UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

ÁREA TÉCNICA

TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Diseño e implementación de una arquitectura para el almacenamiento y análisis de datos obtenidos de una red de sensores para la recolección de variables medioambientales.

TRABAJO DE TITULACIÓN

AUTORES: Figueroa Sarmiento, Byron Rafael

Quiñonez Cuenca, Felipe David

DIRECTOR: Quiñones Cuenca, Manuel Fernando, Ing.

LOJA – ECUADOR

2017

Septiembre, 2017

APROBACIÓN DEL DIRECTOR DEL TRABAJO DE TITULACIÓN

Ingeniero.
Manuel Fernando Quiñones Cuenca

DOCENTE DE LA TITULACIÓN

De nuestra consideración:

El presente trabajo de titulación: "*Diseño e Implementación de una Arquitectura para el almacenamiento y análisis de datos obtenidos de una red de sensores para la recolección de variables medioambientales",* realizado por: Figueroa Sarmiento, Byron Rafael y Quiñonez Cuenca, Felipe David, ha sido orientado y revisado durante su ejecución, por cuanto se aprueba la presentación del mismo.

Loja, 8 de marzo de 2017

f). . . . . .
Ing. Quiñones Cuenca, Manuel Fernando
C.I: 1104032063

DECLARACIÓN DE AUTORÍA Y CESIÓN DE DERECHOS

"Nosotros, Figueroa Sarmiento Byron Rafael y Quiñonez Cuenca Felipe David, declaramos ser autores del presente trabajo de titulación: Diseño e Implementación de una Arquitectura para el almacenamiento y análisis de datos obtenidos de una red de sensores para la recolección de variables medioambientales, de la Titulación: Ingeniería en Sistemas Informáticos y Computación, siendo: Manuel Fernando Quiñones Cuenca, director del presente trabajo; y eximo expresamente a la Universidad Técnica Particular de Loja y a sus representantes legales de posibles reclamos o acciones legales. Además, certifico que las ideas, conceptos, procedimientos y resultados vertidos en el presente trabajo investigativo, son de nuestra exclusiva responsabilidad.

Adicionalmente declaro conocer y aceptar la disposición del Art. 88 del Estatuto Orgánico de la Universidad Técnica Particular de Loja que en su parte pertinente textualmente dice: "Forman parte del patrimonio de la Universidad la propiedad intelectual de investigaciones, trabajos científicos o técnicos y tesis de grado que se realicen a través, o con el apoyo financiero, académico o institucional (operativo) de la Universidad"

f.....
Autor: Figueroa Sarmiento, Byron Rafael
Cédula: 1104202773

f.....
Autor: Quiñonez Cuenca, Felipe David
Cédula: 1104575012

DEDICATORIA

Esta tesis quiero dedicar a mi Padre Byron la persona que siempre ha creído en mí, el cual me ha brindado su apoyo y por ser un ejemplo de perseverancia y constancia, que me ha permitido ser una persona de bien, a mi Madre Paquita, por sus consejos, sus valores, pero más que nada, por su amor, a mis hermanos por estar siempre presentes, los quiero mucho, y a todos aquellos amigos que hice en el trascurso de la carrera por compartir los buenos y malos momentos, y a todos aquellos que participaron directa o indirectamente en la elaboración de esta tesis.

Byron R. Figueroa

En primer lugar, quiero dedicar este trabajo a Dios que siempre está conmigo, por darme la vida y ser la luz que alumbra mi camino. Así también, dedico este logro a mis padres, Elvia Cuenca y José Quiñonez, por brindarme su apoyo incondicional para cumplir mis metas y ser un ejemplo claro de honestidad y trabajo. Por otro lado, lo dedico a mi esposa y queridos hijos Cristina y Miguel, por ser las personas quienes son mi inspiración para continuar cada día en la búsqueda de ser buenas personas, y contribuir con la sociedad.

Felipe David Q.

AGRADECIMIENTO

Agradecemos a Dios, en primer lugar, por permitirnos concluir esta importante etapa de nuestra existencia y alcanzar nuestros nobles ideales; a la Universidad Técnica Particular de Loja, quien nos abrió las puertas para realizar nuestros estudios de pregrado. A las dignas autoridades de la institución por apoyarnos con sabiduría en el proceso de formación personal y académica. A los docentes por enseñarnos a ser personas buenas y honorables, y en especial al Ing. Manuel Quiñones, quien nos asesoró con pertinencia y afecto en el desarrollo del presente trabajo de fin de titulación. Adicionalmente al equipo de trabajo conformado por Ing. Santiago Quiñones, Ing. Víctor González, e Ing. Max Peralta, miembros de los Departamentos de Geología y Minas, Ingeniería Civil, Ciencias de la Computación, y Electrónica de la UTPL, quienes contribuyeron con la ejecución del proyecto. A nuestros padres y familiares por apoyarnos incondicionalmente, especialmente en los duros momentos. Finalmente, a todos en general por sus buenos deseos, para alcanzar nuestras metas personales.

Los autores

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente trabajo surge de la necesidad de implementar un sistema para el almacenamiento y análisis de datos obtenidos de una red de sensores para la recolección de variables medioambientales de la UTPL, que se encuentran desplegados en varios puntos geográficos de las provincias de Loja y El Oro. Para cubrir los requerimientos de la solución se procedió a analizar el estado de arte del problema, abordando el paradigma de Internet de las Cosas como tecnología emergente de solución. Luego, se procedió a diseñar la arquitectura, a través de la selección de tecnologías de software basadas en código abierto, considerando la interoperabilidad de los dispositivos y protocolos. Posteriormente, se implementó una interfaz web para la administración, visualización-análisis de los datos, y el envío de alertas; a través del uso de tecnologías como *Responsive-Web-Design*, *AngularJS*, Html5, *RabbitMQ Broker*, y MQTT como protocolo de comunicación entre los sensores y el sistema de almacenamiento de los datos *MongoDB* y *NodeJS*. Finalmente, se adaptó la solución al estándar SOS, el mismo que define una interfaz estandarizada para el acceso a información desde agentes externos.

Palabras Clave: Red de sensores inalámbricos, variables medioambientales; Arquitectura para almacenamiento y análisis de datos; Internet de las Cosas; Tecnologías: *Responsive Web Design AngularJS*, *Bower*, *Bootstrap*, Css3, Html5, *MongoDB*, *NodeJS*, MQTT, *RabbitMQ Broker*, SOS.

ABSTRACT

This research arose from the need of implementing a software solution for storage and analysis of data gotten from a network of environmental variables sensors, which has been deployed in several geographic points within the provinces of Loja and El Oro in Ecuador.

For the development of this research, a state of the art about storage and deployment of sensor data systems was developed addressing the paradigm of Internet of Things (IoT), as an emerging-solving technology. Then we proceeded to design the architecture through the selection of technologies based on open source software, considering the interoperability of devices and protocols. Later, a web interface for administration, display-data analysis, and sending alerts components were implemented; through the usage of technologies such as Reponsive-Web-Design AngularJS, Bower, Bootstrap, CSS3, HTML5, and MQTT as the communication protocol between the sensors and the storage data system of MongoDB and NodeJS. Finally, the solution was aligned to SOS (Sensor Observation Service), which defines a standardized interface and operations for access to data from sensors and sensor systems that is consistent for all sensor systems.

Keywords: Environmental variables, wireless sensor network, Architecture for data storage and analysis, Internet of Things, Technologies: Responsive Web Design AngularJS, Bower, Bootstrap, CSS3, Html5, MongoDB, MQTT, RabbitMQ Broker, SOS.

ACRÓNIMOS

AMQP Advanced Message Queuing Protocol
API Application Programming Interface
AWS Amazon web services
CERN Organización Europea para la investigación Nuclear
CoAP Constrained Application Protocol
CSS Cascading style sheets
CSV Comma separated values
CMS Content Management System
DUP Duplicate delivery of control packet
GHz Gigahertz
GUI Graphical user interface
HTTP Hypertext Transfer Protocol
IoT Internet of things
JDK Java Development Kit
Json JavaScript Object Notation
M2M Machine to machine
Mb Megabyte
Ml Millisecond
MPL Mozilla Public License
MQTT Message queue telemetry transport
MVC Model-view-controller
MVVM Model–view–view-model
SQL Structured Query Language
NoSql No solo SQL
OASIS Organization for the Advancement of Structured
O&M Observations and Measurements
OGC Open Geospatial Consortium
QoS Quality of Service
REST Representational State Transfer
SAVA Subsistema de Almacenamiento de variables medioambientales
SENSORML Sensor model language
SESMETER Sistema de estaciones meteorológicas
SGEMVM Subsistema de gestión de estaciones meteorológicas y variables medioambientales
SGBD Sistema de gestión de base de datos
SMS Short Message Service
SOS Sensor Observation Service
SPS Sensor Planning Service
STOMP Streaming Text Oriented Messaging Protocol
SWE Sensor Web Enablement
TBD To be defined
TCP Transmission Control Protocol
UOM Unit of Measure
URL Uniform Resource Locator
UTF-8 8-bit Unicode Transformation Format
UTPL Universidad Técnica Particular de Loja
XML Extensible markup language
XMPP Extensible Messaging and Presence Protocol

INTRODUCCIÓN

El Internet es el medio de comunicación más utilizado en las últimas décadas a nivel mundial, que ha evolucionado y cambiado la forma de vida del ser humano en las diferentes áreas del conocimiento. Dando origen a nuevos paradigmas adaptables para el desarrollo e implementación de aplicaciones de software, que dan solución a las necesidades de los usuarios de una manera inteligente, un claro ejemplo de este tipo de paradigmas es el Internet de las Cosas (IoT). En el ámbito de la presente investigación, el monitoreo del medioambiente es uno de los principales campos de aplicación de la IoT, pues permite acceder desde cualquier punto geográfico a información de sensores medioambientales de forma continua. Además, mediante el procesamiento de los datos se puede realizar la generación de alertas automáticas que coadyuven a la prevención de catástrofes. Específicamente, el presente trabajo de titulación con la temática "*Diseño e implementación de una arquitectura para el almacenamiento y análisis de datos obtenidos de una red de sensores para la recolección de variables medioambientales*" busca resolver el problema de recolectar datos en tiempo real, correspondientes a diversas variables medioambientales que son transmitidas por estaciones meteorológicas usando tecnologías inalámbricas de comunicación.

El presente trabajo de titulación, surge a partir de la necesidad de diseñar e implementar una arquitectura para el almacenamiento y análisis de datos obtenidos de una red de sensores; la misma que es administrada por la Universidad Técnica Particular de Loja (UTPL). El problema constituye la falta de un sistema que permita gestionar, visualizar y monitorear las distintas variables medioambientales en tiempo real.

Asimismo, para el desarrollo del presente trabajo de titulación plantea como base seis objetivos específicos. Primero, elaborar un estado actual de sistemas de almacenamiento y de visualización de datos de sensores, para lo cual se revisó investigaciones similares realizadas, encontrando que el tema de investigación se ajusta al paradigma de Internet de las Cosas (IoT), una de las tecnologías emergentes en la actualidad. Segundo, seleccionar tecnologías de software basadas en código abierto, el mismo que fue alcanzado tomando en cuenta que el desafío principal de la implementación de una solución IoT es la interoperabilidad de los dispositivos y los protocolos de diversos proveedores. Tercero, definir un protocolo de comunicación entre los sensores y el sistema de almacenamiento de los datos, permitiendo que nuevos sensores se integraran al sistema. Para lo cual se realizó el análisis entre los recursos de hardware de la red y los requerimientos a ser alcanzados; consecuentemente se eligió MQTT,

por tratarse de un protocolo de mensajería ligero, que utiliza un modelo publicación/suscripción, en el cual el dispositivo cliente que publica un mensaje se lo denomina publicador y los dispositivos clientes que lo reciben se los denomina subscritores; en donde la comunicación se produce a través del servidor, que es una pieza de software capaz de transmitir mensajes desde un publicador para todos los subscritores que están suscritos al tema en particular asignado al mensaje. El servidor es capaz de reenviar el mensaje y retener el último mensaje, si es necesario, sobre cada tema del publicador; a este servidor MQTT también se lo denomina bróker de mensajes. Subsiguientemente, se eligió al bróker Rabbit MQ como implementación del protocolo MQTT. Cuarto, implementar el sistema de almacenamiento de datos provenientes de los sensores medioambientales en tiempo real, el mismo que fue efectuado a través del gestor de base de datos MongoDB y un cliente implementado bajo Nodejs como tecnología óptima para procesamiento asíncrono. Quinto, implementar el estándar SOS para permitir la interoperabilidad con otros sistemas o agentes externos, dicha adaptación se la realizo utilizando el sistema 52º North. Finalmente, como sexto objetivo se desarrolló una interfaz web para la administración, visualización y análisis de los datos, así como el envío de alertas; usando los siguientes lenguajes de programación: Java, Angular, NodeJS, y CSS.

El trabajo de titulación presento algunos retos, como determinar el protocolo de IoT, ya que en la investigación se encontró que existe una gran variedad de protocolos. Sin embargo, no hay un argumento u consenso claro sobre el cual sería el ideal para cada contexto. Por otro lado, es importante mencionar que durante el desarrollo de este tema de tesis existió un gran apoyo de quienes conforman el Grupo de Investigación de Redes de Sensores Inalámbricos de la UTPL, ya que aportaron con información necesaria para abordar el proyecto y las herramientas para cumplir con el mismo.

Finalmente, el desarrollo del presente trabajo de titulación se lo ha organizado en cuatro capítulos. En el primer capítulo, denominado *Estado de Arte*, se ha realizado un compendio teórico con el cual se ha determinado la forma como ha sido tratado el tema en aplicaciones similares; lo que nos permitió determinar cómo se encontraba el avance del conocimiento en el momento de realizar la investigación y cuáles eran las tendencias existentes, para el desarrollo de la problemática que se iba a llevar a cabo. En el segundo capítulo, titulado *Definición del Marco de Trabajo,* se ha realizado un esquema teórico acerca de las tecnologías y herramientas actuales disponibles en el medio, para determinar en la próxima etapa cuales de ellas nos ayudarían a cubrir los requerimientos del presente proyecto de desarrollo de software. En el tercer capítulo,

nombrado *Desarrollo de la Solución,* se ha documentado qué tecnologías han sido utilizadas, logrando de esta manera definir una arquitectura que cubran los requerimientos planteados, y brinden las características básicas de seguridad, escalabilidad, y robustez. Finalmente, en el cuarto capítulo *Pruebas,* se ha procedido a realizar una evaluación de la solución desarrollada. Luego de haber realizado la etapa de análisis y desarrollo es importante hacer pruebas que aseguren la calidad de software. Teniendo como objetivo hacer la retroalimentación, a través del uso de herramientas para medir niveles máximos de rendimiento, para determinar si la solución desarrollada cubre los requerimientos del usuario a cabalidad.

OBJETIVOS

Objetivo General:

Diseñar e implementar una arquitectura para el almacenamiento y análisis de datos obtenidos de una red de sensores para la recolección de variables medioambientales.

Objetivos Específicos

- Elaborar un estado actual de sistemas de almacenamiento y de visualización de datos de sensores.
- Seleccionar tecnologías de software basadas en código abierto.
- Definir un protocolo de comunicación entre los sensores y el sistema de almacenamiento de los datos.
- Implementar el sistema de almacenamiento de datos provenientes de los sensores.
- Implementar el estándar SOS para el almacenamiento de información provenientes de las estaciones meteorológicas.
- Desarrollar una interfaz web para la administración, envío de alertas, visualización y análisis de los datos.

CAPÍTULO I: ESTADO DEL ARTE

1.1 Introducción

En este capítulo, denominado Estado de Arte, se ha realizado un compendio teórico dónde se ha determinado como ha sido tratado el tema en aplicaciones similares, y cuáles eran las tendencias existentes, para el desarrollo de la problemática.

1.2 Internet de las cosas, tecnología emergente en la web

El surgimiento y evolución de la Internet como red masiva de redes, ha determinado su importancia para la sociedad en la actualidad, posicionándola como principal medio de comunicación. Actualmente Internet ofrece una infraestructura de red que conecta a millones de computadores en todo el mundo, formando una red poderosa de gran alcance (Castells, 2011). En otras palabras, la constante evolución de la Internet ha cambiado el estilo de vida del ser humano en las últimas décadas, dando origen a nuevos paradigmas adaptables para el desarrollo e implementación de aplicaciones de software, que compensen las necesidades de los usuarios de una manera inteligente.

La WEB, definida como la forma de acceder a la información sobre el medio de la Internet, ha evolucionado hasta la actualidad en tres etapas. En su primera etapa (1989-1999) categorizada como WEB 1.0, los usuarios se conectaban a la red simplemente con el propósito de consumir información. Luego, gracias a la integración de bases de datos y gestores de contenidos (CMS), el internet evolucionó a su siguiente etapa denominada Web 2.0 (a mediados del 2004), permitiendo a los usuarios ser creadores de contenidos en una web social o colaborativa. Posteriormente, surgió la WEB 3.0, considerada como la que combinará distintos factores como el contenido semántico, inteligencia artificial, inteligencia colectiva y gestión del conocimiento.

Seguidamente, considerando que estamos atravesando el inicio de la WEB 3.0, y proyectándonos al futuro, se han creado diversos paradigmas y herramientas de software y hardware, con el objetivo de crear una web más inteligente, capaz de cubrir las necesidades del usuario de una manera automatizada, incluso antes de solicitarlo. En la actualidad los usuarios pueden conectarse a la red a través de diversos dispositivos electrónicos, y no únicamente computacionales como laptops, teléfonos inteligentes o *tablets*, sino también con electrodomésticos como televisores, refrigeradores, lavadoras, e incluso con automóviles. Es entonces cuando surge un nuevo paradigma denominado *Internet of Things (IoT)* o en español *Internet de las Cosas*, en el cual los objetos son capaces de conectarse a la red, enviar y/o recibir información en tiempo real, la cual puede ser analizada e interpretada para responder con eventos

a través de actuadores.

El IoT, una tecnología emergente, ha cobrado importancia en los últimos años, y se estima que alcanzará la madurez en el mercado entre los 5 a 10 próximos años. Debido a la infraestructura tecnológica que proporciona IoT, es posible crear y desplegar muchas aplicaciones novedosas que mejoren la calidad de vida de las personas en diferentes ámbitos como la medicina, industria automovilística, mejora de servicios de gobiernos locales, monitoreo del medioambiente, entre otras aplicaciones. Por tal motivo, de acuerdo a Gartner (2016), en el ciclo sobre expectativas de tecnologías emergentes, los próximos dos años serán cruciales para las empresas debido a que deben identificar como aprovechar IoT, y con esto evitar quedarse rezagados en el pasado.

Particularmente, en el ámbito de la presente investigación, el monitorio medioambiental ha sido uno de los principales campos de aplicación de la IoT, pues permite acceder desde cualquier parte a información de sensores medioambientales. Esta información en tiempo real ha sido una de las necesidades urgentes, la misma que puede ser analizada e interpretada, para adquirir nuevos conocimientos sobre el cambio climático que permitan formular estrategias para mitigar sus efectos y conservación del medio ambiente.

1.3 Requerimientos de la IoT

En el contexto del IoT, se señala que los objetos conectados poseen inteligencia, lo que no significa necesariamente que la inteligencia se encuentra en el objeto. De acuerdo a Meyer (2009), se pueden identificar dos tendencias a través de la red, o en el objeto. Dentro del contexto de la primera tendencia, *a través de la red,* la inteligencia no se encuentra dentro del objeto físico, sino fuera de él, donde el objeto se encuentra vinculado a un agente-servidor dedicado, encargado de gestionar su funcionamiento basándose en la información obtenida, a través del objeto físico o algún patrón de comportamiento. En este tipo de sistemas la inteligencia del objeto se ejecuta en hosts que suelen denominarse plataformas de portal. En cambio, en el contexto de la segunda tendencia, cuando la inteligencia se encuentra *en el objeto*, este no solo se encarga de recopilar y tratar la información, sino que también es el encargado de realizar la toma de decisiones. Es decir, para poseer inteligencia un objeto debe contar necesariamente con cierta capacidad de computación y/o almacenamiento. Así también es importante recalcar, que algunas soluciones pueden ser intermedias, lo que significa que se pueden diseñar arquitecturas donde se aplique los dos enfoques.

Por otro lado, según el artículo científico *Futuras Generaciones de Sistemas Computacionales*,

titulado *IoT: visión, elementos, y futuras direcciones* (2013), el Internet de las Cosas demanda de tres aspectos importantes para lograr la conectividad inteligente y computación sensible. Primero, análisis y visión de la situación de los usuarios y sus dispositivos. Segundo, arquitecturas de software y redes de comunicación generalizados para procesar y transmitir la información contextual en tiempo real, crítico para su posterior procesamiento. Finalmente, herramientas de análisis en la IoT, que tienen como objetivo lograr en los dispositivos un comportamiento autónomo e inteligente sin la intervención del usuario.

De lo anteriormente mencionado, se puede señalar que la presente investigación está orientada a cubrir específicamente el segundo aspecto, correspondiente al diseño de una arquitectura de software robusta que permita el correcto almacenamiento de datos recolectados de una WSN, cuyo propósito está orientado al análisis e interpretación de información correspondiente a medidas de variables medioambientales como presión atmosférica, velocidad de viento, humedad relativa, etc.

1.4 Estructura y componentes de la IoT

De acuerdo a Lu Zeyong & Jun (2011), la implementación de soluciones orientadas por el paradigma de la IoT, deben estar estructuradas en tres capas. La primera es la *capa de reconocimiento,* cuya función es interconectar los objetos inteligentes con el internet. Mientras que la segunda es la *capa de red,* responsable de transmitir, procesar y clasificar la información recibida de la capa de reconocimiento. Por último, la *capa de aplicaciones,* en la que se localizan los servicios que permiten interactuar con la información recibida de las capas inferiores. De las cuales, la capa de red es la principal debido a que es la que provee los servicios en la IoT y se constituye en el enlace entre la capa de reconocimiento y de aplicaciones (Solano, 2013).

Mientras que por otro lado, en el artículo *IoT: visión, elementos, y futuras direcciones* (2013) se menciona que existen tres componentes IoT: (a) *hardware,* constituido por sensores, actuadores y hardware de comunicaciones embebido; (b) *middleware* establecido por herramientas de software que permitan el almacenamiento masivo en tiempo real y su procesamiento para el análisis de datos de acuerdo a patrones; (c) *presentación* determinado por herramientas de software que permitan la visualización e interpretación de la información, accesibles en las diferentes plataformas y diseñadas para diferentes aplicaciones.

Consecuentemente, se puede evidenciar que existe compatibilidad entre la estructura y los elementos de la IoT, como se representa en figura 1.

