi), a la que se le incorpora un segundo enlace inalámbrico USB. Este dispositivo provee de una red a WiFi Logger a través del segundo enlace, y se conecta a internet a través de la conexión integrada en RPi. Se realiza a su vez el ruteo que permite a los mensajes de WiFi Logger alcanzar el sitio web del fabricante y ser almacenados allí, conservando las vías actuales de comunicación como se estableció en los requisitos.

A la RPi se le incorporan dos servicios implementados por el equipo de trabajo, `awsmonitor` y `awsmessenger`. `awsmonitor` se encarga de recolectar los mensajes de las estaciones y almacenarlos localmente. `awsmessenger` lee la información almacenada localmente y envía la información al repositorio remoto.

Como almacén local de la información, se utiliza SQLite. Este motor de base de datos se basa en utilizar un archivo, y garantiza el acceso *thread-safe*,permitiendo que `awsmonitor` pueda guardar mediciones, y que `awsmesseger` pueda consultarlos y registrar los resultados del envío, sin provocar condiciones de carrera sobre los datos.

`awsmonitor` a una frecuencia configurable, con un valor por omisión de 60 segundos, consulta a WiFi Logger por las mediciones realizadas por la estación. La respuesta tal cual es recibida se almacena en la base de datos. `awsmonitor` también genera mensajes de estado y funcionamiento generales del *gateway* para ser procesados por el repositorio.

`awsmessenger`, también a una frecuencia configurable, busca mensajes sin enviar en la base de datos local, y en caso de encontrarlos intenta transferirlos al repositorio universal. Si por algún motivo el envío falla (pérdida de conexión, errores o falta de disponibilidad del repositorio), se registra el resultado y se reintenta luego de transcurrido un tiempo configurable (por omisión, 60 segundos). El servicio es también responsable de proveer la identidad correspondiente a la estación y las credenciales que habilitan que la información sea almacenada.

Los datos que resultan de la transferencia hacia el repositorio (exitosos o fallidos) son registrados con propósitos de depuración y monitoreo. `awsmessenger` también es responsable de eliminar datos ya enviados del almacenamiento local, para evitar el desbordamiento del espacio disponible.

En función de brindar soporte a múltiples plataformas, la información de WiFi Logger se almacena localmente y se envía al repositorio sin procesar. Es responsabilidad del repositorio contar con el *driver* adecuado que procese la información y genere las mediciones en un formato apropiado.

Se traslada esta responsabilidad al repositorio dado su mayor poder de procesamiento y simplicidad de acceso y actualización. También de esta manera se

simplifica el software en el *gateway* para obtener un sistema más robusto que requiera menos mantenimiento.

4.3. Administración de *gateways*

Durante la evaluación de la solución se establecieron un conjunto de requerimientos relacionados a los *gateways*: Soporte para múltiples tipos de estaciones, recolección de datos local, y actualización OTA.

Una vez el *gateway* se encuentre instalado en el sitio, se encontrará detrás de una red privada e inaccesible desde el exterior. Por las limitaciones de recursos en los sitios de instalación, no es posible habilitar un servicio que permita el acceso externo. Por este motivo, cada *gateway* cuenta con un servicio que establece una conexión SSH reverso [3] con un servicio disponible en Internet. A través de este enlace SSH es posible acceder al sistema operativo del *gateway* y realizar tareas de mantenimiento.

Para automatizar la instalación y actualización en los *gateways* se implementó una plataforma construida sobre *playbooks* Ansible y scripts Bash. La plataforma ejecuta en una máquina *host* e interactúa con los *gateways* a través de los túneles SSH. Provee comandos para realizar el monitoreo de todos los *gateways*, actualización de `awsmessenger` y `awsmonitor`, habilitación y deshabilitación de servicios, y actualización del sistema operativo, entre otros.

4.4. Servicios en Internet

Detrás de un dominio de Internet se accede al servicio de repositorio de mediciones y al *frontend* de visualizaciones. Los servicios se encuentran desplegados dentro de contenedores Docker. Los contenedores que forman parte del sistema son:

- Un servicio Nginx [8] que mapea solicitudes HTTP dentro del dominio hacia el repositorio o hacia el *frontend* web según corresponda.
- El servicio de repositorio universal que forma parte de este trabajo.
- El servicio de base de datos MariaDB que utiliza el repositorio.
- El servidor por omisión que utilizan las aplicaciones NextJS utilizado en la implementación del *frontend*. Es un servicio liviano Node.js.

Los contenedores inician como servicios del sistema operativo Linux subyacente.