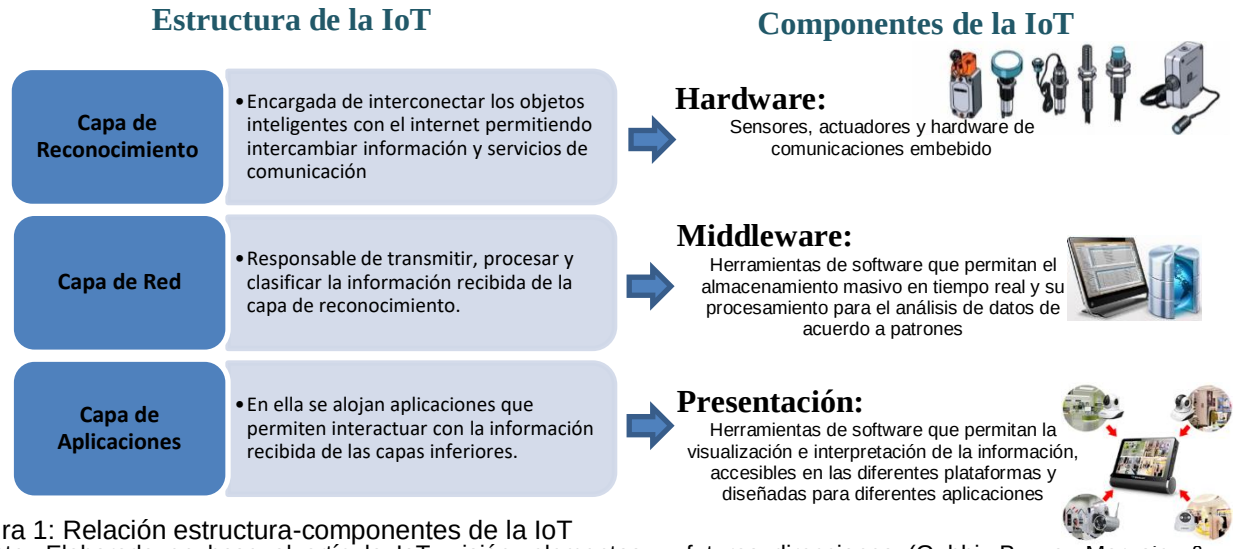


Figura 1: Relación estructura-componentes de la IoT
Fuente: Elaborada en base al artículo IoT: visión, elementos, y futuras direcciones (Gubbi, Buyya, Marusic, & Palaniswami, 2013)

1.5 Estándares y protocolos de IoT

Según Rodríguez (2013), para poder implementar la nueva era tecnológica de la IoT en la red actual de Internet, será necesario resolver el problema asociado a la concurrencia de un número muy elevado de dispositivos de naturaleza heterogénea sobre la Internet; se estima una futura existencia de 7 trillones de dispositivos por 7 billones de personas. Por otro lado, para el diseño e implementación de una solución IoT algunos aspectos claves deben ser considerados, como lograr la interoperabilidad entre los dispositivos interconectados, encontrar el mecanismo para dar a los objetos perspicacia al permitir su comportamiento autónomo, y garantizar la seguridad de la información para inspirar confianza en los usuarios (IOTpreneur, 2016).

De acuerdo a la revista telemática titulada *Arquitectura y Gestión de la IoT* (2013), la IoT se encuentra actualmente muy fragmentada y desarrollada de forma vertical, por sectores independientes de la industria (medicina, hogar, producción, transporte, etc), cada área específica tiene una arquitectura de la IoT diferente. Consecuentemente, existe un gran número de arquitecturas y propuestas de gestión, incluyendo diferentes tecnologías, protocolos y estándares, como se puede evidenciar en la tabla 1.

Tabla 1: Resumen de las principales organizaciones, proyectos, protocolos y arquitecturas propuestas correspondiente al proceso de estandarización de la IoT

Organizaciones	Proyectos	Plataformas	Protocolos
Comité Técnico M2M ETSI	FlexWare	Interdigital	SCADA
3GPP TSG Service & System Aspects	SIRENA	Orange Labs Digital Home	Modbus
TIA TR-50 Comité de Ingeniería	SODA	EDF R&D Smart Gris Experimental Platform	UPnP
IETF	UseNEt	Wuxi Instituto de las cosas (China)	NAT-PMP
Global ICT, fórum de estandarización de la India	EXALTED	Trangrarn (Japón)	DPWS
ITU	AIM	HP CenSe	CoAP
Open Mobile Alliance	Esense	EMMON	6LoWPAN
DMTF, M2M Standarization Task	OpenWSN	Alcatel Home Sensor Gateway Prototype	SOAP
W3C	Berkeley's Local	AIM Gateway Prototype	MQTT
OMG	Intel Iris Net	BOSP	JXTA
EPCGlobal	Berkeley's WEBS	Intel Framework	Zigbee

Fuente: Tomado de la revista telemática titulada Arquitectura y Gestión de la IoT (2013)

Los estándares juegan un papel importante al momento de formar IoT para permitir a todos los actores el acceso y uso equitativo; únicamente a través del desarrollo de estándares y su coordinación promoverá el desarrollo eficiente de aplicaciones, infraestructura, servicios y dispositivos para IoT. Consecuentemente, varios organismos de estandarización, privados y de investigación están involucrados en las actividades de desarrollo de soluciones para cumplir con los requisitos tecnológicos para la IoT.

De acuerdo a Chen S, *et al* (2014), el proceso de desarrollo de estándares deberá ser abierto, libre y accesible para el público; por lo tanto, la estandarización para IoT es relativamente complicada, ya que se puede incluir una amplia gama de diferentes estándares, como estándares de arquitectura, protocolos de comunicación, procesamiento de la información, seguridad, y datos.

1.6 Trabajos relacionados al problema

Cuando se realizó la investigación, sobre trabajos relacionados al presente proyecto de titulación, se determinó que son escasos los trabajos enfocados a la implementación de una arquitectura para el almacenamiento de variables medioambientales. A continuación, se describe los pro y contras que ofrecen al público tres plataformas que actualmente se encuentran disponibles en el mercado de la IoT.

1.6.1 Plataforma Ubidots.

Esta plataforma está orientada a la IoT y ofrece una capacidad de almacenamiento limitada para la versión de educación, la cual es gratuita. La plataforma Ubidots da soporte al protocolo de comunicación HTTP, además permite al usuario común realizar un análisis y visualizar los datos de forma intuitiva ya que el contenido presentado esta correctamente distribuido (estructurado). El soporte ofrecido por esta plataforma es 24/7 para la versión pagada y para la versión gratuita el soporte es través de una amplia comunidad. Otra característica importante que se debe mencionar es que permite exportar la información a través del formato abierto CSV. Sin embargo, las desventajas de esta plataforma, para la solución del problema de investigación es que el protocolo de comunicación HTTP que utiliza Ubidots, generalmente no es el más recomendado para dispositivos con recursos limitados. Asimismo, la capacidad de almacenamiento en la versión gratuita es ilimitada únicamente durante tres meses, luego automáticamente la información es eliminada; lo que no permitiría realizar un estudio sobre los datos a largo plazo. Finalmente la versión gratuita permite la conexión de menos de 20 dispositivos concurrentemente, lo que limitaría de la escalabilidad de la solución a futuro (Ubidots, 2016).

1.6.2 Plataforma Amazon Web Servicies (AWS) IoT.

Amazon IoT es una plataforma que está alcanzando importancia en el medio debido a que es respaldada por la compañía estadounidense Amazon. En lo referente al almacenamiento masivo de información, esto no se realiza directamente en la plataforma, en su lugar este requerimiento se cubre a través del motor de reglas que direcciona mensajes a través del protocolo de comunicación MQTT a puntos de enlace de AWS. Entre ellos tenemos: AWS Lambda, Amazon Kinesis, Amazon S3, Amazon Machine Learning, Amazon DynamoDB, Amazon CloudWatch y Amazon Elasticsearch Service. Por otro lado, esta plataforma ofrece el servicio de análisis de datos únicamente a través de AWS Lambda, la cual tiene costos adicionales dependiendo del número de solicitudes; mientras que el servicio de visualización de datos se lo puede hacer únicamente a través de la Amazon Elasticresearch. Respecto a los costos, la capa gratuita de AWS IoT ofrece 250 000 mensajes al mes sin costo, durante los primeros 12 meses (mensajes que se toma en la medición: MQTT Connect, MQTT PubAck, MQTT Ping, suscripción MQTT, publicación MQTT). Mientras que, en la capa pagada, el precio es de 5 USD por millón de mensajes, el tamaño del mensaje consiste en un bloque de datos de 512 bytes procesados por AWS IoT, ya sean publicados como entregados por el servicio. Se puede enviar mensajes de hasta 128 KB en un bloque y se le cobrará en múltiplos de 512 bytes. Por ejemplo, una carga de

900 bytes se cuenta como dos mensajes (Amazon Web Services, 2016). La figura 2, muestra la arquitectura de la plataforma Amazon Web Services en detalle.

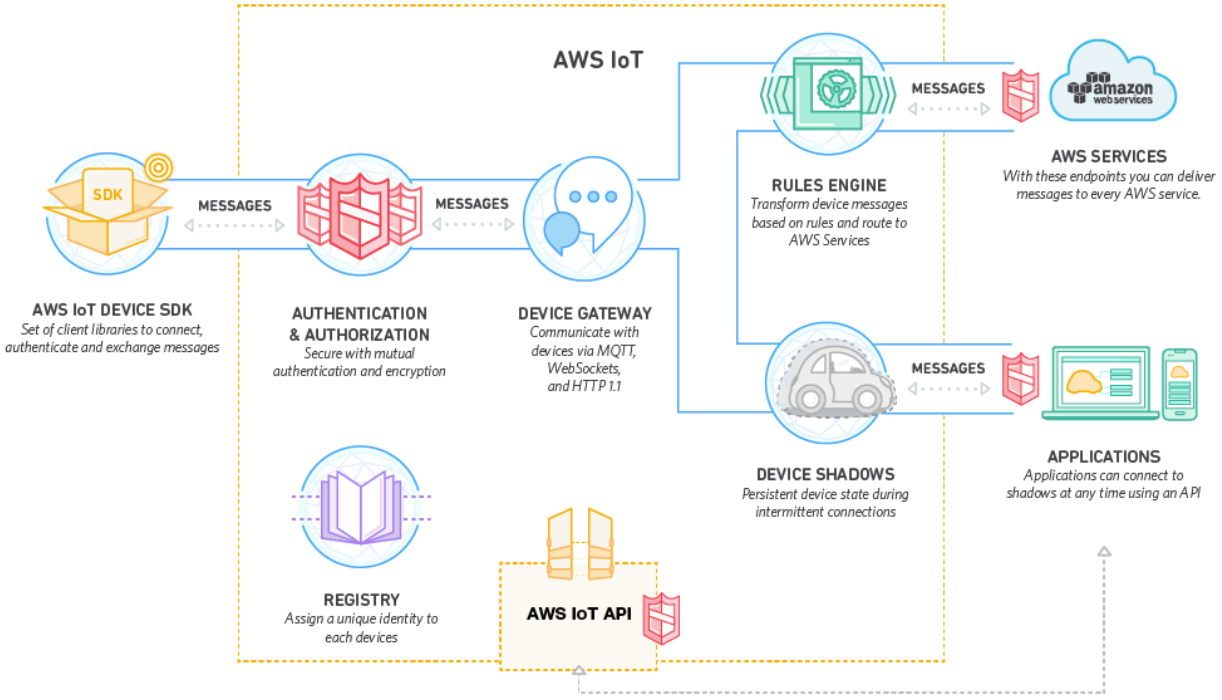


Figura 2: Arquitectura de la plataforma Amazon Web Services (AWS) IoT
Fuente: Amazon Web Services (2016)

1.6.3 Plataforma: IBM Watson Internet of Things.

La plataforma IBM Watson IoT se enfoca en la parte de *back-end* y no tanto a la parte de *front-end*, ya que su arquitectura ofrece soporte para el protocolo de publicación/suscripción MQTT y HTTP (ver figura 3). El servicio de almacenamiento de datos se lo puede hacer únicamente a través de IBM Cloudant NoSQL, un motor de base de datos optimizado que ofrece durabilidad y movilidad, perfecto para aplicaciones móviles y web de rápido crecimiento. Sin embargo, en lo referente a costos, el plan de servicio gratuito para Internet of Things Platform incluye un máximo de 20 dispositivos registrados y un máximo de 100 MB de datos intercambiados, datos analizados y datos límite analizados al mes para cada uno. Pasada estas cuotas IBM factura por cada MB adicional (IBM, 2016).

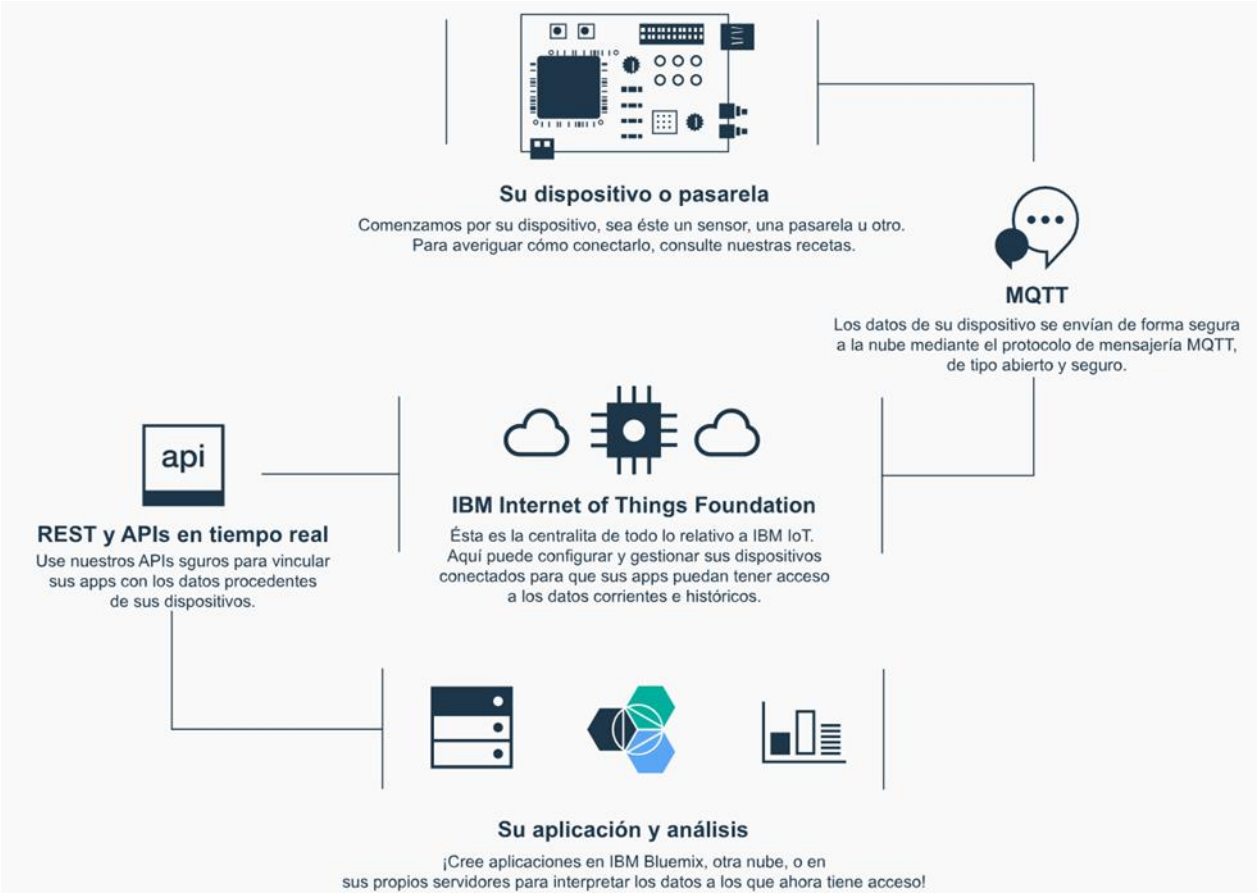


Figura 3: Arquitectura de la plataforma IBM Watson Internet of Things
Fuente: Tomado de la página: “Analyzing Data with IBM Watson Internet of Things” (Heidloff, 2016)

CAPÍTULO II: DEFINICIÓN DEL MARCO DE TRABAJO

2.1 Introducción

En este segundo capítulo, titulado “Definición del Marco de Trabajo”, se ha realizado un esquema teórico acerca de las tecnologías y herramientas actuales disponibles en el medio, para determinar en la próxima etapa cuales de ellas nos ayudarían a cubrir los requerimientos del presente proyecto de desarrollo de software.

Concretamente, en este capítulo se procederá a abordar cinco puntos importantes de la investigación: descripción de la problemática de investigación; análisis detallado del protocolo de comunicación a utilizar; estudio del estándar Sensor Observation Service (SOS) que es una interfaz estandarizada para el acceso a información desde sistemas de sensores; descripción de herramientas de software disponibles correspondientes a lenguaje de programación y gestor de datos; y descripción de la metodología de software a seguir para alcanzar los objetivos de la investigación.

2.2 Problemática

Cuando se investigó trabajos relacionados al presente proyecto de titulación, se pudo constatar que son escasos los trabajos enfocados a la implementación de una arquitectura para el almacenamiento de variables medioambientales. Una de las características que se pudo encontrar en aquellos trabajos, es la falta de estandarización para la selección de un protocolo en la capa de aplicación; por lo que era necesario indagar que estándares se ajustaban mejor a la solución de software a desarrollar, que permitiera la comunicación con las estaciones meteorológicas. Como ya se mencionó en el capítulo de Estado de Arte, la IoT se encuentra actualmente muy fragmentada y desarrollada de forma vertical, por sectores independientes de la industria y cada área específica tiene una arquitectura de la IoT diferente. Por lo que existe un gran número de tecnologías, protocolos y estándares disponibles en el medio.

Consecuentemente, el desafío de este trabajo de investigación será la correcta selección de un protocolo de comunicación, que permita consolidar una base sólida para el diseño de la arquitectura de software a implementar. El aporte que se pretende brindar con la implementación de este proyecto es proporcionar una herramienta que permita almacenar, visualizar e interactuar con los datos provenientes de estaciones meteorológicas previamente instaladas. Por otro lado, es importante mencionar que otro requerimiento de usuario, a ser cubierto es lograr la estandarización de datos a almacenarse bajo el estándar SOS de la Open Geospatial Consortium (OGC), de manera que la información generada pueda ser analizada, visualizada, y que se

encuentre disponible para ser consumida por otro agente en el futuro. Asimismo, es importante mencionar que se tiene como requerimiento la selección de herramientas de software *open source* (código abierto), que es un término que se utiliza para denominar a cierto tipo de software que se distribuye mediante una licencia que le permite al usuario final, si tiene los conocimientos necesarios, utilizar el código fuente del programa para estudiarlo, modificarlo y realizar mejoras en el mismo, pudiendo incluso hasta redistribuirlo.

2.3 Análisis de protocolos de IoT

Dentro de los objetivos planteados, se encuentra la selección de un protocolo en la capa de aplicación; para ello se ha realizado una investigación bibliográfica entre varios protocolos existentes, como CoAP, XMPP, HTTP y MQTT. Como resultado de esta investigación se determinó que el protocolo MQTT se ajusta a los requerimientos de la solución y ofrece grandes ventajas. A continuación, un detalle más profundo acerca del protocolo MQTT.

2.3.1 Protocolo MQTT.

MQTT (Transporte de Telemetría de Cola de Mensajes) es un protocolo de conectividad bidireccional (two-way) para redes inalámbricas orientado a paradigmas como "máquina a máquina" (M2M) o "Internet de las cosas" (IoT). Se trata de un protocolo de mensajería ligero que se encuentra situado en la capa de aplicación, y utiliza como base el protocolo TCP en la capa de transporte. MQTT sigue un modelo publicación/suscripción, donde hay un nodo que publica mensajes y otros nodos interesados que se "suscriben" al primer nodo para poder recibir estos mensajes. MQTT tiene como objetivos minimizar el uso del ancho de banda de red, optimizar los requerimientos de recursos del dispositivo; y asegurar un cierto nivel de fiabilidad. MQTT está enfocado a los dispositivos interconectados y aplicaciones móviles; donde el ancho de banda y la energía de la batería son limitados. Arquitectónicamente, MQTT se configura como un modelo cliente / servidor. La primera versión fue creada por Eurotech en 1999, en el año 2013 MQTT fue reconocido como un estándar certificado por la OASIS (Organization for the Advancement of Structured), y actualmente el protocolo se encuentra en la versión 3.1.1 (MQTT.ORG, 2016).

En el anexo 1, se encuentra un detalle completo del formato de mensaje MQTT, así como los niveles de calidad de servicio que ofrece este protocolo.

2.3.1.1 Paradigma publicación / suscripción en MQTT.

El protocolo MQTT sigue el paradigma basado en tópicos (temas) de publicación / suscripción, en el cual los clientes conectados a través de MQTT son capaces de comunicarse entre sí mediante el envío de mensajes. Cada mensaje tiene siempre un tópico, que puede ser una palabra clave a una cadena de caracteres. Un cliente es una pieza de software capaz de enviar y/o recibir mensajes; el cliente que publica un mensaje se lo denomina “publicador”, mientras que los clientes que reciben los mensajes se los denomina “subscritores”. El paradigma de publicación/suscripción permite a los subscritores que expresen su interés en los eventos generados por un publicador. Una vez que los subscritores expresan su interés en un evento o un patrón de eventos, son notificados cada momento que se genere un nuevo evento por el subscritor que coincide con su interés. Normalmente los suscriptores no están interesados en todos los eventos que se producen en el sistema, sólo en algunos en particular. Por lo tanto, el cliente que esté suscrito a por lo menos un tema se lo llama subscriptor (IBM, 2016).

En conclusión, en el paradigma de publicación/suscripción los clientes se comunican entre sí a través del servidor, que es un ente de software capaz de transmitir mensajes desde un publicador para todos los subscritores que estén registrados al tema en particular asignado al mensaje. Es importante mencionar que el servidor es capaz de reenviar el mensaje y retener el último mensaje (si es necesario) sobre cada tema del publicador. Al servidor MQTT también se lo conoce como bróker de mensajes.