4.5. Repositorio

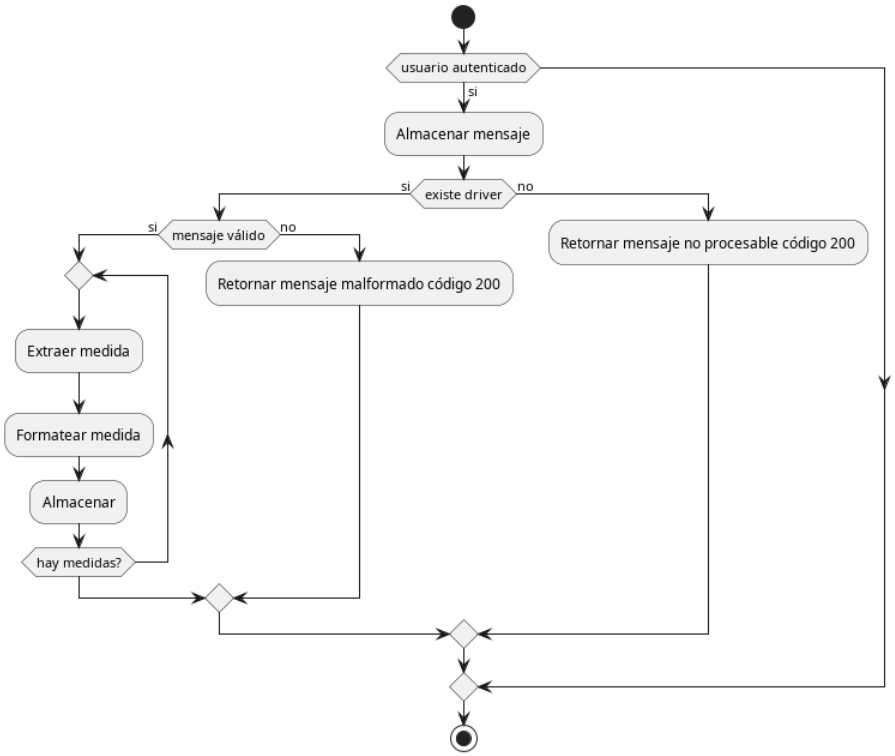
El repositorio posee una interfaz REST. Sus *endpoints* se distribuyen en dos categorías: *endpoints* de carga y *endpoints* de consulta.

Los *endpoints* de carga permiten a los *gateways* y otras fuentes enviar mensajes con las mediciones de las estaciones. Estas comunicaciones son autenticadas y autorizadas previamente, y se registran con propósitos de auditoría y monitoreo.

El repositorio tiene cargado previamente la lista de fuentes de información admitidas (incluyendo los *gateways*), con sus credenciales e identificación del *driver* que procesa los mensajes.

Una vez reconocida la fuente e identificado el *driver* de un mensaje, se lo procesa y convierte en mediciones con un formato común. Las distintas mediciones pueden requerir procesamientos particulares que son considerados por el *driver*.

Un ejemplo de procesamiento particular es el campo de temperatura máxima en los mensajes de una estación Davis. Cada mensaje cuenta con un valor de temperatura máxima del día recuperado cada 60 segundos. Sin embargo, el último mensaje del día contendrá efectivamente la temperatura máxima registrada por la estación. No se almacena un dato de temperatura máxima cada vez que se recibe el mensaje, sino que actualiza el valor de temperatura máxima, generando un único valor de temperatura máxima diario.



Los *endpoints* de consulta tienen como propósito responder a solicitudes de un cliente. Estas solicitudes se entregan en formato JSON. El cliente por omisión de los *endpoints* de consulta es el *frontend* web que provee visualizaciones. Sin

embargo, estos *endpoints* pueden ser consultados por procesos automatizados o generadores de informes.

Finalmente, los *endpoints* también proveen información detallada sobre las estaciones.

4.6. Interfaz web

La última pieza del desarrollo es una Interfaz de Usuario (UI, del inglés *User Interface*). La UI corre en el navegador del usuario, recuperando la información desde los *endpoints* de consulta del repositorio. La interfaz se diseña de modo tal que la renderización de las visualizaciones no utilice recursos del repositorio, sino que se concreten utilizando el dispositivo del usuario.

La interfaz debe permitir visualizar las mediciones de las estaciones en modo tabular, y también de forma gráfica para analizar tendencias. También muestra la localización de las estaciones que alimentan al repositorio y detalles sobre ella, como últimas medidas, geolocalización y tiempo desde la última comunicación.