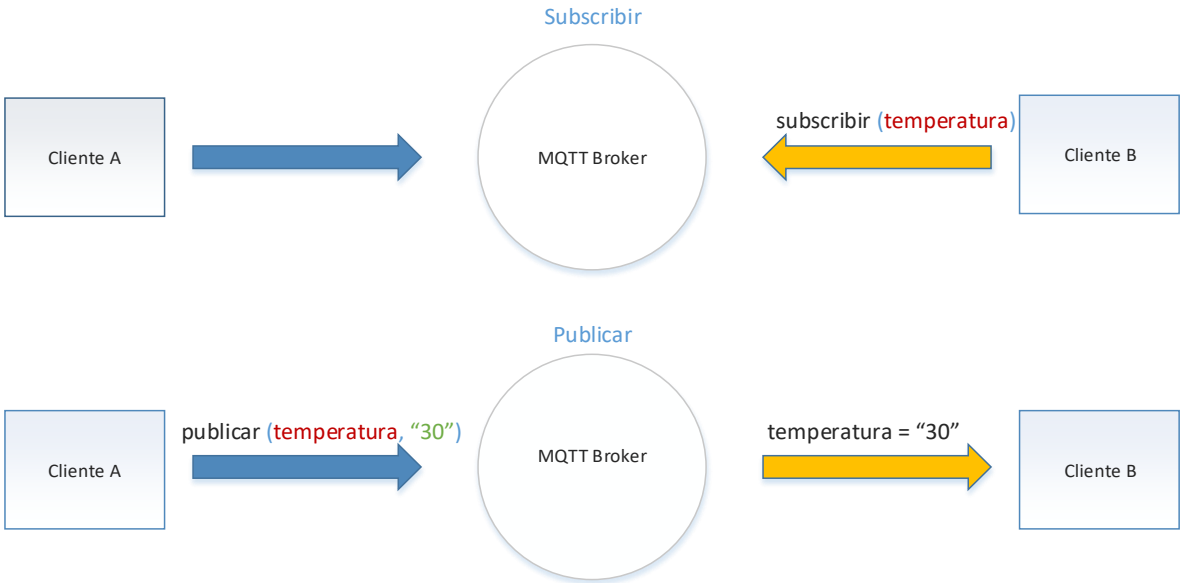


Figura 4: Paradigma publicación / suscripción en MQTT
Fuente: Elaborada por los autores

En la Figura 4, se muestra el paradigma publicación / subscripción en MQTT; en la cual se muestra una red en la que los clientes están conectados a un bróker. Primeramente, el cliente se subscribe al tópico “temperatura”. Pasado un tiempo, un cliente A publica el valor de “30” para el tópico temperatura, el bróker re-direcciona todos estos mensajes a todos los clientes que están subscritos a este tópico.

2.3.1.2 Tópicos.

Para poder enviar un mensaje es necesario el uso de un tópico o tema, y para que un cliente pueda recibir un mensaje necesita subscribirse a un tópico; así es como funciona el sistema por Tópicos de MQTT. Un tópico es una cadena UTF-8 que usa el bróker para filtrar mensajes para cada cliente conectado. Las subscripciones pueden contener caracteres especiales, que permiten subscribirse a múltiples temas a la vez, los cuales se pueden estructurar de manera jerárquica. Cada jerarquía se separa con el signo diagonal (/), por ejemplo:

```
utpl/temperatura
utpl/luminosidad
```

En el ejemplo, se puede ver que los tópicos “temperatura” y “luminosidad” pertenecen al tópico principal *utpl*, pero son temas independientes entre sí. El primero permite subscribir a un tópico en el que se publica la temperatura, y el segundo a un tópico donde se publica luminosidad.

Otro carácter especial o comodín (*wildcard*) es el signo numeral (#) que coincide con cualquier número de niveles dentro de un tema, por ejemplo.

```
utpl/estacionZamora/temperatura
utpl/estacionLoja/temperatura
utpl/#
```

En el ejemplo se observa que el tópico `utpl/#` puede recibir mensajes sobre los temas `utpl/estacionZamora/temperatura` y `utpl/estacionLoja/temperatura.`

Existe otro comodín, el signo más (+) corresponde sólo a un nivel de tema, por ejemplo:

```
utpl/estacionZamora/+
```

En el ejemplo, `utpl/estacionZamora/+` corresponde a `utpl/estacionZamora/temperatura` pero no `utpl/estacionZamora/temperatura/suelo.`

2.3.1.3 Bróker.

Un bróker MQTT es un servidor que implementa el protocolo MQTT, para intervenir en la comunicación entre las aplicaciones de clientes MQTT. Es decir, es la pieza clave de los sistemas basados en la publicación/suscripción de eventos, tales como los que se ejecuta en sensores remotos y otros dispositivos. Asimismo, es importante recalcar que un bróker utiliza un concepto llamado *cola de mensajes*. El flujo de este proceso inicia cuando los emisores producen mensajes. Luego, para que los mensajes lleguen a su destinatario, estos deben ser entregados a un intercambiador que los colocará en la cola del respectivo destinatario. Finalmente, el destinatario puede ir progresivamente desencolando y procesando los mensajes, o dejar que el intercambiador se los haga llegar por medio de diferentes tipos de rutas. En conclusión, el concepto de “cola de mensajes” es un intermediario entre los emisores y los destinatarios, es decir entre cliente-servidor o publicador-consumidor.

2.3.1.3.1 Bróker RabbitMQ.

RabbitMQ es una implementación de bróker de mensajes MQTT. Su núcleo se basa en Spring-erlang, un módulo Java desarrollado por SpringSource. Dicho módulo fue desarrollado utilizando Erlang, un lenguaje específico de altas prestaciones para programación distribuida y tolerancia a fallos; el código fuente está liberado bajo la licencia MPL (Mozilla Public License) (RabbitMQ, 2016). Esta herramienta ofrece al usuario varias ventajas como:

- Soporta múltiples protocolos; trabaja con el protocolo AMQP, sin embargo, mediante *plugins* es posible lograr compatibilidad con otros protocolos como STOMP y MQTT.
- Existe una extensa documentación acerca de sus características y funcionalidades.
- Dispone de librerías en casi todos los lenguajes de programación.
- Ofrece el rastreo de anomalías en las colas de mensajes.
- Ofrece el servicio de *clusterización* de varios servidores de colas para evitar que el sistema de encolamiento falle.
- Brinda múltiples tipos de enrutamiento, e incluso es posible personalizar una propia ruta.
- Soporta *plugins* para extender su comportamiento.
- Posee una interfaz gráfica intuitiva para el usuario. (La figura 5, ilustra la interfaz gráfica RabbitMQ)

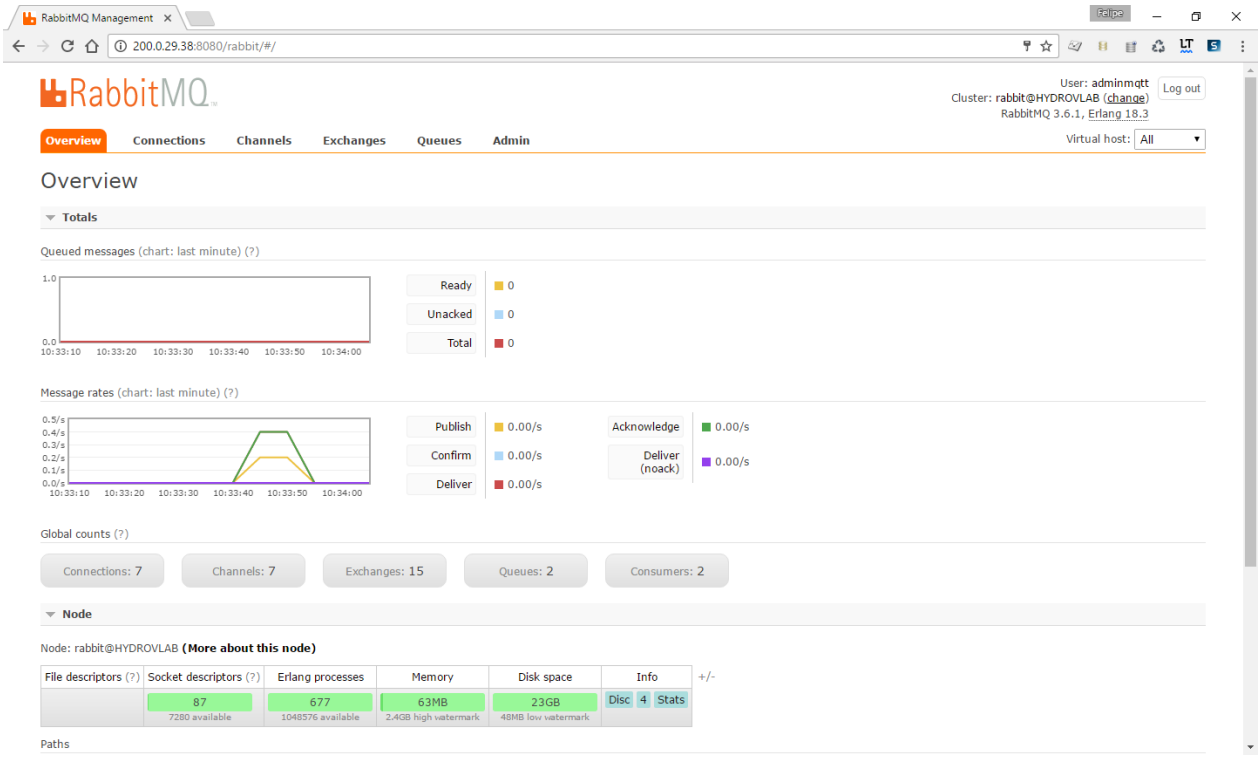


Figura 5: Interfaz de usuario RabbitMQ
Fuente: Tomada de RabbitMQ luego de haber sido implementado en el proyecto.

2.3.2 Ventajas de MQTT.

A continuación, listaremos las ventajas por las cuales preferimos MQTT.

- Gracias a la característica *Last will testament*, MQTT permite al cliente informar al bróker que mensajes debe enviar a determinados tópicos cuando se produce una desconexión anormal.
- Permite múltiples suscripciones.
- Asegura la entrega de mensajes.
- El modelo publicación/suscripción facilita la comunicación de 1 a 1, 1 a muchos.
- Es ideal para redes que tienen poco ancho de banda, alta latencia y conexiones frágiles.
- Es sencillo de implementar y usar.
- Ofrece tres niveles de calidad de servicio.
- Utiliza un único puerto TCP/IP para conectar los clientes a los servidores, lo que permite la sencilla implementación de un firewall y seguridad.
- Funciona en dispositivos con bajos recursos computacionales.

2.3.3 Escenarios donde MQTT ha sido utilizado.

El protocolo MQTT ha sido implementado en algunas áreas; a continuación, algunos ejemplos:

- Facebook utiliza características de MQTT en su Facebook Messenger (Zhang, 2011).
- Amazon Web Services anunció que su plataforma AWS IoT está basada en MQTT.
- Plataformas web como Adafruit IO, IBM Bluemix también utilizan MQTT.
- Isle of Wight Ferries, la compañía de ferries Red Funnel, ha implementado un sistema de notificaciones basado con MQTT. Con este sistema los usuarios pueden observar en tiempo real advertencias acerca de demoras, o estimaciones de tiempo de arribo de ferris.
- El centro médico St. Jude en conjunto con IBM, hicieron un proyecto denominado Sistema Merlin; el cual es un sistema que permite el monitoreo de pacientes desde su hogar. El sistema envía notificaciones al doctor o al hospital a través de MQTT.

A través de los ejemplos, se puede observar que MQTT ha sido utilizado en aplicaciones reales, en diferentes campos en algunos tan importantes, como él envió de la telemetría de un paciente.

2.4 Comparación entre protocolos HTTP y MQTT.

Aunque la comparación se debería hacer entre MQTT y otros protocolos de IoT, la comparación más útil seria con HTTP por las siguientes razones:

- HTTP es el protocolo más utilizado, casi todos los dispositivos informáticos con TCP/IP lo tienen. Además, HTTP y MQTT ambos están basadas en TCP/IP. (Lampkin, y otros, 2012)
- El protocolo HTTP utiliza un modelo de solicitud/respuesta, que es actualmente el protocolo de intercambio de mensajes más común mientras que MQTT utiliza un el modelo publicación/suscripción (Lampkin, y otros, 2012).

En la tabla 2, se realiza una comparación técnica entre los protocolos MQTT y HTTP con respeto a los parámetros de: diseño, modelo de mensajería, complejidad, tamaño de mensaje, calidad de servicio, librerías extra, y comunicación.

Tabla 2: Comparación entre HTTP y MQTT.

	Hypertext Transfer Protocol HTTP	**MQ Telemetry Transport MQTT**
Diseño	HTTP Document Centric, soporta el estándar MIME, que es útil para definir el tipo de contenido. Sin embargo, dispositivos limitados no necesitan esta característica.	MQTT Data Centric permite transmitir los datos como un arreglo de bytes, el cual no se preocupa por el contenido.

Modelo de mensajería	HTTP utiliza el modelo petición/respuesta, un patrón básico y potente de intercambio de mensajes. Pero una limitante es que el cliente necesita saber la dirección de todos los dispositivos a los que se conecta.	MQTT utiliza el modelo publicación/suscripción, un patrón con acoplamiento flexible. Con este protocolo los clientes no necesitan saber que existen otros dispositivos, solo necesitan preocuparse de que el mensaje sea entregado o recibido.
Complejidad	HTTP es un protocolo más complejo, con una especificación que tiene más de 160 páginas. Utiliza varios códigos de retorno y métodos (tales como POST, PUT, GET, DELETE, HEAD, TRACE, CONNECT, etc..). Funciona bien para los sistemas de información hipermedia, pero los dispositivos limitados normalmente no necesitan todas sus características.	La especificación de MQTT es sencilla, tiene pocos tipos de mensajes, solo CONNECT, PUBLISH, SUBSCRIBE, UNSUBSCRIBE, y DISCONNECT son importante para los desarrolladores.
Calidad de Servicio	HTTP no tiene características de calidad de servicio, si los desarrolladores necesitan la entrega de mensajes garantizada, tienen que implementarla ellos mismos un mecanismo de garantía.	Soporta tres niveles de calidad de servicio para la publicación de mensajes. Los desarrolladores no necesitan implementar características adicionales y complejas para garantizar la entrega de mensajes.
Tamaño del mensaje	HTTP utiliza un formato de texto, no un formato binario, que permite extensas cabeceras y mensajes. El formato de texto es legible por los seres humanos, lo que hace que el protocolo HTTP sea fácil de solucionar problemas, contribuyendo a su popularidad. Sin embargo, este formato es más de lo necesario, para dispositivos con recursos computacionales limitados, especialmente en entornos de red de bajo ancho de banda.	MQTT está diseñado específicamente para dispositivos con recursos limitados. Incluye sólo las características que son necesarias para apoyarlos, el encabezado del mensaje en MQTT es corto, y el tamaño del paquete más pequeño para un mensaje es de 2 bytes.
Librerías extra	HTTP no requiere ninguna librería por sí mismo, pero se requieren librerías adicionales de programas que permitan manipular contenido JSON o XML si se utiliza SOAP o en servicios web.	MQTT funciona bien en dispositivos con memoria limitada, en parte debido a sus módicos requerimientos. Por ejemplo, un cliente en C requiere alrededor de 30 KB; mientras que un cliente en Java 100 KB.
Comunicación	La comunicación en HTTP es punto a punto y no tiene ninguna función de distribución incorporada. Los desarrolladores deben crear su propio mecanismo de distribución.	MQTT, incluye un mecanismo que soporta la comunicación de 1 a 0, 1 a 1 y 1 a muchos.

Fuente: Elaborada por los autores basados en OASIS (2013)

La tabla 3 ofrece una comparación de resultados obtenidos entre los protocolos HTTP y MQTT,

tras haber realizado pruebas enviando 1024 mensajes de 1 byte cada uno. Dichas pruebas demostraron que MQTT utiliza menos energía para mantener una conexión abierta, para recibir mensajes y enviarlos, y permite que él envió de mensajes sea más rápida y fiable.

Tabla 3: Comparación de rendimiento y uso de batería entre HTTP y MQTT

Características		**3G**		**WIFI**	
		HTTP	**MQTT**	**HTTP**	**MQTT**
Mensajes Recibidos	Mensajes/Hora	1708	160278	3628	263314
	Porcentaje de la Batería/ Hora	18.43%	16.13%	3.45%	4.23%
	Porcentaje de la Batería/Mensaje	0,01709	0,00010	0,00095	0,00002
	Mensajes recibidos	240/1024	1024/1024	524/1024	1024/1014
Mensajes Enviados	Mensajes/Hora	1926	21685	5229	23184
	Porcentaje de la Batería/ Hora	18,79%	17,80%	5,44%	3,66%
	Porcentaje de la Batería/Mensaje	0,00975	0,00082	0,00104	0,00016

Fuente: Elaborada basada en la información publicada por Nicholas *(2012)*

2.5 Estándar SOS (*Sensor Observation Service*)

Además del desarrollo de una plataforma web con su propio sistema de almacenamiento de datos, otro requerimiento a cubrir es almacenar los datos provenientes de las estaciones meteorológicas bajo el estándar Sensor Observation Service (SOS). SOS es un estándar parte de Sensor Web Enablement (SWE), que al igual que SWE Common (un modelo XML para datos de sensores), SensorML, O&M, y SPS (Sensor Planning Service), ha sido aprobado por OGC (Open Geospatial Consortium) en sus versiones 1.0 y 2.0.

De acuerdo a Bröring, Stasch, & Echterhoff (2012), SOS provee una interfaz estandarizada para administrar y recuperar *metadata* proveniente de sistemas de sensores heterogéneos. Los sistemas de sensores constituyen la mayor parte de la información usada por los sistemas geoespaciales de la actualidad. Los sistemas de sensores incluyen: sensores in-situ (como medidores de nivel de agua de un río), plataformas móviles de sensores (sistemas de satélites o vehículos aéreos no tripulados), y/o redes de sensores estáticos (como sistemas de redes que monitorean movimientos sísmicos).

El estándar SOS define una interfaz de servicio web que permite consultar observaciones, metadatos de sensores, y representaciones de las características observadas. Además, SOS define medios para registrar nuevos sensores y eliminar los existentes; así como operaciones para insertar nuevas observaciones del sensor. Para codificar observaciones, se utiliza el

estándar de Observaciones y Mediciones (O&M); mientras que, para codificar descripciones de sensores se utiliza el SensorML (Sensor Model Language).

2.5.1 Modelo O&M (Observaciones y Mediciones).

Como se mencionó anteriormente, el SOS está diseñado para proporcionar acceso a las observaciones. Para el SOS 2.0, estas observaciones deben estar codificadas de forma predeterminada conforme al estándar de Observaciones y Medidas 2.0 (O&M2.0). La figura 6, muestra los términos centrales que forman parte del modelo O&M, figurando como principales el sitio donde se produce la observación (*feature of interest*) por ejemplo campus universitario UTPL, la fecha en la que se produce el fenómeno (*phenomenon time*), la propiedad que se observa (*observed property*) como ejemplo la velocidad del viento y el valor escalar recolectado en una unidad (*result*) por ejemplo 23 m/s.

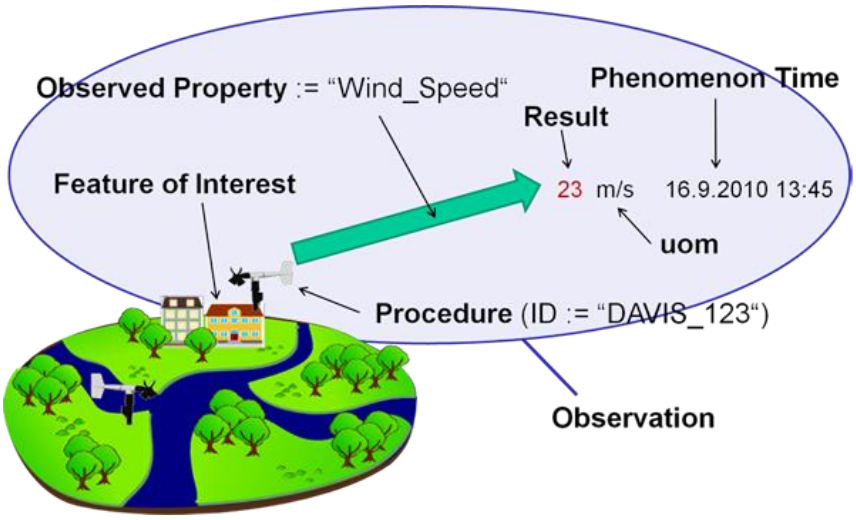


Figura 6: Términos centrales del modelo O&M
Fuente: Tomada del sitio web OGC Network (Open Geospatial Consortium, 2011)

De acuerdo al modelo O&M una observación agrega los siguientes elementos:

- **Lugar de interés *(Feature of interest)*:** es una característica que representa un objeto del mundo real, del cual se observa una o varias propiedades. Por ejemplo: medición del nivel de agua del Río Malacatos.
- **Tiempo observación *(Phenomenon time):*** corresponde al tiempo en que se aplica el resultado de la observación.
- **Tiempo del resultado *(Result time):*** concierne al tiempo en que se ha creado el resultado de la observación.

- **Procedimiento *(Procedure):*** se refiere a la instancia de un proceso que ha realizado la observación. Este es normalmente un sensor físico o sistema de sensores. Pero también puede corresponder a un cálculo, una etapa de procesamiento posterior, o una simulación.
- **Propiedad observada *(Observed property):*** corresponde a una descripción de una propiedad que se observa, por ejemplo: en el agua se podría observar propiedades como temperatura, salinidad, etc.
- **Resultado *(Result):*** este elemento se refiere al valor de resultado de la observación, el mismo que puede ser un valor escalar o una matriz multidimensional compleja. Por ejemplo: 23 m / s.
- **Unidad de medida *(Unit of Measure, uom)***: permite la interpretación del valor del resultado, usualmente se usan códigos UCUM.

2.5.1.1 Procedimiento para modelar datos en O&M.

De acuerdo a la OGC (2011), por lo general el procedimiento para asignar los datos a O&M consiste en los siguientes pasos:

Primero: Identificar los sensores que están ejecutando las observaciones. Estos son los procedimientos del modelo O&M. Si no hay identificadores globales disponibles, es necesario crearlos mediante el formato URL TBD.

Segundo: Identificar las características de interés que son observadas por los sensores, que son usualmente objetos del mundo real como edificios, puntos de interés, objetos rastreados, etc. Para esto se debe tratar de reutilizar las representaciones de funciones ya existentes. En el caso que no lo haya, es necesario definir su propio tipo de entidad y crear identificadores globales utilizando el esquema de URL TBD. Muy a menudo, las observaciones son muestras de fenómenos continuos en el espacio. En ese caso, se puede reutilizar los tipos de características predefinidos para las características de muestreo, tal como se definen en O&M.

Tercero: Identificar las propiedades observadas por los sensores. Para lo cual es necesario tratar de reutilizar las definiciones ya existentes en los diccionarios / vocabularios como se proporciona en la ontología SWEET de la NASA. Si no hay ninguna disponible, entonces se debe proporcionar descripciones y crear identificadores globales usando el esquema de URL.

Cuarto: Identificar los tiempos en los que se ha realizado la observación (phenomenonTime) y cuando se ha creado el resultado de la observación (resultTime).

Quinto: Identificar el tipo de resultado de la observación. Existen varios subtipos de la observación genérica definida en O&M que proporciona tipos de resultados bien definidos. Por ejemplo, si los resultados de la observación son valores numéricos escalares, se debe utilizar la OM_Measurement. Mientras que, si los resultados de la observación son sólo valores booleanos, se puede utilizar OM_TruthObservation.

2.5.2 Operaciones SOS.

SOS utilizado junto con otras especificaciones, ofrece una amplia gama de operaciones interoperables para descubrir, unir e interrogar sensores individuales, plataformas de sensores o constelaciones de sensores en red; los mismos que pueden pertenecer a entornos en tiempo real, archivados o simulados. (Bröring, Stasch, & Echterhoff, 2012).

A continuación, en la tabla 4 se listan operaciones SOS, las misma que de acuerdo a las especificaciones OGC, es obligatorio implementar las tres operaciones correspondientes al perfil Core (GetCapabilities, DescribeSensor, y GetObservation), mientras que las operaciones correspondientes a los perfiles de Transactional y Enhanced son de implementación opcional.

Tabla 4: Operaciones básicas del estándar SOS

Operaciones	Perfil	Implementación	Descripción
GetCapabilities	Core	Mandatoria	Permite describir el servicio que proporciona información sobre el administrador, las capacidades ofrecidas, las propiedades observadas, las características, etc.
DescribeSensor	Core	Mandatoria	Proporciona una descripción potencialmente detallada de un componente, sistema o proceso registrado en formato SensorML.
GetObservation	Core	Mandatoria	Proporciona observaciones basadas en el establecimiento de filtros que incluyen tiempo, procesos, fenómenos, características de interés y otros parámetros en el modelo O&M.
RegisterSensor	Transactional	Opcional	Provee la capacidad de registrar automáticamente un nuevo sensor al servicio existente.
InsertObservation	Transactional	Opcional	Proporciona la capacidad de insertar dinámicamente nuevas observaciones relacionadas con un sensor registrado.

GetFeatureOfInterest	Enhanced	Opcional	Proporciona la característica solicitada de interés en formato GML.
GetResult	Enhanced	Opcional	Proporciona una manera ligera de solicitar observación sin proporcionar una solicitud completa cada vez.
GetObservationByID	Enhanced	Opcional	Presta un acceso rápido a la observación por número de identificación.
GetFeatureOfInterestTime	Enhanced	Opcional	Proporciona el intervalo de tiempo cuando se ha observado una característica de interés.
DescribeFeatureType	Enhanced	Opcional	Facilita el esquema utilizado para representar las características de interés.
DescribeObservationType	Enhanced	Opcional	Proporciona el esquema utilizado para representar las Observaciones.
DescribeResultModel	Enhanced	Opcional	Proporciona el esquema utilizado para representar el objeto de resultado dentro de la observación SML.

Fuente: Elaborada basada en la información publicada en el sitio web “Introduction to the SOS standard” (Milan Antonovic , 2015).

2.5.3 Elementos SOS

De acuerdo a Milan Antonovic (2015), el estándar SOS está basado en cinco elementos, como se ilustra en la figura 7.

Figura 7: Elementos SOS
Fuente: Tomado del sitio web "Introduction to the SOS standard" (Milan Antonovic , 2015)

- **Observaciones *(Observations)*:** son el centro de la norma y representan los valores medidos en instantes de tiempo determinados representados de acuerdo con el modelo de datos estándar de O&M. Por ejemplo: valor: 0,2, tiempo: 08-11-2012 12:12.
- **Procedimiento *(Procedure)*:** indica quién proporciona las observaciones, éste es generalmente el sensor, pero también puede ser un proceso genérico que conduce a algunas observaciones (por ejemplo: procedimiento: TREVANO) y se representa como modelo de datos estándar de SensorML.
- **Propiedades observadas *(Observed properties)*:** representan los fenómenos que se observan (por ejemplo: temperatura del aire) y se identifican con un URI (Identificador de recurso uniforme) compuesto por un texto separado por dos puntos según la característica *om:observedProperty* de la norma O&M.
- **Lugar de interés *(Feature of interest)*:** es el factor que se relaciona con las observaciones, por lo que para un instrumento in-situ es la ubicación del sensor, mientras que para el dispositivo remoto es la ubicación destino; representado de acuerdo con el elemento *om: featureOfInterest* de la norma O&M.
- **Muestrario *(Offering)*:** es una colección de sensores usados para agruparlos convenientemente (por ejemplo: sensor de tiempo SUPSI) y se representa con el elemento *sos:ObservationOffering* del estándar SOS.

2.5.4 52º North SOS, una implementación del estándar SOS.

52º North SOS (Sensor Observation Service) es una plataforma de código abierto que permite

publicar los datos de un sistema en el estándar SOS. Esta plataforma soporta la provisión interoperable de datos correspondiente a la observación de sensores, en entornos en tiempo real, archivados o simulados. Un sensor podría ser un medidor de nivel de agua en una corriente, una estación meteorológica, o una estación de monitoreo de la calidad del aire.

La implementación 52º North SOS ofrece las siguientes características:

- **Cliente web *(Browser client)*:** proporciona medios para la administración y configuración de la instancia de servicio, así como el envío de solicitudes de prueba para operaciones SOS. Además, se incluyen ejemplos de peticiones, que facilitan la comprensión del usuario. En la figura 8, se presenta una captura de pantalla de la interfaz web que ofrece 52° North, en la que se encuentran disponibles ejemplos de peticiones SOS, las mismas que pueden ser tomadas como referencia para la elaboración de una trama personalizada para su posterior ejecución.
- **Cliente Javascript *(JavaScript client)***: El SOS contiene una aplicación JavaScript para mostrar datos de series temporales. Con su ayuda, las series de tiempo se visualizarán sin ningún software adicional. Dentro de la figura 8 se encuentra una serie temporal de datos recolectados para tres variables medioambientales que reflejan el uso de cliente *JavaScript*.
- **Especificaciones *(Specifications)***: El 52° North SOS implementa las operaciones de la versión SOS 2.0., así como de la versión anterior 1.0.
- ***Inspire download service extension***: A partir de la versión 4.2.0, está disponible una extensión que permite utilizar SOS como servicio de descarga INSPIRE. La información de fondo se proporciona en un blog y en wiki.
- ***Air quality data e-reporting extension:*** A partir de la versión 4.3.0 hay una extensión disponible que permite proporcionar los datos compatibles E1a, E1b y E2a de los flujos de datos de calidad del aire.
- ***Restful api for timeseries:*** comenzando con la Versión 4.1.1, es una versión de paquete disponible, que trae directamente la API RESTful. Por lo tanto, timeseries se pueden solicitar de una petición RESTful en las solicitudes codificadas JSON.

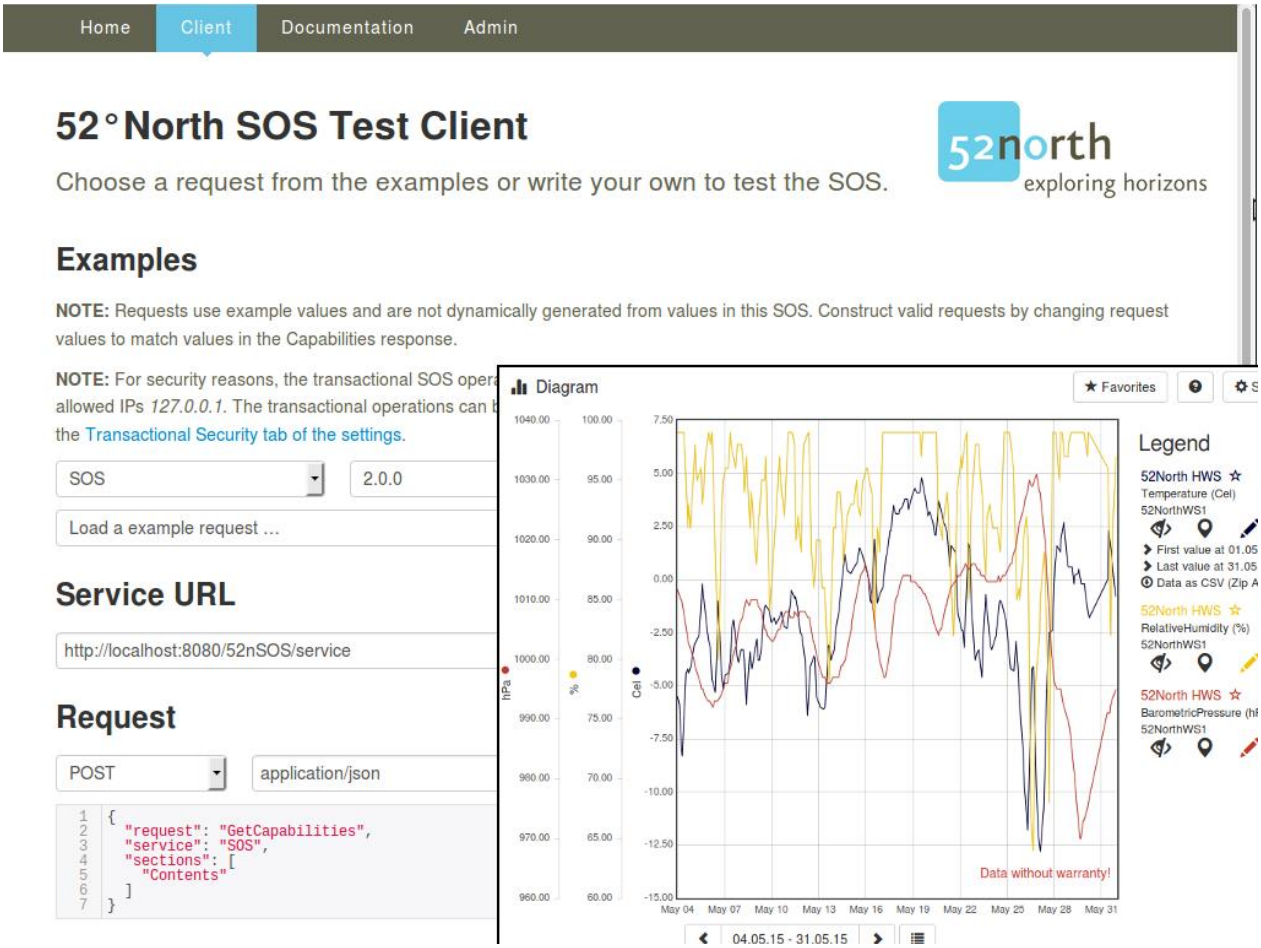
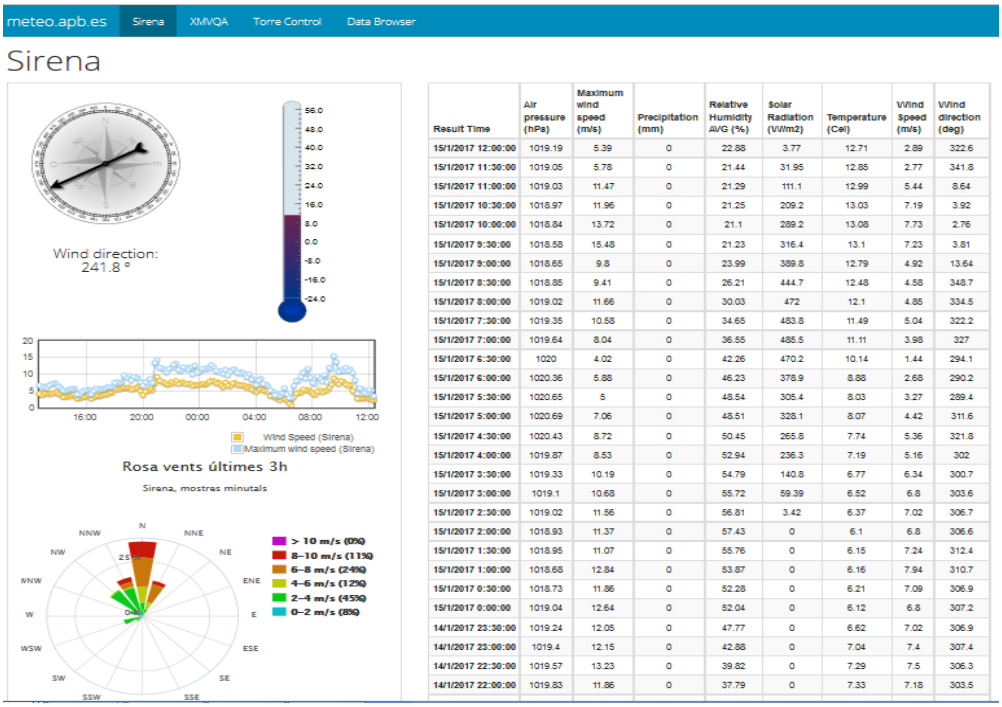


Figura 8: Interfaz 52°North SOS
Fuente: Tomado del sitio web “52°North SOS” (OSGeo-Live, 2016)

2.5.5 Sensor Widgets, un componente gráfico para presentar datos en formato SOS.

Es una librería de código abierto basado en Javascript, que ofrece una gama de componentes gráficos configurables para la presentación de datos-sensor modelados en estándar SOS. Cada componente gráfico se lo denomina “widget”, el cual contiene una colección de parámetros obligatorios y otros opcionales. La configuración de un *widget* es básicamente definir los valores adecuados a un conjunto de parámetros para obtener el resultado deseado. La manera más fácil de configurar un *widget* es usando el Wizard, que provee asistencia en la selección de los parámetros basándose en una lista de posibles valores. La figura 9 muestra los componentes gráficos disponibles en *Sensor Widgets*; mientras que la figura 10 corresponde a una captura de pantalla del sitio web “meteo.apb.es”, el cual implementa la librería *Sensor Widgets*.

Figura 9: Componentes gráficos Sensor Widgets
Fuente: Tomado del sitio web “Sensor Widgets” (GidHub, 2016)



Result Time	Air pressure (hPa)	Maximum wind speed (m/s)	Precipitation (mm)	Relative Humidity AVG (%)	Solar Radiation (W/m2)	Temperature (Cel)	Wind speed (m/s)	Wind direction (deg)
15/1/2017 12:00:00	1019.19	5.39	0	22.88	3.77	12.71	2.89	322.6
15/1/2017 11:30:00	1019.05	5.78	0	21.44	31.95	12.85	2.77	341.8
15/1/2017 11:00:00	1019.03	11.47	0	21.29	111.1	12.99	5.44	8.64
15/1/2017 10:30:00	1018.97	11.96	0	21.25	209.2	13.03	7.19	3.92
15/1/2017 10:00:00	1018.84	13.72	0	21.1	289.2	13.08	7.73	2.76
15/1/2017 9:30:00	1018.58	15.48	0	21.23	316.4	13.1	7.23	3.81
15/1/2017 9:00:00	1018.65	9.8	0	23.99	389.8	12.79	4.92	13.64
15/1/2017 8:30:00	1018.85	9.41	0	26.21	444.7	12.48	4.58	348.7
15/1/2017 8:00:00	1019.02	11.66	0	30.03	472	12.1	4.85	334.5
15/1/2017 7:30:00	1019.35	10.58	0	34.65	483.8	11.49	5.04	322.2
15/1/2017 7:00:00	1019.64	8.04	0	36.55	485.5	11.11	3.98	327
15/1/2017 6:30:00	1020	4.02	0	42.26	470.2	10.14	1.44	294.1
15/1/2017 6:00:00	1020.36	5.88	0	46.23	378.9	8.88	2.68	290.2
15/1/2017 5:30:00	1020.65	5	0	48.54	305.4	8.03	3.27	289.4
15/1/2017 5:00:00	1020.69	7.06	0	48.51	328.1	8.07	4.42	311.6
15/1/2017 4:30:00	1020.43	8.72	0	50.45	265.8	7.74	5.36	321.8
15/1/2017 4:00:00	1019.87	8.53	0	52.94	236.3	7.19	5.16	302
15/1/2017 3:30:00	1019.33	10.19	0	54.79	140.8	6.77	6.34	300.7
15/1/2017 3:00:00	1019.1	10.68	0	55.72	59.39	6.52	6.8	303.6
15/1/2017 2:30:00	1019.02	11.56	0	56.81	3.42	6.37	7.02	306.7
15/1/2017 2:00:00	1018.93	11.37	0	57.43	0	6.1	6.8	306.6
15/1/2017 1:30:00	1018.95	11.07	0	55.76	0	6.15	7.24	312.4
15/1/2017 1:00:00	1018.68	12.84	0	53.87	0	6.16	7.94	310.7
15/1/2017 0:30:00	1018.73	11.86	0	52.28	0	6.21	7.09	306.9
15/1/2017 0:00:00	1019.04	12.64	0	52.04	0	6.12	6.8	307.2
14/1/2017 23:30:00	1019.24	12.05	0	47.77	0	6.62	7.02	306.9
14/1/2017 23:00:00	1019.4	12.15	0	42.88	0	7.04	7.4	307.4
14/1/2017 22:30:00	1019.57	13.23	0	39.82	0	7.29	7.5	306.3
14/1/2017 22:00:00	1019.83	11.86	0	37.79	0	7.33	7.18	303.5

Figura 10: Sitio web que implementa la librería Sensor Widgets
Fuente: Tomado del sitio web “*meteo.apb.es*” (2016)

2.6 Análisis de herramientas de software disponibles

A continuación, se realiza una descripción de las herramientas de software disponibles en el medio correspondientes a lenguajes de programación y gestor de datos.

2.6.1 Lenguajes de programación.

Los lenguajes de programación permiten desarrollar aplicaciones de cualquier tipo, mediante las cuales se puede ejecutar un determinado proceso o tarea. Es por ello que es necesario tener en cuenta que, el desarrollo web ha evolucionado en las últimas décadas de una manera impresionante. Básicamente, un sitio web se compone de dos partes importantes *back-end* y *front-end*, las cuales interactúan activamente para brindar un correcto funcionamiento al usuario. El *back-end* es la parte que se encuentra de lado del servidor; esta parte es la que se encarga de

interactuar con la base de datos, permite verificar el manejo de sesiones de usuarios, es la encargada de la manipulación de los datos, y constituye la parte no visible para el usuario ya que no se trata de diseño, o elementos gráficos. Mientras que el *front-end*, es la parte que se encuentra en el lado del cliente; esta parte es la que interactúa con el usuario. Luego de una exhaustiva investigación bibliográficas, se ha seleccionado a Java y AngularJs como lenguajes para el *back-end* y *front-end*; en las secciones siguientes, se explicará los criterios para su selección.

2.6.1.1 Java (back-end).

Java es un lenguaje de programación que permite desarrollar programas que se ejecuten en una gran variedad de sistemas computacionales y dispositivos controlados por computadora. A esto se le conoce algunas veces como “escribir una vez, ejecutar en cualquier parte” (Deitel & Deitel, 2012)

Feb 2016	Feb 2015	Change	Programming Language	Ratings	Change
1	2	^	Java	21.145%	+5.80%
2	1	v	C	15.594%	-0.89%
3	3		C++	6.907%	+0.29%
4	5	^	C#	4.400%	-1.34%
5	8	^	Python	4.180%	+1.30%
6	7	^	PHP	2.770%	-0.40%
7	9	^	Visual Basic .NET	2.454%	+0.43%
8	12	^^	Perl	2.251%	+0.86%

Figura 11: Popularidad de lenguajes de programación
Fuente: Tomado del sitio web “TIOBE, the software quality company ” (TIOBE software BV, 2016)

Tal como se puede apreciar en la figura 11, el lenguaje de programación que está en primer lugar en base a su popularidad es Java. Hoy por hoy, Java es uno de los líderes en el entorno empresarial, principalmente por las ventajas que presta al desarrollador. A través de este lenguaje se puede crear aplicaciones que permite la interacción con el usuario.

Entre las principales ventajas que Java presenta a los desarrolladores que hacen uso de este lenguaje, se tiene:

- **Confiabilidad:** es un lenguaje que trata de minimizar los errores, al máximo.
- **Robustez:** presenta los errores simultáneamente, lo que permite una depuración fácil.

- **Reutilización de código**: permite la reutilización de código según sea necesario, ya que está compuesto de piezas llamadas clases.
- **Portabilidad:** su compilador genera aplicaciones en *byte-code*, que es un formato que permite la transportación de código a múltiples plataformas, permitiendo así la ejecución de los binarios en varios sistemas Operativos.
- **Continuo desarrollo:** a través del JDK es una herramienta libre de licencias (sin costo) creada por Sun que permite tener actualización permanente.
- **Flexibilidad:** permite a los desarrolladores aprovechar la flexibilidad de la Programación Orientada a Objetos en el diseño de sus aplicaciones.

Java es un lenguaje que es bastante expansible, muy productivo y simple de manejar. Con el pasar de los años este lenguaje de programación ha madurado, debido a que se ha implementado un número importante de librerías y componentes con la finalidad de mejorar sus funcionalidades. Consecuentemente, Java permite la implementación de aplicaciones web con mucha más facilidad y en tiempos considerables que otros lenguajes (Deitel & Deitel, 2012).

En base a las características descritas se ha determinado usar Java como lenguaje de programación para el desarrollo del sistema y en base a los siguientes argumentos:

- *Opera de forma independiente de la plataforma.* Cuando se programa cualquier tipo de aplicación, no se necesita saber a priori el tipo de ordenador o el sistema operativo en donde se ejecutará la aplicación.
- *Maneja técnicas de programación orientada a objetos.* Java maneja excelentemente el paradigma de la programación orientada a objeto, ya que es capaz de semejar la forma de programar a la forma de pensar del ser humano.
- *Facilidad de Aprendizaje.* El Lenguaje Java es relativamente fácil de aprender, comparado con otros lenguajes, ya que la sintaxis es muy intuitiva; además de existir abundante información en la web.

2.6.1.2 JavaScript – AngularJs (front-end).

AngularJs es un *framework* de JavaScript y se utiliza para el desarrollo del *front-end* del sistema, debido a que está basado en los Modelos Vista Controlador (MVC) y el Modelo Vista-Vista modelo (MVVM). Por otro lado, al ser AngularJS un *framework* desarrollado en JavaScript, todas sus acciones se ejecutan en el lado del cliente; lo que permite que las aplicaciones interactúen de manera inmediata con el usuario (Google, 2016).

Entre las razones que justifican la selección del *framework* Angular Js, se tiene las siguientes:

- Es el *framework* con mayor crecimiento en cuanto a uso y popularidad, esto ayuda mucho en el aprendizaje del *framework*; pues es muy fácil encontrar tutoriales para todo tipo de tareas y resolución de dudas.
- Tiene el soporte de una empresa plenamente consolidada, como lo es Google; por lo que el *framework* siempre se encuentra en constante crecimiento y mejora.
- Proporciona controladores, servicios y directivas para organizar el proyecto.
- Permite crear componentes fácilmente reutilizables, al tener componentes aislados, se puede testear su comportamiento de manera independiente.
- Permite separar la lógica de negocio de la presentación.
- Finalmente, AngularJS puede implementarse en conjunto con otros marcos de desarrollo.

2.6.2 Gestor de base de datos.

En la actualidad las bases de datos cumplen un rol fundamental en nuestras vidas cotidianas, hasta llegar al punto en que muchas de las veces no estamos conscientes en que en determinado proceso se está utilizando una base de datos. Según Saravia (2014), una base de datos es una colección de datos organizados y relacionados entre sí, los cuales son recolectados y explotados por los sistemas de una empresa o negocio en particular. Conforme los sistemas computacionales fueron evolucionando, los usuarios comenzaron a denotar que la información era un recurso corporativo de vital importancia para las empresas. Lo que dio inicio a los sistemas de base de datos, cuyo objetivo principal era garantizar el acceso a los datos y su manipulación desde cualquier plataforma. Actualmente, un Sistema de Gestión de Base de Datos (SGBD) cumple un rol fundamental en el funcionamiento de cualquier aplicación, permitiendo recuperar y almacenar la información generada en determinada entidad. El modelo de SGBD actualmente más utilizado es el relacional, pero en los últimos años una visión alternativa a dicho modelo que está en auge es el no relacional, también conocido como NoSQL.