5. Evaluación

La evaluación del diseño se realiza en el marco del proyecto Impactar Desafío 44 del ex Ministerio de Ciencia y Tecnología. En este proyecto, el Ministerio de Desarrollo Agrario de la Provincia de Buenos Aires (MDA), en conjunto con el Centro de Investigaciones del Mar y la Atmósfera (CIMA), buscan integrar su ecosistema de EMA junto con estaciones de otros organismos como el Instituto Nacional de Tecnología Agropecuaria (INTA) y la Bolsa de Cereales de Bahía Blanca.

Se instalaron los prototipos de *gateway* en 9 estaciones Davis distribuidas en la Provincia de Buenos Aires: Barrow (figura 1), Bellocq, Chascomús, Corfo, Miramar, Napostá, Patagones, Rausch, y UNS. Un prototipo del repositorio universal se pone a disposición en la dirección mdaimpactar.cs.uns.edu.ar como parte del proceso de prueba, hasta contar con un ambiente de producción adecuado en la infraestructura de la provincia.

Para el desarrollo de la solución experimental se contó con un ingeniero de software *senior* liderando dos pasantes del Departamento de Ciencias e Ingeniería (DCIC) de la Universidad Nacional del Sur (UNS).

La primera fase del proyecto consistió en reemplazar y reparar las estaciones existentes. Las EMA, por razones obvias, se encuentran expuestas al medioambiente, no sólo sujetas a condiciones climáticas, sino también a animales domésticos y salvajes. Las chacras tienen características heterogéneas en cuanto a disponibilidad y estabilidad de alimentación eléctrica y conectividad.

Un problema lateral surgió de la dificultad de establecer las expectativas de los usuarios como parte del proceso de desarrollo. Al resolver en el marco del proyecto Impactar la reparación de las estaciones y la instalación de los *gateways*, los usuarios y algunos miembros del equipo consideraron válidos y listos los datos recolectados. Sin embargo, los distintos artefactos de software se

encontraban aún en estado de prueba y desarrollo y esa información no resultaba confiable, generando trabajo extra posterior de validación y procesamiento.

El diseño se evalúa siguiendo los requisitos establecidos en la sección 3. Se logra con éxito mantener los enlaces y modos de operación utilizados, independiente del estado del *gateway*. La información está presente tanto en el repositorio como en el sitio del fabricante y las consolas locales a las estaciones.

El límite establecido en los *gateways* para el tamaño de la base de datos local se estableció en 5 Gigabytes. Cuatro meses de mensajes con sus registros de envíos requiere alrededor de 400 Megabytes para la base de datos SQLite. A pesar de las limitaciones de los *gateways* con respecto a procesamiento y velocidad de acceso a memoria secundaria (una microSD), las inserciones y consultas tienen tiempos adecuados para responder a los requisitos de `awsmessenger` y `awsmonitor`. El objetivo de almacenar datos en caso de problemas de acceso al repositorio resultó superior a lo necesario.

Los *gateways* fueron configurados para muestrear una vez por minuto, para un total de 1.440 mensajes diarios de cada estación. Un mensaje contiene 71 mediciones, para un total diario de 102.240 registros por día por estación. Esto presenta dos consideraciones. Primero, el tamaño de la base de datos, que en cuatro meses de operación llegó a los 6 Gigabytes aproximadamente. Con una infraestructura adecuada este tamaño es manejable tanto para servicios Cloud como *On-premise*.

Figura 1. Instalación Barrow

Sin embargo, estos números presentan desafíos a las consultas de información. Millones de entradas se acumulan en pocos meses. Afortunadamente, como se analizó en un principio, las inserciones de datos son infrecuentes y requieren pocos recursos en comparación a las consultas. Por esta razón, pueden incorporarse los índices necesarios a la base de datos sin temor a que una inserción lenta comprometa el desempeño del sistema.

Una vez operativos los prototipos, la inestabilidad de conectividad observada en las chacras generó que la información de uno o más días se acumulara en la base de datos local para luego ser transmitida al repositorio. Los pasos que implica la transferencia de un mensaje hacia un *endpoint* REST en el repositorio requiere aproximadamente un segundo, limitando la tasa de transferencia a una frecuencia de 1 Hz. Por lo tanto para transferir los 1440 mensajes de un día se requieren al menos 24 minutos. A pesar de que este tiempo es aceptable, se implementó un *script* que permite cargar los datos desde una base SQLite local al repositorio. De esa forma resulta factible copiar una base de un *gateway* al servidor donde se aloja el repositorio y cargar los mensajes directamente sin necesidad de establecer la conexión.