En la siguiente sección se presentará una implementación del modelo no relacional y las ventajas que presenta para su uso.

2.6.2.1 MongoDB.

MongoDB es un sistema de base de datos multiplataforma orientado a documentos de esquema libre. Esto significa que cada entrada o registro puede tener un esquema de datos diferentes, con

atributos o “columnas” que no tienen por qué repetirse de un registro a otro. MongoDB ha sido desarrollado en C++, lo que proporciona un acceso directo a los recursos de hardware de la máquina, brindando rapidez a la hora de ejecutar tareas. Por otro lado, MongoDB proporciona una plataforma para el desarrollo de aplicaciones altamente escalables orientadas al paradigma IoT. MongoBD funciona en sistemas operativos Windows, Linux, OS X o Solaris; además está licenciado bajo la licencia GNU AGPL 3.0, de modo que se trata de un software libre (Bosch Software Innovations and Mongo DB, 2015).

El motor de base de datos MongoDB ofrece las siguientes características:

- Permite manejar un gran volumen de datos, los cuales en el desarrollo de la presente solución corresponderían a los datos provenientes de las estaciones meteorológicas. Las cuales se estima que tendrá un crecimiento exponencial de los datos.
- MongoDB cuenta con la ventaja de estar pensado para escalar horizontalmente, a través de la combinación de los métodos de *replication* y *sharding* que soporta. Lo que permite el manejo rápido de un gran volumen de lecturas y escrituras de datos.
- Tiene abundante documentación para su implementación, mantenimiento y escalabilidad.
- MongoDB tiene la capacidad de realizar consultas utilizando Javascript. haciendo que estas sean enviadas directamente a la base de datos para ser ejecutada.
- La configuración automática de nuevas máquinas a MongoDB en tiempo real, hace posible que el sistema se expanda sin necesidad que el sistema base sea reseteado para continuar con su ejecución.
- Se han garantizado casos de éxito para distintos tipos de aplicaciones. Por ejemplo, MongoDB es utilizado en aplicaciones de empresas tan conocidas como Foursquare, Sourceforge, Google, Facebook o New York Times, e incluso el CERN (Organización Europea para la Investigación Nuclear) utiliza MongoDB para los grandes volúmenes de datos que genera el acelerador de partículas.

Otro aspecto que destaca a MongoDB es su característica de indexación. En las bases de datos relacionales, el uso de índices es algo indispensable; sería una tarea abismal realizar una consulta en una tabla con millones de registros si no se ha configurado al menos un índice. Con MongoDB sucede lo mismo, sería difícil de repasar una colección con millones de documentos, sin tener índices sobre uno o varios campos. Los índices en MongoDB se generan en forma de *Árbol-B* o *B-Tree*. Es decir, que los datos se guardan en forma de árbol, pero manteniendo los nodos balanceados. Esto permite un incremento en la velocidad a la hora de buscar y también a

la hora de devolver resultados ya ordenados. MongoDB es capaz de recorrer los índices en ambos sentidos, por lo que, con un solo índice, se es posible conseguir ordenación tanto ascendente como descendente. Por lo tanto, el uso de índices ayuda a que las búsquedas se hagan de una manera más rápida; siendo esto un aspecto crucial para el desarrollo del presente proyecto, ya que se requiere que los tiempos de respuesta sean los más óptimos (Bosch Software Innovations and Mongo DB, 2015).

Por todas las ventajas mencionadas, se ha seleccionado como gestor de base de datos a MongoDB para el desarrollo del presente proyecto de investigación. Particularmente, las características MongoDB más relevantes, que cubre los requerimientos de la arquitectura a desarrollar, son la velocidad y su sistema de consulta sencillos. Lo que permitirá alcanzar un balance entre rendimiento y funcionalidad.

2.7 Metodología de desarrollo

Las metodologías para el desarrollo de software constituyen modos metódicos de realizar, gestionar y administrar proyectos para su implementación, con altas posibilidades de éxito. Cada metodología comprende procesos a seguir sistemáticamente para idear, desarrollar y mantener un producto de software desde que surge la necesidad del producto hasta que se cumplan el objetivo para el cual fue creado. Por tal razón el uso de las metodologías de desarrollo de aplicaciones es un aspecto casi imposible ignorar, ya que con ello se lleva un mejor control y logrando un gran nivel de competitividad en todo momento. Generalmente las metodologías llevan a cabo una serie de tareas comunes que son buenas prácticas, para lograr los objetivos planteados, independientemente de cómo se hayan diseñadas. De acuerdo a Sampalo et al. (2003, pág. 77), una metodología de desarrollo es una recopilación de técnicas y procedimientos estructurados en fases para la producción de productos software de manera eficaz, y englobando todo el ciclo de vida del mismo. En otras palabras, la metodología de desarrollo indica qué hacer, cuándo y quién debe hacerlo; así como determina las etapas y controles a aplicar.

2.7.1 SCRUM.

Con la finalidad de garantizar la calidad del proyecto y la satisfacción de las necesidades de los usuarios finales, el presente proyecto se desarrollará en base a las fases de la metodología SCRUM. La misma que es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos. En SCRUM se realizan entregas

parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, SCRUM está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, los requisitos son cambiantes o poco definidos, y la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

De acuerdo a Layton (2015), entre los beneficios que se obtiene al aplicar SCRUM como metodología de desarrollo están los siguientes:

- *Cumplimento de expectativas:* El cliente establece sus expectativas indicando el valor que le aporta cada requisito / historia del proyecto, el equipo los estima y con esta información el Product Owner establece su prioridad. De manera regular, en las demos de Sprint el Product Owner comprueba que efectivamente los requisitos se han cumplido y transmite el feedback al equipo.
- *Flexibilidad a cambios:* Alta capacidad de reacción ante los cambios de requerimientos generados por necesidades del cliente o evoluciones del mercado. La metodología está diseñada para adaptarse a los cambios de requerimientos que conllevan los proyectos complejos.
- *Reducción del time to market*: El cliente puede empezar a utilizar las funcionalidades más importantes del proyecto antes de que esté finalizado por completo.
- *Mayor calidad del software:* La metódica de trabajo y la necesidad de obtener una versión funcional después de cada iteración, ayuda a la obtención de un software de calidad superior.
- *Mayor productividad:* Se consigue entre otras razones, gracias a la eliminación de la burocracia y a la motivación del equipo que proporciona el hecho de que sean autónomos para organizarse.
- *Reducción de riesgos*: El hecho de llevar a cabo las funcionalidades de más valor en primer lugar y de conocer la velocidad con que el equipo avanza en el proyecto, permite despejar riesgos eficazmente de manera anticipada.

CAPÍTULO 3: DESARROLLO DE LA SOLUCIÓN

3.1 Introducción

En este capítulo se describe el diseño de la solución construida para el presente trabajo de titulación, específicamente enfocado en tener un sistema que permita recolectar la información en tiempo real desde una red de estaciones meteorológicas distribuida en algunos puntos geográficos de las provincias de Loja y El Oro. Se ha diseñado y estructurado la solución de manera que sea fácil e intuitiva para el usuario final, así como se ha mantenido un enfoque de escalabilidad. Concretamente, el contenido de este capítulo presenta la especificación de requerimientos de la solución a desarrollar, la arquitectura del sistema, y la adecuación del ambiente de trabajo para el desarrollo de software.

3.2 Problema

Se requiere diseñar e implementar una arquitectura para el almacenamiento y análisis de datos obtenidos en la red de sensores de la UTPL, la cual es utilizada para la recolección variables medioambientales en varios puntos geográficos de la provincia de Loja y El Oro.

3.3 Metodología de desarrollo

La metodología que se aplicó para la planificación y desarrollo del sistema es *scrum*, la cual se abordó en el capítulo 2.

Para más detalle de la planificación revisar el anexo #2 Planificación de la solución.

3.3.1 Roles del sistema

El equipo de trabajo consolidado acorde a los roles que ofrece *scrum* se encuentra detallado en la tabla 5.

Tabla 5: Roles y responsabilidades dentro del trabajo de titulación

Rol	Responsabilidades	Responsable
Interesados	Brindar los requisitos del sistema.	Manuel Quiñones Víctor González Luis Santiago Quiñones
Dueño de producto (*product owner*)	Definir los objetivos del producto o proyecto.	Manuel Quiñones
Jefe de proyecto(*Scrum master*)	Realizar el seguimiento de los procesos	Víctor González
Equipo de desarrollo (*Development*)	Realizar el seguimiento de los procesos. Grupo de profesionales con los conocimientos técnicos necesarios y que desarrollan el proyecto de manera conjunta. Ejecutar buenas prácticas.	Felipe Quiñonez Byron Figueroa

Fuente: Elaborado por los autores.

3.3.2 Listado de objetivos del sistema (*product backlog*)

La lista de objetivos priorizada constituye la visión y expectativa del cliente respecto a los entregables del producto, este detalle se encuentra reflejado en la tabla 6.

Tabla 6: Listado de objetivos del sistema (*product backlog*)

Identificador de Objetivo	Descripción	Prioridad
O1	Conexión con bróker	Media
O2	Evaluación de tramas y deserialización	Media
O3	Almacenamiento de tramas en base de datos	Alta
O4	Definir y crear una interfaz grafica	Alta
O5	Autentificación de usuarios	Alta
O6	Crear usuarios	Alta
O7	Dar de baja usuarios	Media
O8	Asignar roles	Media
O9	Crear estaciones meteorológicas	Alta
O10	Crear variables medioambientales	Alta
O11	Descarga de historial de variable medioambiental	Media
O12	Visualizador de datos históricos	Media
O13	Crear alertas	Media
O14	Visualizador de alertas	Media
O15	Notificador de alertas	Alta
O16	Definir y crear una interfaz grafica	Media
O17	Clusterización de estaciones	Media
O18	Consumo de servicio 52North	Media

Fuente: Elaborado por los autores.

3.3.3 Listado de tareas (*spring backlog*)

En la tabla 7, se presenta un listado clasificado de las diferentes iteraciones (*sprints*) ejecutadas para el desarrollo de la solución.

Tabla 7: Clasificación de iteraciones (*product backlog*)

Número de iteración	Tema
Sprint 0	Definición de los requisitos del producto.
Sprint 1	Desarrollo del subsistema de almacenamiento.
Sprint 2	Desarrollo del módulo de administración.
Sprint 3	Desarrollo del módulo de gestión de estaciones y variables medioambientales.
Sprint 4	Desarrollo del módulo de notificación.
Sprint 5	Desarrollo del subsistema de visualización de datos.

Fuente: Elaborado por los autores.

3.4 Especificación de requerimientos de la solución

Para dar solución al problema, se ha procedido a aplicar varias técnicas y estrategias para la captura y recolección de información como: entrevistas al personal involucrado y observación directa al proceso, para lograr su efectiva representación a través de diagramas de procesos. Posteriormente, se realizó un análisis de la información recolectada con el fin de definir las necesidades y características esenciales del sistema. En la tabla 8 se detalla los requerimientos funcionales y no funcionales (ver tabla 9) identificados luego de la ejecución del *sprint 0*.

Tabla 8: Requerimientos funcionales del sistema

Identificador	Nombre	Descripción
RF01	Autenticación de los usuarios.	El acceso al sistema deberá ser restringido a usuarios autorizados. Para ello, el sistema deberá solicitar el usuario y contraseña del usuario para validar su ingreso.
RF02	Gestión de estaciones meteorológicas.	Las estaciones meteorológicas deben ser registradas en el sistema, además se deberá permitir la asignación de las variables medioambientales a una estación. Además, se deberá permitir la actualización de los campos ingresados.
RF03	Gestión de variables medioambientales	Las variables medioambientales deben ser registradas en el sistema, así como pueden ser actualizados sus campos. Además, las variables medioambientales podrán ser asociadas a determinada estación meteorológica.
RF04	Visualizar información meteorológica.	El sistema ofrecerá al público en general la visualización de la información referente a las variables medioambientales almacenadas en las diferentes estaciones meteorológicas.
RF05	Gestión de descargas	El sistema debe permitir realizar la descarga del histórico almacenado de una variable medioambiental asociada determinada estación meteorológica. Para lo cual es necesario especificar el rango de las fechas de las cuales se quiere obtener los datos.
RF06	Gestión de alertas.	Las alertas deben ser registradas en el sistema, así como se puede actualizar la condición establecida.

RF07	Gestión de Notificaciones de Alertas.	El sistema deberá ser capaz de enviar notificaciones vía mail o SMS cuando una alerta ha cumplido las condiciones establecidas. Siendo las notificaciones vía mail activadas para todos los usuarios; mientras que las notificaciones por SMS serán activadas únicamente por el usuario administrador de la plataforma para un usuario especifico.
RF08	Almacenar tramas de estaciones meteorológicas.	Es necesario que las tramas que se reciben de las estaciones meteorológicas puedan almacenarse en una base de datos MongoDB.
RF09	Envió de observaciones al sistema SOS.	Es necesario que las tramas provenientes de una estación meteorológica se conviertan al lenguaje sensorML y se envíen mediante una petición POST al sistema 52North.

* Para más detalle de los requerimientos funcionales revisar Anexo #3 Especificación de Requerimientos Software
Fuente: Elaborada por los autores.

Tabla 9: Requerimientos no funcionales del sistema

Identificador	Nombre	Descripción
RNF01	Rendimiento	Se debe garantizar que las consultas u otros procesos no afecte el desempeño de la base de datos.
RNF02	Seguridad	El sistema debe proveer mecanismos de seguridad que impidan la vulnerabilidad de los datos y garantice la integridad de la información.
RNF03	Disponibilidad	El sistema debe ser desarrollado tomando en cuenta las necesidades, requerimientos y objetivos de la UTPL, por lo que deberá estar disponible al menos el 90% de 24/7.
RNF04	Portabilidad	Al utilizar herramientas de software libre se está garantizando la portabilidad, además se utilizará el lenguaje Java y base de datos MongoDB.
RNF05	Fiabilidad	El sistema debe tener una interfaz de uso intuitiva y sencilla, así como debe capaz de responder ante todo tipo de incidente.
RNF06	Mantenibilidad	El sistema debe disponer de una documentación que permita realizar operaciones de mantenimiento con el

menor esfuerzo posible.

* Para más detalle de los requerimientos no funcionales revisar Anexo #3 Especificación de Requerimientos Software
Fuente: Elaborada por los autores.

3.5 Arquitectura del sistema

Luego de especificar los requerimientos tanto funcionales como no funcionales, se procede a la implementación de una arquitectura de software capaz de abarcar los mismos. Para lo cual se subdividió el sistema en tres grandes subsistemas: subsistema de almacenamiento, subsistema de gestión de estaciones meteorológicas y variables medioambientales, y el subsistema de visualización de datos para el público en general. Consecuentemente se definió aplicar la arquitectura basada en eventos para el primer subsistema; mientras que la arquitectura en tres capas para el segundo subsistema.

Para más detalle de la arquitectura de software revisar el anexo #4 Documento de arquitectura de software. En la figura 12, se presenta la arquitectura global de estos tres subsistemas

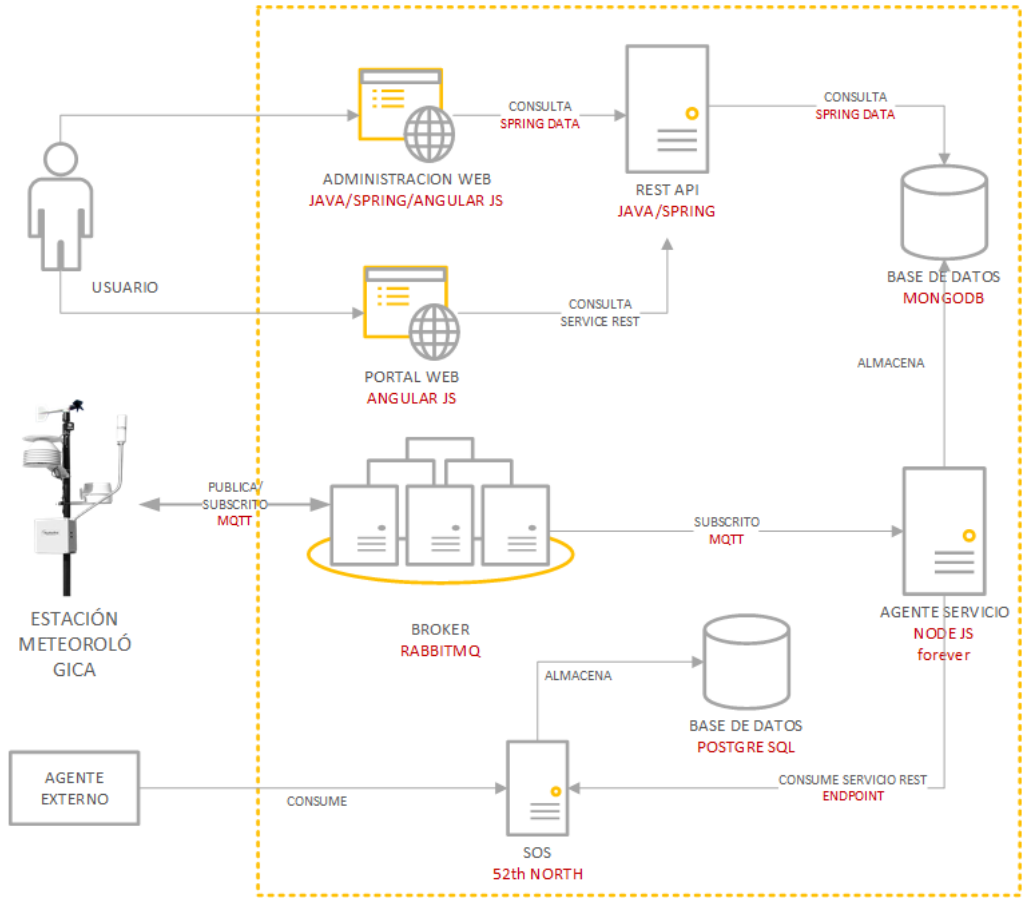


Figura 12: Arquitectura global de los subsistemas diseñados
Fuente: Elaborada por los autores.

3.5.1 Subsistema de almacenamiento.

Para el subsistema de almacenamiento (SAVA) se ha definido una arquitectura basada en eventos. En este tipo de modelo se promueven la producción, detección, y reacción a eventos. Un evento es el cambio de un estado, el consumidor tiene la responsabilidad de llevar a cabo una reacción tan pronto como el evento esté presente. El subsistema SAVA activa condiciones ante los eventos entrantes como mensajes provenientes de las estaciones. El subsistema de almacenamiento está constituido por dos componentes: componente de almacenamiento en Mongo BD, y componente de almacenamiento en SOS. Asimismo, es importante mencionar que para el desarrollado de este subsistema se ha utilizado el lenguaje de programación JavaScript y el entorno de ejecución NodeJs. La figura 13 presenta la estructura de archivos que conforman este subsistema.

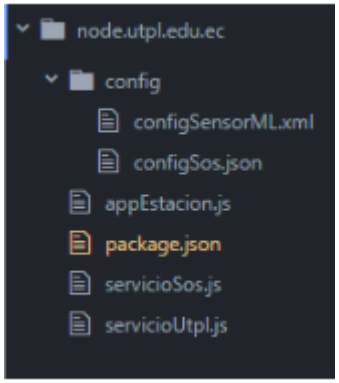


Figura 13: Estructura de archivos del subsistema SAVA
Fuente: Elaborada por los autores.

3.5.1.1 Componente de almacenamiento en Mongo BD.

Este componente es el encargado de almacenar todos los mensajes que envían las estaciones meteorológicas o cualquier otro dispositivo al bróker MQTT; su función es deserializar todos los mensajes que recibe de acuerdo a un patrón establecido y los almacena en las diferentes colecciones de MongoDB que se ha establecido. En la figura 14, se puede ver un diagrama de secuencia de este proceso.

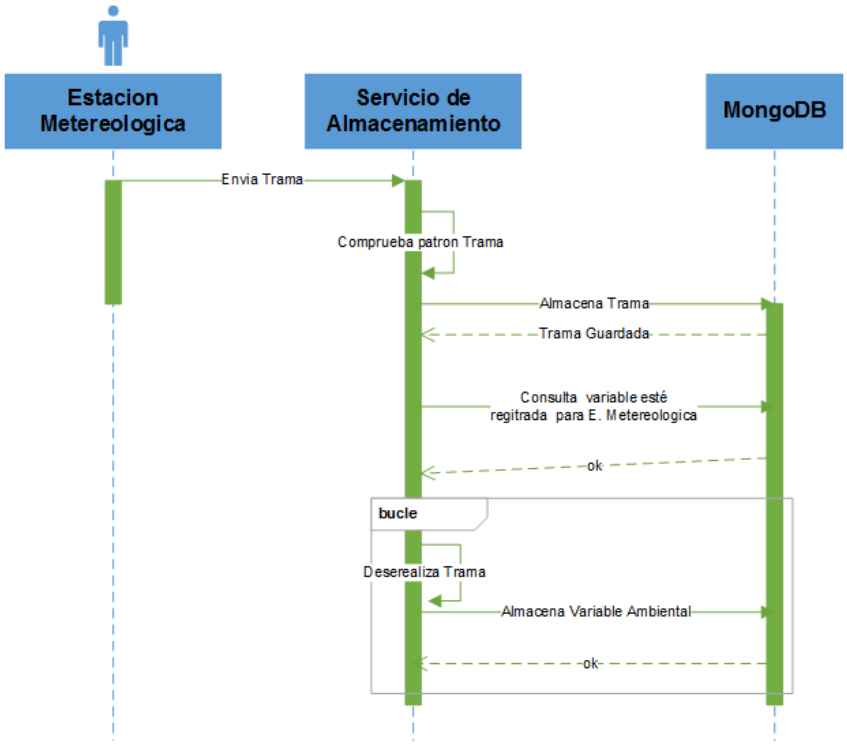


Figura 14: Diagrama de secuencia para el componente de almacenamiento en MongoDB
Fuente: Elaborado por los autores.

Los principales archivos para la ejecución de este componente son **appEstacion.js** y **servicioUtpl.js.**

Archivo appEstacion.js (conexión al servicio de almacenamiento)

Este archivo es el que da inicio al servicio de almacenamiento con su ejecución. La lógica de este archivo es la de realizar la conexión al bróker MQTT y subscribirse al tópico **plataforma/utpl/#;** así como el de trasferir los parámetros necesarios a los archivos **servicioSos.js** y **servicioUtpl.js**. En la figura 15, desde la línea 69 hasta la 78, se puede observar el código que se implementó para realizar la conexión al bróker MQTT.

```
69. client = mqtt.connect('mqtt://' + argv.host + ":" +argv.port, {
70.     username: argv.user,
71.     password: argv.password,
72.     protocolId: 'MQIsdp',
73.     protocolVersion: 3,
74.     keepalive: 6000,
75.     reconnectPeriod: 1000,
76.     connectTimeout: 1000,
77.     clientId: argv.user + argv.clientId
78. });
```

Figura 15: Código fuente - appEstacion.js – conexión al servicio de almacenamiento
Fuente: Elaborado por los autores.