Las predicciones de simplicidad y robustez en los servicios de los *gateways* resultó acertada. Sólo se manifestó un error en el caso de que la estación deje de funcionar con el servicio `awsmonitor`, y un error para `awsmessenger` cuando el mensaje almacenado `awsmonitor` está vacío o corrupto en alguna forma.

El repositorio de mensajes, implementado utilizando el *framework* Django para el lenguaje Python, es de fácil mantenimiento y se pueden incorporar los *drivers* de otras fuentes con un esfuerzo mínimo de desarrollo.

Estas dos consideraciones simplifican el mantenimiento para facilitar el crecimiento de la red a nuevos servicios y estaciones.

Con respecto a las visualizaciones, resulta difícil como parte del proceso de desarrollo predecir cuáles resultan más atractivas para los usuarios. El formato tabular obviamente será el más utilizado, pero las visualizaciones gráficas requieren una mejor realimentación por parte de investigadores y meteorólogos.

6. Trabajo futuro

Con respecto a la robustez de la recolección de datos, resulta importante proveer de fuentes de alimentación de emergencia para proteger a la estación y su *gateway* de los cortes de energía eléctrica. Las estaciones cuentan con paneles solares y acumuladores y siguen muestreando, por lo que si el *gateway* pudiera continuar funcionando, esa información podría ser conservada.

Sin embargo esto plantea otros problemas, como es el mantenimiento de las UPS. El *gateway* debería incorporar el monitoreo de la batería para que el repositorio pueda generar informes adecuados.

Durante las pruebas, un error en el despliegue en el ambiente de producción produjo la pérdida total de la base de datos del repositorio. La información perdida pudo recuperarse a partir de los *gateways*. Sin embargo, resulta importante

implementar un mecanismo de copias de respaldo periódicas y rotativo de la base de datos.

7. Conclusiones

Este trabajo demuestra que la implementación de un repositorio centralizado donde almacenar datos de EMA resulta viable técnicamente.

El desafío principal del proyecto es el mantenimiento. Es necesario contar con una inversión continua en recursos: Personal que realice mantenimiento de las EMA y los *gateways*; una plataforma para dar soporte a los servicios expuestos a Internet (conexiones SSH para los *gateways*, servicio repositorio, *frontend* web); y un equipo de desarrollo que realice el mantenimiento del sistema.

Mantenimiento del sistema refiere no sólo a corregir errores y mantener las dependencias actualizadas, sino también a incorporar nuevos *drivers* para otras plataformas, o realizar desarrollos de sistemas similares a los *gateways*, pero para otros servicios que responden a requerimientos diferentes.

Referencias

1. https://icaa.gov.ar/
2. https://clima.sanluis.gob.ar/
3. Barrett, D.J., Silverman, R.E., Byrnes, R.G.: SSH, The Secure Shell: The Definitive Guide: The Definitive Guide. .ºReilly Media, Inc."(2005)
4. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS Quarterly **28**(1), 75–105 (2004), http://www.jstor.org/stable/25148625
5. Khan, W., Kumar, T., Zhang, C., Raj, K., Roy, A.M., Luo, B.: Sql and nosql database software architecture performance analysis and assessments—a systematic literature review. Big Data and Cognitive Computing **7**(2), 97 (2023)
6. Lagouvardos, K., Kotroni, V., Bezes, A., Koletsis, I., Kopania, T., Lykoudis, S., Mazarakis, N., Papagiannaki, K., Vougioukas, S.: The automatic weather stations noann network of the national observatory of athens: Operation and database. Geoscience Data Journal **4**(1), 4–16 (2017)
7. Rautmare, S., Bhalerao, D.M.: Mysql and nosql database comparison for iot application. In: 2016 IEEE International Conference on Advances in Computer Applications (ICACA). pp. 235–238 (2016). https://doi.org/10.1109/ICACA.2016.7887957
8. Reese, W.: Nginx: the high-performance web server and reverse proxy. Linux Journal **2008**(173), 2 (2008)
9. Vaishnavi, V., Kuechler, W.: Design Science Research Methods and Patterns: Innovating Information and Communication Technology, 2nd Edition (2nd ed.), p. 416 (2015). https://doi.org/https://doi.org/10.1201/b18448
10. Winand, M.: SQL Performance Explained: Everything Developers Need to Know about SQL Performance. M. Winand (2012)
11. Zachariassen, J.: A review of the forest service remote automated weather station (raws) network (2003)