La función **`mqtt.connect`** permite parametrizar la conexión con el bróker MQTT. En esta sección de código se configura el host y el puerto al cual se va a conectar, los cuales son asignados en las variables **`argv.host`** y **`argv.port`**. Asimismo, se asigna la versión del protocolo MQTT con la cual se va a trabajar, y el tiempo que se mantendrá abierta la conexión. Estas funciones se implementan utilizando la librería *MQTT.js* versión 2.4.0, que se instala mediante el gestor de librerías de NodeJs conocido como NPM (Node Package Manager).

```
123. client.on('connect', function (connack) {
124.     logger.silly('Cliente conectandose al rabbit');
125.     client.subscribe('plataforma/sensor/#', { qos: 1 });
126. });
```

Figura 16: Código fuente - appEstacion.js - función *connect* para conexión
Fuente: Elaborado por los autores.

Una vez que se ha establecido los parámetros necesarios para la conexión al bróker, el paso siguiente es suscribirse a un tópico. Para esto la librería MQTT dispone de la función **`client.on`**, (se la puede apreciar en la línea 125 de la figura 16), con la cual el cliente se suscribe al tópico **`plataforma/sensor/#`**. Se ha utilizado el comodín # para que el servicio pueda subscribirse a múltiples tópicos a la vez y sea capaz de poder recibir todos los mensajes que envían las estaciones meteorológicas.

```
107. client.on('message', function (topic, message) {
108.     logger.silly('Mensaje recibido mqtt');
109.     logger.info('Recibiendo datos de sensor', { topico: topic, dato:  message});
110.     var mensajeJson;
111.     try {
112.         mensajeJson = JSON.parse(message);
113.         servicioEstacion.sincronizar(mensajeJson, topic);
114.         servicioSos.sincronizar(mensajeJson, topic);
115.
116.     } catch (e) {
117.         logger.error(e.stack);
118.     }
119.
120.   //client.end();
121. });
```

Figura 17: Código fuente - appEstacion.js – suscripción al tópico
Fuente: Elaborado por los autores.

Luego, cuando se estable la conexión con el bróker y subscrito al tópico, el siguiente paso es recibir el mensaje. Este proceso se realiza mediante la función `message` (ver figura 17), que recibe el mensaje y lo pasa al servicio llamado `servicioEstacion`. Este servicio invoca a la función `sincronizar`, la cual recibe como entrada los parámetros `mensajeJson` y `topic`, los cuales corresponden al mensaje que envía determinada estación y el tópico al cual estaba subscrito la misma.

Archivo servicioUtpl.js (almacenamiento en MongoDB)

Este servicio es el encargado en des-serializar los mensajes y almacenarlos en las colecciones que se han asignado en la base de datos MongoDB. La primera acción que se debe realizar es la de recibir el mensaje, para esto se ha implementado la función **`sincronizar`**. Dicha función recibe los parámetros **`mensaje`** y **`tópico`** que serán enviados desde el servicio **`appEstacion.js`** y a su vez los pasa a las funciones **`registrarTrama`** y **`actualizarEstacion`**; como se puede observar en la figura 18 (desde la línea 127 hasta la 130)

```
127. ServicioEstacion.prototype.sincronizar = function (mensaje, topico) {
128.   ServicioEstacion.prototype.registrarTrama(mensaje, topico);
129.   ServicioEstacion.prototype.actualizarEstacion(mensaje, topico);
130. };
```

Figura 18: Código fuente - servicioUtpl.js (1)
Fuente: Elaborado por los autores.

En la figura 19 (líneas 28 y 29), se puede apreciar que la función **`registrarTrama`** recibe los parámetros correspondientes a mensaje y tópico. Esta función comprueba que las tramas recibidas cumplan un patrón que se definido para los mensajes que trasmitirán las estaciones. Si el mensaje cumple el patrón, la función lo almacena en la colección llamada **`Trama`**, en la base de datos MongoDB. Caso contrario, se hace el manejo de error correspondiente.

```
27. ServicioEstacion.prototype.registrarTrama = function (mensaje, topico) {
28.   var trama = new ServicioEstacion.prototype.TramaModel({id:mensaje.id, topic: topico, fechaRegistro: Date.now(),
29.   fechaPuplicacion: moment(mensaje.m, FORMATO_FECHA_PUBLICACION) , dato:  mensaje});
30.   trama.save(function (err) {
31.     if (err){ // ...
32.       logger.error(err.stack);
33.     }
34.   });
35. };
```

Figura 19: Código fuente - servicioUtpl.js (2)
Fuente: Elaborado por los autores.

Por otro lado, la función **`actualizarEstacion`** recibe los parámetros **`mensaje`** y **`tópico`**, similar a la lógica anteriormente descrita. Esta función, a través del **`mensaje.id`** de la estación, busca todos los registros que le corresponden a dicha estación en la colección **`VariableEstacion`**. para luego actualizar dichos registros acorde a las variables que vienen en el mensaje. Se puede visualizar esta lógica implementada en la figura 20, desde la línea 104 a la 113.

```
/**Actulizacion de valores sobre estacion*/
ServicioEstacion.prototype.actualizarEstacion = function (mensaje, topico) {
  ServicioEstacion.prototype.EstacionModel.findOne({ codigo: mensaje.id }, function (err, doc){
    if(doc != null){
      console.log('actualizacion estacion');
      console.log('variables estacion remota: %j', mensaje.var);
      for(var variableAmbiental in mensaje.var){
          ServicioEstacion.prototype.buscarVariableAmbiental(variableAmbiental, mensaje.var[variableAmbiental],
            mensaje.id, mensaje.m,
            //estacion objeto base
            doc);
      };

      doc.ultima_actualizacion = moment();
      doc.latitud = mensaje.la == null ? doc.latitud : mensaje.la;
      doc.longitud = mensaje.lo == null ? doc.longitud : mensaje.lo;

      doc.save(function (err) {
        if (err){ // ...
          logger.error(err.stack);
        }
      });
    }
  });
};
```

Figura 20: Código fuente – servicioUtpl.js (3)
Fuente: Elaborado por los autores.

3.5.1.2 Componente de almacenamiento en SOS.

Este componente permite transformar los mensajes que se recibe desde las estaciones meteorológicas en formato Json, a un mensaje que cumpla el estándar SOS (Sensor Observation Service). Los mismos que son re-transmitidos a través de un servicio REST que expone 52North (Implementación SOS). Es importante mencionar que 52North dispone de una lógica de almacenamiento a través del gestor de base de datos PostgreSQL. En la figura 21, se puede visualizar un diagrama de secuencia referente a este proceso.

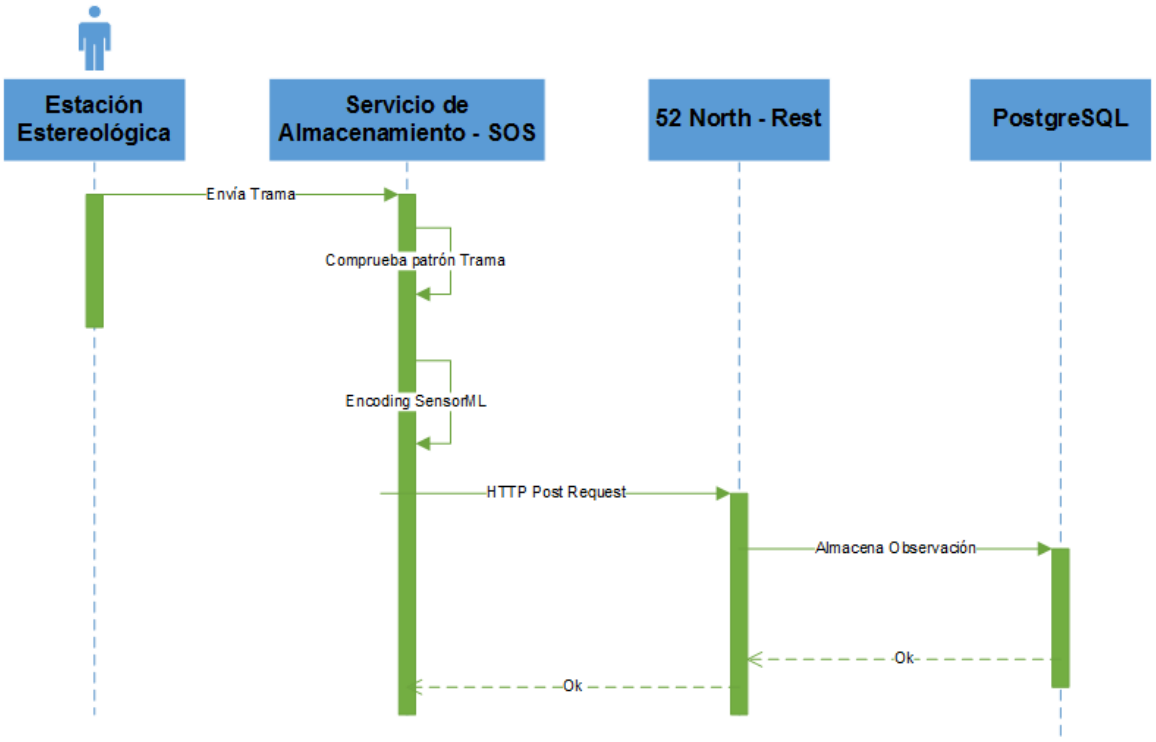


Figura 21: Diagrama de secuencia SOS
Fuente: Elaborado por los autores.

El archivo que implementa esta lógica correspondiente al componente de almacenamiento en SOS es **`servicioSos.js.`** En la tabla 10, se muestra la transformación de un mensaje en formato Json al formato SOS.

Tabla 10: Transformación de mensaje en formato Json a formato SOS

Mensaje recibido desde las estaciones	Mensaje SOS

Mensaje recibido desde las estaciones

```
{
  "id":"1",
  "m":"",
  "var": [
    "h": 18.0
  ]
}
```

Mensaje SOS

```
{
  "request": "InsertObservation",
  "service": "SOS",
  "version": "2.0.0",
  "offering": "http://www.utpl.edu.ec/estacion/offering/1",
  "observation": {
    "identifier": {
      "codespace": "http://www.opengis.net/def/nil/OGC/0/unknown",
      "value": "http://www.utpl.edu.ec/estacion/observation/1485946080/1485946080";
    },
    "type": "http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_Measurement",
    "procedure": "http://www.utpl.edu.ec/estacion/procedure/1",
    "observedProperty": "http://mmisw.org/ont/cf/parameter/air_temperature",
    "phenomenonTime": "2017/02/01 10:54",
    "resultTime": "2017/02/01 10:54",
    "result": {
      "uom": "Cel",
      "value": 18
    },
    "featureOfInterest": {
      "identifier": {
        "value": "http://www.utpl.edu.ec/featureOfInterest/2",
        "codespace": "http://www.opengis.net/def/nil/OGC/0/unknown"
      },
      "name":
      {
        "value": "loja:zapotillo:mangahurco:2",
        "codespace": "http://www.opengis.net/def/nil/OGC/0/unknown"
      },
      "sampledFeature": [
        "http://www.utpl.edu.ec/feature/2"
      ],
      "geometry": {
        "type": "Point",
        "coordinates": [
          -4.156003,
          -80.4320669
        ],
        "crs": {
          "type": "name",
          "properties": {
            "name": "EPSG:4326"
          }
        }
      }
    }
  }
}
```

Fuente: Elaborado por los autores.

Archivo servicioSos.js (conexión a 52North)

La función en la que se implementa la lógica de transformación de formato se llama **`registrarEnSOS`**, la cual recibe como parámetros el mensaje y el tópico al cual publico la estación. Esta implementación, a nivel de estación, des-serializa el mensaje recibido y busca los parámetros: featureOfInterest, procedure, offering, y observation, en el archivo de configuración `configSos.js`. En el caso de no encontrar estos parámetros configurados para una estación, la publicación SOS no se lleva a cabo. Lo anteriormente mencionado, se lo puede visualizar en la figura 22, en la línea 4. Asimismo, a nivel de variables, la función **`registrarEnSOS`** busca la correspondiente configuración para cada una de las variables; esto se evidencia desde la línea 8 a la 10 de la figura 22. Una vez que se realiza este proceso, se hace una petición POST HTTP al servicio REST de 52North, lo que puede ser visualizado en la línea 25 a la 45 de la figura 22.

```
01. ServicioSos.prototype.registrarEnSOS = function (mensaje, topico) {
02.   console.log('codigo de estacion: %j', mensaje.id);
03.   console.log('variables estacion remota: %j', mensaje.var);
04.   var estacionConfig = _.find(configSos.estaciones, function(item) { return item.code == mensaje.id;});
05.   var fechaRegistroTrama = moment();
06.   if(estacionConfig){
07.     for(var variableAmbiental in mensaje.var){
08.       var varConfig = _.find(estacionConfig.vars, function(item) {
09.         return item.codePlatform == variableAmbiental;
10.       });
11.
12.       if(varConfig){
13.         var observacion = _.clone(estacionConfig.configuration.observation);
14.         //generar un identificador para la observacion
15.         observacion.identifier.value =  "http://www.utpl.edu.ec/estacion/observation/" + moment().unix() + "/" + _.uniqueId([prefix='']);
16.         //seleccionar la propiedad observada
17.         observacion.observedProperty = varConfig.observedProperty;
18.         observacion.phenomenonTime = moment(mensaje.m, FORMATO_FECHA_PUBLICACION);
19.         observacion.resultTime =  fechaRegistroTrama;
20.         observacion.result = {
21.           "uom": varConfig.uom,
22.           "value": mensaje.var[variableAmbiental]
23.         };
24.
25.             request({
26.                 url: configSos.pathSos, //URL to hit
27.                 method: 'POST',
28.                 headers: { //We can define headers too
29.                     'Content-Type': 'application/json;charset=UTF-8'
30.                 },
31.                 //Lets post the following key/values as form
32.                 json: {
33.                     "request": "InsertObservation",
34.                     "service": "SOS",
35.                     "version": "2.0.0",
36.                     "offering": estacionConfig.configuration.offering,
37.                     "observation": observacion
38.                 }
39.             }, function(error, response, body){
40.                 if(error) {
41.                     logger.warn(error);
42.                 } else {
43.                     console.log(response.statusCode, body);
44.             }
45.             });
46.       }
47.
48.     }
49.
50.   }
51. };
```

Figura 22: Código fuente servicioSos.js
* Para más detalle del archivo de configuración `configSos.js`, revisar el anexo #5 Parametrización SOS.
Fuente: Elaborado por los autores.

3.5.2 Subsistema de gestión de estaciones meteorológicas y variables medioambientales (SGEMVM)

En este subsistema se ha definido una arquitectura en tres capas. El mismo que está diseñado para que los usuarios registrados puedan gestionar la información referente a estaciones meteorológicas y sus variables medioambientales, independientemente del lugar donde se encuentren, aprovechando las ventajas que ofrece Internet. Partiendo de los requerimientos funcionales, se ha definido un diagrama de casos de uso, el cual proporciona una perspectiva global de las funcionalidades del subsistema (ver figura 23).

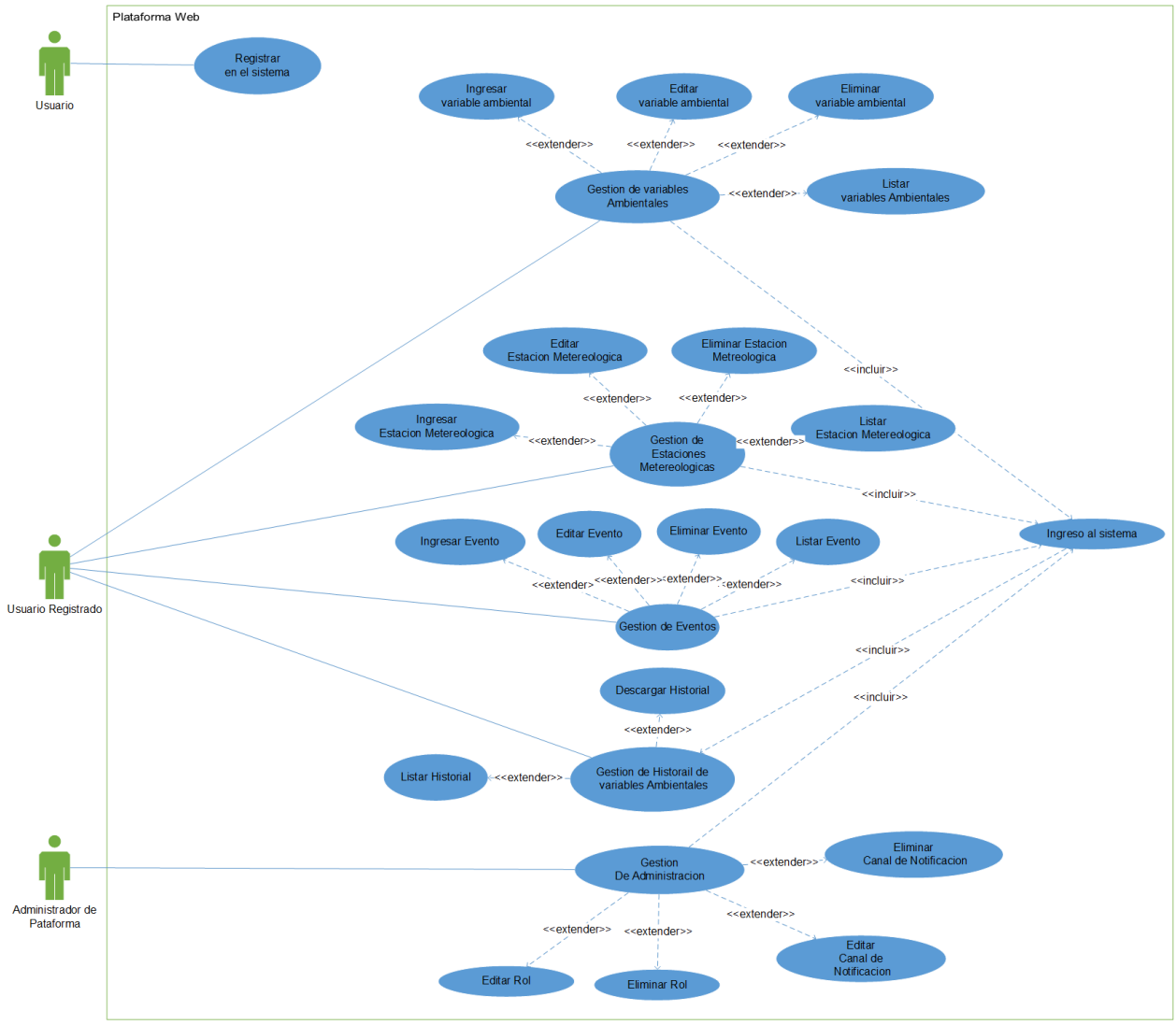


Figura 23: Diagrama de casos de uso para subsistema de gestión de estaciones meteorológicas y variables medioambientales
* Para más detalle de las funcionalidades de este subsistema, revisar el anexo #3 Parametrización SOS.
Fuente: Elaborado por los autores.

Como parte de desarrollo de este subsistema se implementaron varios servicios REST, los cuales permiten la interacción entre usuario cliente y el sistema; para tener un proceso controlado de acceso a datos. A continuación, se describirá dos componentes que forman parte de este subsistema, los mismos que son: notificación de alertas y descarga de historial de una variable medioambiental.

3.5.2.1 Componente de notificación de alertas.

En este componente fue necesario implementar el patrón de diseño *Chain of Responsibility* o "cadena de responsabilidades", cuyo enfoque consiste en definir una serie de clases las cuales van a recibir una petición y cada una de ellas evalúa sí es su responsabilidad atender la petición o no. Este se implementa cuando se programa una alerta por condición o por inactividad. Por lo tanto, cuando se ejecuta la consola de alertas, el manejador de responsabilidades procede a gestionar cada una de las alertas a través de la cadena implementada. Particularmente, para este subsistema se dispone de la siguiente jerarquía de responsabilidades: primero alertas por condición, y segundo alertas por inactividad.

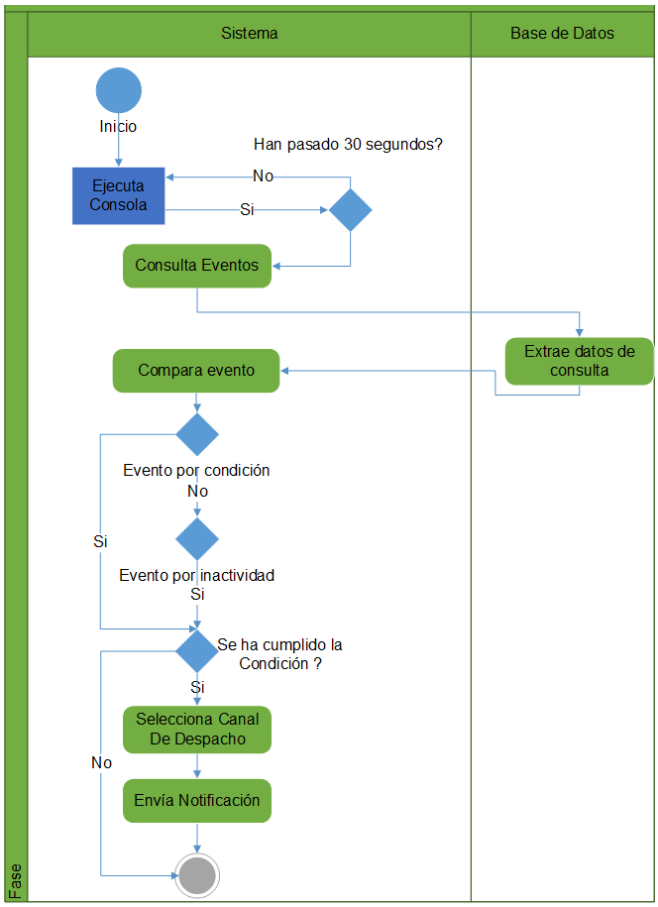


Figura 24: Diagrama de actividad - notificación de alertas
Fuente: Elaborado por los autores.

Para describir el proceso de “cadena de responsabilidades”, es necesario diseñar un flujo de trabajo, el mismo que está representado en el diagrama de actividades de la figura 24. La implementación del patrón *Chain of Responsibility* se encuentra en el archivo **`ProcesadorAlertasComponent.java`**. La figura 25 ilustra el patrón implementado.

```java
@Component
public class ProcesadorAlertasComponent {

    @Autowired
    private ProcesadorByCondicional procesadorByCondicional;

    @Autowired
    private ProcesadorByInactividad procesadorByInactividad;

    public void procesar(ParamProcesador paramProcesador) {
        if(procesadorByCondicional.nextHandler == null){
            procesadorByCondicional.setNextHandler(procesadorByInactividad);
        }

        procesadorByCondicional.receiveRequest(paramProcesador);

    }

}
```

Figura 25: Implementación del patrón *Chain of Responsibility*
Fuente: Elaborado por los autores.

3.5.2.2 Componente de descarga de historial.

Este componente permite a los usuarios descargar el historial de mediciones de una variable medioambiental, la cual está asociada a una estación meteorológica, dentro de un rango de tiempo especificado por el usuario. En la figura 26, se puede observar el flujo de trabajo correspondiente a este proceso.

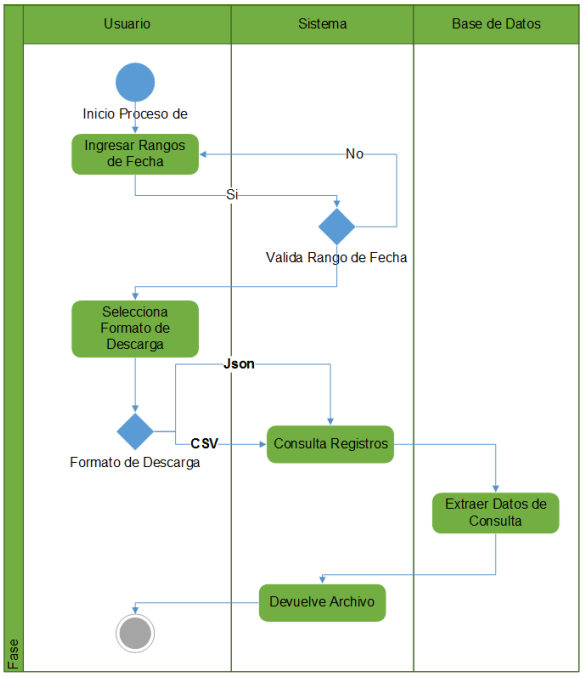


Figura 26: Diagrama de actividad – módulo de descarga
Fuente: Elaborado por los autores.

La descarga de archivos se la puede realizar en dos formatos Json y Csv; como parte del desarrollo de este componente, se generaron dos servicios REST.

Servicio REST: Descarga en formato Json

En la figura 27, se puede observar la implementación del servicio de descarga en formato Json. Este servicio requiere como parámetros de entrada: un rango de fecha para la consulta, código de variable medioambiental, y código de la estación meteorológica. Una vez que se han establecido los parámetros, el servicio retornará un listado con todos los registros que se encuentren almacenados.

```
01.  /**
02.   * POST /descargaJson -> .
03.   * @throws JsonProcessingException
04.   */
05.  @RequestMapping(value = "/descargarJson", method = RequestMethod.POST, produces = MediaType.APPLICATION_JSON_VALUE)
06.  @Timed
07.  public ResponseEntity<VariableDescargarDTO> descargarJson (
08.          @Valid @RequestBody EstacionRequest estacionRequest)
09.          throws URISyntaxException, JsonProcessingException  {
10.      log.debug("REST request to save VariableEstacion : {}", estacionRequest);
11.      if (estacionRequest.getCodigoEstacion() == null
12.              || estacionRequest.getCodigoVariable() == null
13.              || estacionRequest.getDesdeFecha() == null
14.              || estacionRequest.getHastaFecha() == null) {
15.          return ResponseEntity
16.                  .badRequest()
17.                  .header("Failure",
18.                          "A new variableEstacion cannot already have an ID")
19.                  .body(null);
20.      }
21.
22.      Optional<Estacion> estacion = estacionRepository
23.              .findOneByCodigo(estacionRequest.getCodigoEstacion());
24.      Optional<Variable> variable = variableRepository
25.              .findOneByCodigo(estacionRequest.getCodigoVariable());
26.      List<VariableHistorico> summary = estacionService
27.              .busquedaHistorico(estacionRequest);
28.      VariableDescargarDTO variableDTO =  new VariableDescargarDTO(variable,
29.              estacion, summary);
30.
31.      return new ResponseEntity<>(variableDTO,HttpStatus.OK);
32.  }
```

Figura 27: Código fuente para descarga de historial de datos en formato Json
Fuente: Elaborado por los autores.

Servicio REST: Descarga en formato Csv

En la figura 28, se puede observar el código que permite realizar la descarga en formato Csv. El cual requiere: el rango de fechas, la variable medioambiental, y la estación meteorológica que se desea consultar; esto devolverá un listado con los valores almacenados.

```java
/**
 * POST /descargaCsv -> .
 * @throws JsonProcessingException
 */
@RequestMapping(value = "/descargarCsv", method = RequestMethod.POST, produces = "text/csv")
@Timed
public ResponseEntity<String> descargarCsv (
        @Valid @RequestBody EstacionRequest estacionRequest)
        throws URISyntaxException, JsonProcessingException  {
    log.debug("REST request to get CSV : {}", estacionRequest);
    if (estacionRequest.getCodigoEstacion() == null
            || estacionRequest.getCodigoVariable() == null
            || estacionRequest.getDesdeFecha() == null
            || estacionRequest.getHastaFecha() == null) {
        return ResponseEntity
                .badRequest()
                .header("Failure",
                        "A new variableEstacion cannot already have an ID")
                .body(null);
    }


    List<VariableHistorico> summary = estacionService
            .busquedaHistorico(estacionRequest);

    log.debug("items para csv: {}", summary.size());

    final String cabeceraCsv = "estacion_codigo,variable_codigo,fecha_registro,fecha_publicacion,valor\n";
    final String csv = summary.stream()
            .map(item ->
              estacionRequest.getCodigoEstacion() + "," + estacionRequest.getCodigoVariable() + "," +
              item.getFechaRegistro() + "," + item.getFechaPublicacion() +"," + item.getValor())
            .collect(Collectors.joining("\n"));

    return new ResponseEntity<>(cabeceraCsv.concat(csv),HttpStatus.OK);
}
```

Figura 28: Código fuente para descarga de historial de datos en formato CSV
Fuente: Elaborado por los autores.

3.5.3 Subsistema de visualización de datos

Este subsistema es un sitio web, en el cual que no se requiere el registro de usuarios para acceder a la información dispuesta. En la figura 29, se tiene una vista previa de este subsistema.

Figura 29: Subsistema de visualización de datos
Fuente: Elaborado por los autores.

Una de las principales características del sitio web, es que dispone de un mapa con la ubicación y la información del último dato adquirido correspondiente a cada una de las estaciones meteorológicas, que se encuentran distribuidas en las diferentes regiones del Ecuador. Para llevar a cabo la implementación de este subsistema se desarrolló un servicio REST que provea esta información al sitio web. Un aspecto importante a mencionar es que los marcadores que indican la ubicación de las estaciones meteorológicas han sido agrupados de acuerdo a la proximidad entre ellas. Esto permite que se visualice presente de una manera más armoniosa en el mapa. En la figura 30 se puede ver el resultado de esta característica.

Figura 30: Clusterización de marcadores meteorológicos – componente de visualización de datos
Fuente: Elaborado por los autores.

En la figura 31, se muestra un ejemplo en el cual en la parte izquierda se presentan los marcadores de geolocalización aglutinados en el mapa; mientras que a la derecha de la imagen se ha aplicado el algoritmo de clusterización de marcadores. Se puede evidenciar como estos se han agrupado, mientras mayor el nivel de acercamiento al mapa dichos marcadores se irán desagrupando progresivamente.

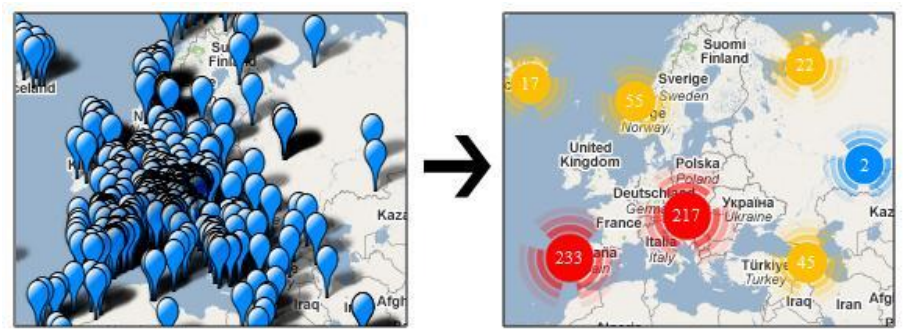


Figura 31: Clusterización de estaciones
Fuente: Tomado del sitio web Handling Large Amounts of Markers in Google Maps (Svennerberg , 2009)

3.6 Adecuación del entorno de desarrollo

El entorno de desarrollo es el conjunto de procesos y herramientas de programación utilizados para crear el programa o producto de software. Actualmente, se maneja el concepto de entorno de desarrollo integrado, el cual es aquel en el que los procesos y las herramientas están coordinados para proporcionar a los desarrolladores una interfaz ordenada y una vista conveniente para el proceso de desarrollo (Rouse, 2007). De acuerdo a Xamarin Inc (2015), integración continua es una práctica de ingeniería de software en la que una compilación automatizada compila y, opcionalmente, prueba una aplicación cuando el código es agregado o cambiado por los desarrolladores en el repositorio de control de versiones del proyecto. Consecuentemente, para el desarrollo del presente proyecto se dispuso entorno de desarrollo expuesto en la figura 32.

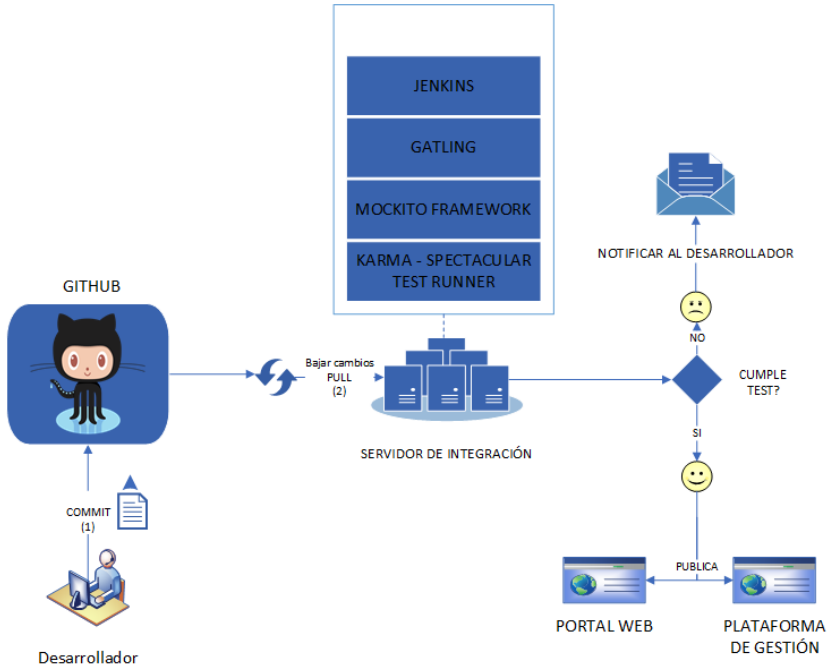


Figura 32: Flujo de desarrollo de software utilizado dentro de un entorno de integración continuo
Fuente: Elaborado por los autores.

A continuación, en la tabla 11 se presenta el conjunto de herramientas que conforman el entorno de desarrollo integrado del proyecto de desarrollo de software.

Tabla 11: Conjunto de herramientas que conformaron el entorno de desarrollo integrado

Herramienta	Descripción
Github	Plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git.
Jenkins	Es un software código abierto basado en Java, el cual permite realizar integración continua en el proceso de desarrollo de software.
Gatling	Sofware código abierto para pruebas de carga basado en el lenguaje Scala.
Mockito	Mockito, un framework, es una biblioteca basada en JAVA que es utilizada para

framework	realizar pruebas de unidad en aplicaciones JAVA
Karma – Spectacular test runner	Es una herramienta sencilla que permite ejecutar código JavaScript en varios navegadores reales.
Atom	Es un editor de código abierto para macOS, Linux y Windows con soporte para plug-ins escritos en Node.js y Git incorporado.

Fuente: Elaborado por los autores.

En la tabla 12, se describe los repositorios utilizados para almacenar el código fuente generado para los diferentes subsistemas que conforman la solución propuesta.

Tabla 12: Repositorios utilizados para almacenar el código fuente

Nombre	**Url**	**Subsistema**
node.utpl.edu.ec	https://github.com/fdquinones/lotplatform/node.utpl.edu.ec	Subsistema de almacenamiento
Platform	https://github.com/fdquinones/lotplatform/platform	Subsistema de gestión de estaciones meteorológicas
Iotportal	https://github.com/fdquinones/iotportal	Subsistema de visualización de datos

Fuente: Elaborado por los autores.

CAPÍTULO 4: PRUEBAS

4.1 Introducción

Una vez que se ha concluido con la implementación de la arquitectura propuesta, el siguiente paso es validar lo que se ha desarrollado. Es por ello que, en este capítulo se detalla la fase de validación y pruebas que se aplicaron, lo cual permitió asegurar el correcto funcionamiento de la solución desarrollada.

4.2 Plan de pruebas para la solución desarrollada

El presente plan de pruebas ha sido elaborado y diseñado en base a los lineamientos de la norma IEEE 829. Dicho plan de pruebas contiene los siguientes elementos: propósito y alcance del plan, especificación del ambiente de pruebas, ejecución y resultado de pruebas.

4.2.1 Propósito y alcance.

El propósito de este plan de pruebas es descubrir defectos o inconsistencias que podría tener el sistema de almacenamiento y visualización de estaciones meteorológicas, como solución de software desarrollada en el presente proyecto de investigación. Específicamente, se requiere validar temas referentes a integridad de datos, funcionalidad, interfaz de usuario, y capacidad máxima de servicio.

4.2.2 Objetivos del plan de pruebas.

Los objetivos que se han definido para este plan de pruebas son los siguientes:

- Evaluar la confiabilidad del sistema desarrollado.
- Identificar los elementos que se van a probar.
- Describir la estrategia de pruebas que se van a seguir.
- Identificar los recursos necesarios para llevar a cabo el proceso de pruebas.
- Presentar los resultados obtenidos en las actividades de prueba.

4.2.3 Especificación del ambiente de pruebas.

El ambiente de pruebas está constituido por dos tipos de insumos: lógico y físico. El insumo lógico se refiere a la infraestructura de software integrada que permita la manipulación de información en un ambiente de pruebas; para lo cual se ha creado una réplica de los componentes del sistema del ambiente de producción. Mientras que el insumo físico se refiere a los equipos de hardware

utilizados para realizar la ejecución de las pruebas. A continuación, en la tabla 13 y 14 se detalla las características de los insumos que constituyeron el ambiente de pruebas.

Tabla 13: Insumo lógico del ambiente de pruebas

INSUMO LÓGICO		
Tipo		**Descripción**
Base de datos	Mongo DB	Bases de datos utilizada por el servicio de almacenamiento y la plataforma de administración de estaciones meteorológicas. *IP:* 200.0.29.38 *Database:* Platform *Username:* root *Password:* ***** *Port:* 28017
Bróker MQTT	RabbitMQTT	Implementación del protocolo MQTT como protocolo de comunicación publicación / subscripción. *URL*: http://200.0.29.38:8080/rabbit/ *Username:* adminmqtt *Password:* ****** *Port:* 1883
SOS	52North	Implementación del estándar SOS para el almacenamiento de variables medioambientales. *URL:* http://200.0.29.38:8080/52n-sos/ *Usuario:* admin
Plataforma de Administración		Plataforma de administración de estaciones meteorológicas. *URL*: 200.0.29.38:8080/platform/#/estacions *Username:* admin *Password:* ****** *Port:* 8080
Portal		Landing page de la solución construida. *URL:* 200.0.29.38:8080/cedia/
Twilio		Plataforma Cloud para el envio masivo de SMS. *URL:* https://www.twilio.com/

Fuente: Elaborada por los autores.

Tabla 14: Insumo físico del ambiente de pruebas

INSUMO FÍSICO		
Equipo	**Sistema Operativo**	**Características**
Servidor Cedia	Win 2012 r2 64 bits	Ram: 8GB. Disco duro: 120GB Procesador: Xeon de 2 núcleos de 2.6 Ghz.
Equipo para ejecución de pruebas	Win 11 64 bits	Ram: 16GB. Disco Duro: 500GB Procesador: Core i7 2.8 Ghz

Fuente: Elaborada por los autores.

4.2.4 Pruebas.

Para la ejecución del plan de pruebas, se han diseñado y ejecutado cuatro tipos de pruebas: integridad de datos, funcionalidad, interfaz de usuario, y pruebas de carga. La tabla 15, muestra el tipo de pruebas, y su función. Asimismo, es importante mencionar que para cada tipo de pruebas se plantearon objetivos, estrategias, y escenarios de prueba.

Tabla 15: Tipos de pruebas a ejecutar

Tipo de Pruebas	Función
Pruebas de integridad de datos	Verificar la exactitud, calidad y funcionalidad de los datos almacenados para garantizar la integridad y fiabilidad de la información almacenada.
Pruebas de funcionalidad	Verificar que la solución de software desarrollada funciona correctamente de acuerdo con las especificaciones de diseño.
Pruebas de interfaz de usuario	Interactuar con la interfaz gráfica de usuario de la aplicación para validar las propiedades y estado de sus elementos.
Pruebas de carga	Someter los componentes de software desarrolladas a un nivel de trabajo que permita determinar los límites de rendimiento.

Fuente: Elaborada por los autores.

4.2.4.1 Pruebas de integridad de datos.

El objetivo de las pruebas de integridad de datos es garantizar la calidad de la información almacenada en la base de datos utilizada por la solución de software a testear. Este tipo de pruebas se refiere a la ejecución de un proceso manual o automatizado que permita verificar la exactitud, calidad y funcionalidad de los datos almacenados (Tutorials Point, 2016). Para ello, es necesario comprobar que el sistema almacena los datos de manera consistente, sin que se vean comprometidos por la actualización, la restauración o el procesamiento de recuperación. De acuerdo a Ghahrai (2009), las pruebas de integridad tienen la intención de descubrir fallas de diseño que pueden resultar en corrupción de datos, acceso a datos no autorizados, falta de integridad de datos en varias tablas y falta de rendimiento de transacción adecuado.

Objetivos:

- Asegurar que las tramas de datos provenientes de las estaciones meteorológicas se almacenen de manera íntegra en las colecciones de MongoDB.
- Asegurar que las tramas de datos provenientes de las estaciones meteorológicas se

almacenen de manera íntegra en el estándar SOS dentro de 52North (PostgreSQL).

Estrategia: Revisar manualmente los registros en la base de datos para asegurar que la información ha sido almacenada correctamente, en consecuencia, la recuperación de datos sea coherente.

Escenarios: Se diseñaron cuatro escenarios para validar la integridad de datos (ver tabla 16)

Tabla 16: Escenarios de pruebas de integridad

		Descripción	Precondiciones	Pasos	Resultado Esperado
1	ALMACENAMIENTO EN MONGODB	Verificar que la trama enviada por una estación meteorológica se almacene correctamente, cuando la trama se encuentra en un formato Json válido.	– La conexión entre estación meteorológica y bróker se encuentra correctamente configurada. – Servicio de almacenamiento se encuentra disponible y correctamente configurado. – La trama debe estar en formato Json. – La trama debe pertenecer a una estación previamente creada y las variables deben estar previamente creadas.	1. Publicar trama desde estación meteorológica. 2. Rabbit MQTT realiza el enrutamiento de la trama hacia el subscriptor SERVICIO DE ALMACENAMIENTO. 3. Servicio de almacenamiento valida la trama. 4. Servicio de almacenamiento guarda la trama.	El SERVICIO DE ALMACENAMIENTO, registra la trama correctamente en las colecciones: Estacion, Variable_Estacion, Variable_Historico
2		Verificar que la trama enviada por una estación meteorológica no sea almacenada en la base de datos, cuando la trama se encuentra en un formato invalido.	– La conexión entre estación meteorológica y bróker se encuentra correctamente configurada. – El servicio de almacenamiento se encuentra disponible y correctamente configurado. – La trama se encuentra en un formato invalido.	1. Publicar trama desde estación meteorológica. 2. Rabbit MQTT realiza el enrutamiento de la trama hacia el subscriptor SERVICIO DE ALMACENAMIENTO. 3. Servicio de almacenamiento valida la trama. 4. Servicio de almacenamiento rechaza la trama.	El SERVICIO DE ALMACENAMIENTO, NO registra la trama en las colecciones: Estacion, Variable_Estacion, Variable_Historico

3		Verificar que la trama enviada por una estación no definida, no se registre en la base de datos.	– La conexión entre estación meteorológica y bróker se encuentra correctamente configurada. – El Servicio de almacenamiento se encuentra disponible y correctamente configurado. – La trama debe estar en formato Json. – La trama NO pertenecer a una estación previamente creada y las variables NO están creadas**.**	1. Publicar trama desde estación meteorológica. 2. Rabbit MQTT realiza el enrutamiento de la trama hacia el subscriptor SERVICIO DE ALMACENAMIENTO. 3. Servicio de almacenamiento valida la trama. 4. Servicio de almacenamiento rechaza la trama.	El SERVICIO DE ALMACENAMIENTO, NO registra la trama en las colecciones: Estacion, Variable_Estacion, Variable_Historico
4	ALMACENAMIENTO EN SOS	Verificar que la trama enviada por una estación meteorológica se almacene acorde al estándar SOS.	– La conexión entre estación meteorológica y bróker se encuentra correctamente configurada. – Servicio SOS se encuentra disponible y correctamente configurado. – La trama en formato Json. – La trama debe pertenecer a una estación previamente creada y las variables deben estar previamente creadas. – Instancia 52north correctamente instalada y configurada.	1. Publicar trama desde estación meteorológica. 2. Rabbit MQTT realiza el enrutamiento de la trama hacia el subscriptor SERVICIO SOS. 3. Servicio SOS valida la trama. 4. Servicio SOS genera petición REST hacia la instancia 52North. 5. Instancia 52North almacena correctamente la trama.	El SERVICIO SOS almacena correctamente la trama en la base de datos PostgreSQL, acorde al formato O&M.

Fuente: Elaborada por los autores.

Resultados: A continuación, la tabla 17 muestra los resultados correspondientes a las pruebas de integración de datos. Por consiguiente, las pruebas de integridad han sido superadas con satisfacción.

Tabla 17: Resultados de pruebas de integridad

Escenario	Componente	Descripción	Resultado Validación
1	Almacenamiento en MongoDB	Verificar que la trama enviada por una estación meteorológica se almacene correctamente, cuando la trama se encuentra en un formato JSON válido.	✓
2		Verificar que la trama enviada por una estación meteorológica no sea almacenada en la base de datos, cuando la trama se encuentra en un formato invalido.	✓
3		Verificar que la trama enviada por una estación no definida, no se registre en la base de datos	✓
4	Almacenamiento en SOS	Verificar que la trama enviada por una estación meteorológica se almacene acorde al estándar SOS.	✓

Fuente: Elaborada por los autores.

4.2.4.2 Pruebas de funcionalidad.

Las pruebas de funcionalidad tienen como propósito verificar que una aplicación de software funciona correctamente de acuerdo con las especificaciones de diseño. Durante las pruebas de funcionalidad se comprueba que las principales funciones de la aplicación (como entrada de texto, funciones de menú, instalación y configuración, etc.) realicen las acciones que se desea de acuerdo a los requerimientos funcionales especificados durante el análisis de la solución (ISTQBExamCertification, 2016).

Objetivo: Validar que los requerimientos funcionales se cumplan a cabalidad dentro de los subsistemas de: gestión SGEMVM, y visualización de datos.

Estrategia: Se realiza la validación de los componentes de software correspondientes al subsistema para garantizar que las funcionalidades de la misma cubran los requerimientos funcionales establecidos por el usuario cliente. Consecuentemente, se manipulará el sistema utilizando datos válidos y no válidos para verificar lo siguiente: (a) se obtienen los resultados esperados cuando se utilizan datos válidos, y (b) cuando se utilizan datos no válidos se muestran los mensajes de error o advertencia adecuados.

Escenarios: Se diseñaron veintiún escenarios para validar la funcionalidad de la aplicación (ver tabla 18).

Tabla 18: Escenarios de pruebas de funcionalidad

			Descripción	Precondiciones	Pasos	Resultado Esperado
1	GESTIÓN DE ESTACIONES METEREOLÓGICAS Y VARIABLES MEDIOAMBIENTALES	**ADMINISTRACIÓN DE USUARIO**	Verificar la autentificación de un usuario a la plataforma.	– Se tiene creadas las credenciales para el usuario.	1. Ingresar usuario y contraseña. 2. Presionar <Iniciar sesión>.	El sistema debe presentar pantalla de inicio.
2			Verificar que al ingresar credenciales incorrectas el sistema NO permita el ingreso.	– No se tiene credenciales para el usuario.	1. Ingresar usuario y contraseña. 2. Presionar <Iniciar sesión>.	El sistema debe presentar un mensaje indicando que <¡El inicio de sesión ha fallado!>
3			Verificar la asignación de rol a un usuario.	– Se tiene creado el usuario previamente. – Usuario autentificado en el sistema.	1. Ingresar al menú <Administración/Administración de usuarios> 2. Seleccionar usuario y seleccionar <Editar> 3. Asignar nuevo rol al usuario seleccionado. 4. Presionar botón <Guardar>.	El sistema persiste los cambios correctamente.
4		**VARIABLES MEDIOAMBIENTALES**	Verificar la creación de una variable meteorológica.	– Usuario autentificado en el sistema.	1. Ingresar al menú <Variable medioambiental> 2. Presionar el botón <Crear nueva variable> 3. Completar el formulario 4. Presionar el botón <Guardar>	El sistema presenta el mensaje <La variable con el código (), se ha creado correctamente.>
5			Verificar el funcionamiento de la edición de una variable meteorológica.	– Usuario autentificado en el sistema. – Variable meteorológica creada.	1. Ingresar al menú <Variable medioambiental> 2. Selecciona una variable y presiona el botón <Editar> 3. Modificar el formulario. 4. Presionar el botón <Guardar>	El sistema presenta el mensaje <La variable con el código (), se ha modificado correctamente.>
6			Verificar la eliminación de una variable meteorológica.	– Usuario autentificado en el sistema. – Variable meteorológica creada.	1. Ingresar al menú <Variable medioambiental> 2. Seleccionar una variable y presiona en el botón <Borrar> 3. Confirma la eliminación presionando el botón <Borrar>	El sistema elimina de la base de datos la variable, y presenta el mensaje <La variable con código (), ha sido eliminada>

7	GESTIÓN DE ESTACIONES METEREOLÓGICAS Y VARIABLES MEDIOAMBIENTALES	**ESTACIONES METEREOLÓGICAS**	Verificar la creación de una estación meteorológica.	– Usuario autentificado en el sistema.	1. Ingresar al menú <Estaciones> 2. Presionar el botón <Crear nueva estación> 3. Completar el formulario 4. Presionar el botón <Guardar>	El sistema presenta el mensaje <La estación con el código(), se ha creado correctamente.>
8			Verificar el funcionamiento de la edición de una estación meteorológica.	– Usuario autentificado en el sistema. – Estación meteorológica creada.	1. Ingresar al menú <Estaciones> 2. Presionar el botón <Crear nueva estación> 3. Completar el formulario 4. Presionar el botón <Guardar>	El sistema presenta el mensaje <La estación con el código (), ha sido modificado correctamente.>
9			Verificar la eliminación de una estación meteorológica.	– Usuario autentificado en el sistema. – Estación meteorológica creada.	1. Ingresar al menú <Estaciones> 2. Seleccionar una estación y presiona en el botón <Borrar> 3. Modificar el formulario. 4. Confirma la eliminación presionando el botón <Borrar>	El sistema elimina de la base de datos la estación, y presenta el mensaje <La estación con código (), ha sido eliminada correctamente>
10			Verificar la ubicación geográfica de una estación.	– Usuario autentificado en el sistema – Estación meteorológica creada.	1. Ingresar al menú <Estaciones> 2. Seleccionar una estación y presiona en el botón <Ubicar>	El sistema presenta un mapa, dentro del cual se presenta la estación meteorológica representada por el símbolo de torre.
11			Verificar presentación geográfica de un conjunto de estaciones dentro de un mapa de google.	– Usuario autentificado en el sistema – Estaciones meteorológicas creadas.	1. Ingresar al menú <Estaciones> 2. Presionar el botón <Ubicar estaciones>	El sistema presenta un mapa, dentro del cual se presenta un conjunto de estaciones meteorológicas representadas por el símbolo de torre.
12			Verificar la ubicación geográfica de una estación.	– Usuario autentificado en el sistema – Estación meteorológica creada.	1. Ingresar al menú <Estaciones> 2. Seleccionar una estación y presiona en el botón <Ver>	El sistema presenta la información principal de la estación, así como un detalle de las variables medioambientales que están asignadas.
13			Verificar la presentación de registros históricos de una variable correspondiente a los 2 últimos días.	– Usuario autentificado en el sistema – Estación meteorológica creada. – Registros de tramas previamente almacenadas	1. Ingresar al menú <Estaciones> 2. Seleccionar una estación y presiona en el botón <Ver> 3. Seleccionar una variable y presionar el botón <Ver>	El sistema presenta una gráfica lineal de los registros. Además, presenta los registros de valores mínimo y máximo registrados. Adicionalmente, se presenta una tabla detallada de los registros almacenados.

14	GESTIÓN DE ESTACIONES METEREOLÓGICAS Y VARIABLES MEDIOAMBIENTALES	**GESTIÓN DE DESCARGAS**	Verificar la descarga histórica de registros almacenados para una estación y variable, en formato CSV/JSON.	– Usuario autentificado en el sistema. – Estación meteorológica creada. – Variables asignadas a la estación. – Registros de tramas previamente almacenadas	1. Ingresar menú <Estaciones> 2. Seleccionar una estación y presiona en el botón <Ver> 3. Seleccionar una variable y presionar el botón <Descargar> 4. Establecer un rango de fechas y presionar el botón <Descargar> 5. Seleccionar el formato CSV – JSON	El sistema genera un archivo con los datos filtrados acorde al formato seleccionado.
15		**GESTIÓN DE ALERTAS**	Verificar la creación de una alerta para una estación meteorológica.	– Usuario autentificado en el sistema.	1. Ingresar al menú <Estaciones> 2. Seleccionar una estación y presiona en el botón <Ver> 3. Seleccionar una variable y presionar el botón <Alerta> 4. Presionar el botón <Crear nuevo evento> 5. Completar el formulario 6. Presionar el botón <Guardar>	El sistema presenta el mensaje <El evento ha sido configurado correctamente>
16			Verificar la edición de una alerta para una estación meteorológica.	– Usuario autentificado en el sistema. – Alerta creada.	1. Ingresar al menú <Estaciones> 2. Seleccionar una estación y presiona en el botón <Ver> 3. Seleccionar una variable y presionar el botón <Alerta> 4. Seleccionar un evento y presionar el botón <Editar> 5. Modificar el formulario 6. Presionar el botón <Guardar>	El sistema presenta el mensaje <El evento ha sido actualizado correctamente>
17			Verificar la eliminación de una alerta para una estación meteorológica.	– Usuario autentificado en el sistema. – Alerta creada.	1. Ingresar al menú <Estaciones> 2. Seleccionar una estación y presiona en el botón <Ver> 3. Seleccionar una variable y presionar el botón <Alerta> 4. Seleccionar un evento y presionar el botón <Borrar> 5. Confirmar la eliminación presionando el <Borrar>	El sistema presenta el mensaje <El evento ha sido eliminado correctamente>

18	GESTIÓN DE ESTACIONES METEREOLÓGICAS Y VARIABLES MEDIOAMBIENTALES	**GESTIÓN DE ALERTAS**	Verificar la notificación del suceso de una alerta a través del canal SMS.	– Usuario autentificado en el sistema. – Alerta creada. – Usuario previamente a establecido el número de celular en su perfil. – Canal de despacho SMS establecido para el usuario.	1. Configurar adecuadamente el evento para una estación y variable determinada.	El sistema envía un SMS al destinatario notificándole del suceso del evento.
19			Verificar la notificación del suceso de una alerta a través del canal EMAIL	– Usuario autentificado en el sistema. – Alerta creada. – Usuario previamente a establecido el número de celular en su perfil. – Canal de despacho EMAIL establecido para el usuario.	1. Configurar adecuadamente el evento para una estación y variable determinada.	El sistema envía un EMAIL al destinatario notificándole del suceso del evento.
20	**VISUALIZACIÓN DE DATOS**	**ESTACIONES METEOROLÓGICAS**	Verificar que en el portal se presente un mapa de Google con las estaciones meteorológicas.	– Estaciones georreferenciadas.	1. Ingresar al portal web 2. Ubicarse en la sección Estaciones	El portal presentara un mapa con cada una de las estaciones dispuestas acorde a su latitud y longitud; además el sistema no presentara estaciones que no estén georreferenciadas.
21			Verificar que en el portal se presente un gráfico comparativo de la temperatura de diferentes estaciones.	– Estaciones georreferenciadas. – SOS previamente configurado.	1. Ingresar al portal web 2. Ubicarse en la sección SOS	El sistema presenta una gráfica con los valores de las temperaturas registradas en los últimos días.

Fuente: Elaborada por los autores.

Resultados: A continuación, la tabla 19 muestra los resultados correspondientes a las pruebas de funcionalidad, donde son validados veintiún escenarios de prueba. Por consiguiente, las pruebas de funcionalidad han sido superadas con satisfacción.

Tabla 19: Resultados de pruebas de funcionalidad

Escenario		Funcionalidad	Descripción	Resultado Validación
1	GESTIÓN DE ESTACIONES METEOROLÓGICAS Y VARIABLES MEDIOAMBIENTALES	ADM. USUARIO	Verificar la autentificación de un usuario a la plataforma.	✓
2			Verificar que al ingresar credenciales incorrectas el sistema NO permita el ingreso.	✓
3			Verificar la asignación de rol a un usuario.	✓
4		VARIABLES MEDIOAMBIENTALES	Verificar la creación de una variable meteorológica.	✓
5			Verificar el funcionamiento de la edición de una variable meteorológica.	✓
6			Verificar la eliminación de una variable meteorológica.	✓
7		ESTACIONES METEOROLÓGICAS	Verificar la creación de una variable meteorológica.	✓
8			Verificar el funcionamiento de la edición de una estación meteorológica.	✓
9			Verificar la eliminación de una estación meteorológica.	✓
10			Verificar la ubicación geográfica de una estación.	✓
11			Verificar presentación geográfica de un conjunto de estaciones dentro de un mapa de google.	✓
12			Verificar la visualización detallada por variables de una estación meteorológica.	✓
13			Verificar la presentación de registros históricos de una variable correspondiente a los 2 últimos días.	✓
14		GESTIÓN DESCARGAS	Verificar la descarga histórica de registros almacenados para una estación y variable, en formato Csv/Json.	✓
15		GESTIÓN ALERTAS	Verificar la creación de un evento para una estación meteorológica.	✓
16			Verificar la edición de un evento para una estación meteorológica.	✓
17			Verificar la eliminación de un evento.	✓
18			Verificar la notificación del suceso de un EVENTO a través del canal SMS	✓
19			Verificar la notificación del suceso de un EVENTO a través del canal EMAIL.	✓
20	VISUALIZACIÓN DE DATOS	ESTACIONES METEOROLÓGICAS	Verificar que en el portal se presente un mapa de Google con las estaciones meteorológicas.	✓
21		SOS	Verificar que en el portal se presente un gráfico comparativo de la temperatura de diferentes estaciones.	✓

Fuente: Elaborada por los autores.

4.2.4.3 Pruebas de interfaz gráfica de usuario.

Este tipo de pruebas tiene como objetivo garantizar la funcionalidad adecuada de la interfaz gráfica de usuario (GUI) de una aplicación determinada, así como asegurar que la GUI se ajusta a sus especificaciones escritas. Es decir, este tipo de pruebas permiten interactuar con la interfaz gráfica de usuario de la aplicación para validar las propiedades y estado de sus elementos. Además de la funcionalidad, las pruebas de GUI evalúan elementos de diseño tales como colores, fuentes, tamaños de fuente, etiquetas, cuadros de texto, formato de texto, subtítulos, botones, listas, iconos, enlaces y contenido. Los procesos de prueba de interfaz gráfica de usuario pueden ser manuales o automáticos y, a menudo, son realizados por empresas de terceros, en lugar de desarrolladores o usuarios finales (Tech Target, 2016).

Objetivos:

- Verificar la compatibilidad con navegadores y dispositivos.
- Validar que tanto el tiempo de carga en un navegador, así como el tamaño de los objetos del Subsistema de Visualización de Datos y plataforma de administración estén alineados a los estándares de calidad recomendados.

Estrategia: A través del uso de herramientas de software se validará las características de compatibilidad con navegadores y dispositivos, tiempos de carga, y navegabilidad móvil; en el Subsistema de Gestión de Estaciones Meteorológicas y Variables medioambientales, y el Subsistema de Visualización de Datos.

Escenarios: Se diseñaron seis escenarios para validar la GUI, a continuación, su detalle en la tabla 20.

Tabla 20: Escenarios de pruebas gráfica de interfaz de usuario

			Descripción	Herramienta	Pasos	Resultado Esperado
1	**VISUALIZACIÓN DE DATOS**	**COMPATIBILIDAD CON NAVEGADORES Y DISPOSITIVOS**	Verificar que el portal sea compatible con los 4 navegadores más usados CHROME 50, MICROSOFT EDGE 13.0, FIREFOX 44 y SAFARI 9.1 (http://www.w3schools.com/browsers/). Además, validar que la aplicación se presente correctamente en los dispositivos Android5.0 y iPhone6 en adelante.	– MICROSOFT EDGE screenshots (https://developer.microsoft.com/en-us/microsoft-edge/tools/screenshots/)	1. Ingresar en la herramienta en línea MICROSOFT EDGE SCREENSHOTS. 2. Ingresar la URL del portal web. 3. Presionar la tecla <ENTER>	El portal se presenta adecuadamente en los navegadores establecidos.
2		**TIEMPO DE CARGA Y PESO**	Verificar que el tiempo de carga del portal sea inferior a los 5 segundos que se recomienda por norma general.	– WEBPAGETEST (http://www.webpagetest.org/)	1. Ingresar en la herramienta en línea WEBPAGETEST (http://www.webpagetest.org/)	El portal responde en menos de 5 segundos.
3		**NAVEGABILIDAD MÓVIL**	Verificar que el contenido del sitio esté optimizado para dispositivos móviles.	– Mobile-Friendly Test de Google (https://search.google.com/search-console/mobile-friendly)	1. Ingresa en la herramienta Mobile-Friendly Test de Google. 2. Ingresar la URL del portal web. 3. Presionar el botón <Ejecutar prueba>	El portal se presenta correctamente en entornos móviles.
4	GESTIÓN DE ESTACIONES METEREOLÓGICAS Y VARIABLES	**COMPATIBILIDAD CON NAVEGADORES Y DISPOSITIVOS**	Verificar que la plataforma sea compatible con los 4 navegadores más usados CHROME 50, MICROSOFT EDGE 13.0, FIREFOX 44 y SAFARI 9.1 (http://www.w3schools.com/browsers/).	– MICROSOFT EDGE screenshots (https://developer.microsoft.com/en-us/microsoft-edge/tools/screenshots/)	1. Ingresar en la herramienta en línea MICROSOFT EDGE SCREENSHOTS. 2. Ingresar la URL de la web. 3. Presionar la tecla <ENTER>	La plataforma web se presenta adecuadamente en los navegadores establecidos.

5	GESTIÓN DE ESTACIONES METEREOLÓGICAS Y VARIABLES MEDIOAMBIENTALES	**TIEMPO DE CARGA Y PESO**	Verificar que el tiempo de carga de la plataforma web sea inferior a los 5 segundos que se recomienda por norma general.	– WEBPAGETEST (http://www.webpagetest.org/)	1. Ingresar en la herramienta en línea WEBPAGETEST (http://www.webpagetest.org/)	La plataforma web responde en menos de 5 segundos.
6		**NAVEGABILIDAD MÓVIL**	Verificar que el contenido del sitio web esta optimizado para dispositivos móviles.	– Mobile-Friendly Test de Google (https://search.google.com/search-console/mobile-friendly)	1. Ingresa en la herramienta Mobile-Friendly Test de Google. 2. Ingresar la URL de la plataforma web. 3. Presionar el botón <Ejecutar prueba>	La plataforma web se presenta correctamente en entornos móviles.

Fuente: Elaborada por los autores.

Resultados: A continuación, la tabla 21 muestra los resultados correspondientes a las pruebas de interfaz de usuario, en las cuales fueron validados seis escenarios de prueba.

Tabla 21: Resultados de pruebas de interfaz de gráfica de usuario

Escenario		Funcionalidad	Descripción	Resultado Validación
1	VISUALIZACIÓN DE DATOS	COMPATIBILIDAD CON NAVEGADORES Y DISPOSITIVOS	Verificar que el portal sea compatible con los 4 navegadores más usados CHROME 50, MICROSOFT EDGE 13.0, FIREFOX 44 y SAFARI 9.1 (http://www.w3schools.com/browsers/). Además, validar que la aplicación se presente correctamente en los dispositivos Android 5.0 y iPhone 6 en adelante.	✓
2		TIEMPO DE CARGA Y PESO	Verificar que el tiempo de carga del portal sea inferior a los 5 segundos que se recomienda por norma general.	✓
3		NAVEGABILIDAD MÓVIL	Verificar que el contenido del sitio esta optimizado para dispositivos móviles.	✓
4	GESTIÓN DE ESTACIONES METEREOLÓGICAS Y VARIABLES MEDIOAMBIENTALES	COMPATIBILIDAD CON NAVEGADORES Y DISPOSITIVOS	Verificar que la plataforma web sea compatible con los 4 navegadores más usados CHROME 50, MICROSOFT EDGE 13.0, FIREFOX 44 y SAFARI 9.1 (http://www.w3schools.com/browsers/).	✓
5		TIEMPO DE CARGA Y PESO	Verificar que el tiempo de carga de la plataforma web sea inferior a los 5 segundos que se recomienda por norma general.	✓
6		NAVEGABILIDAD MÓVIL	Verificar que el contenido de la plataforma web esta optimizado para dispositivos móviles.	✓

Fuente: Elaborada por los autores.

Consecuentemente, se puede evidenciar que las pruebas de interfaz de usuario han sido superadas con satisfacción. Desde la figura 33 a la 36 se muestra los resultados obtenidos a través de las herramientas de prueba disponibles.

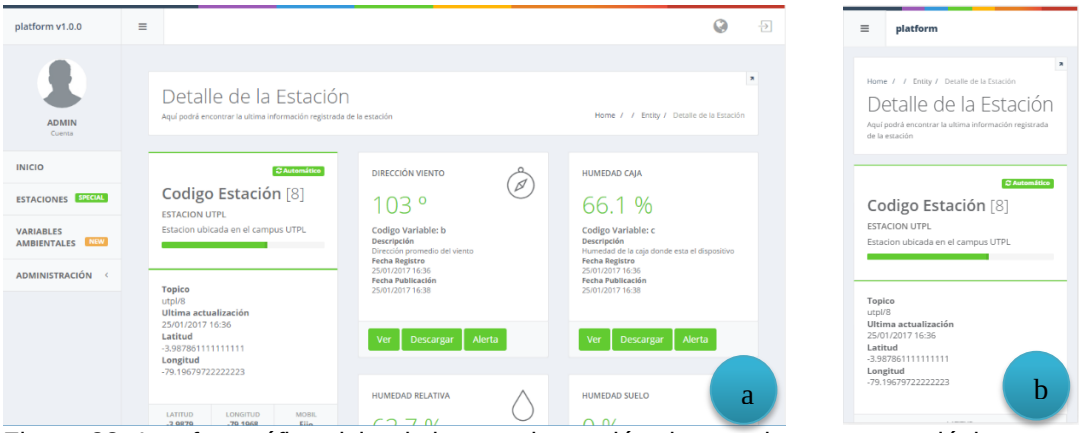


Figura 33: Interfaz gráfica del subsistema de gestión de estaciones meteorológicas y variables medioambientales en navegadores de escritorio (a) y en dispositivos móviles (b).
Fuente: Obtenido con la herramienta MICROSOFT EDGE SCREENSHOTS (2016)

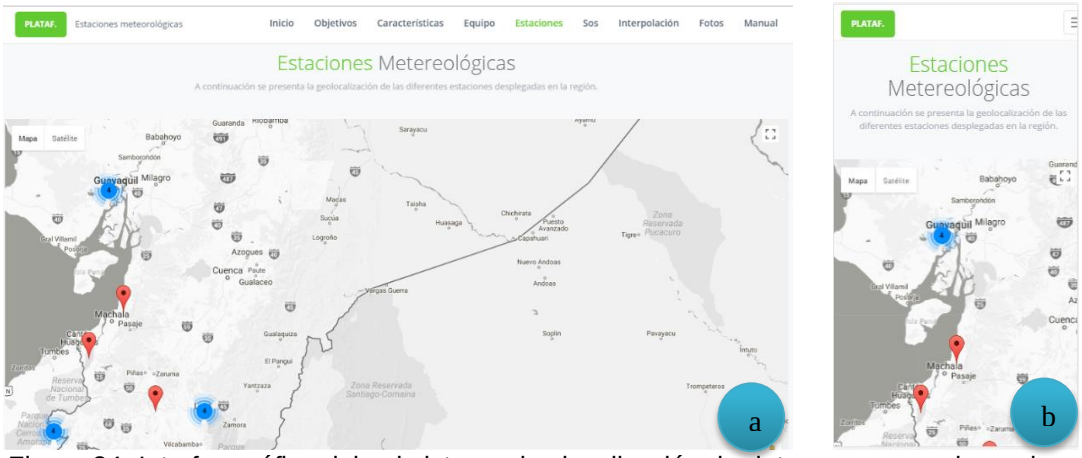


Figura 34: Interfaz gráfica del subsistema de visualización de datos en navegadores de escritorio (a) y en dispositivos móviles (b).
Fuente: Obtenido con la herramienta MICROSOFT EDGE SCREENSHOTS (2016)

Figura 35: Desglose de contenido de la GUI del subsistema de visualización de datos
Fuente: Obtenido con la herramienta Web Page Performance Test (2016)

Figura 36: Desglose de contenido de la GUI- subsistema de gestión de estaciones meteorológicas y variables medioambientales
Fuente: Obtenido con la herramienta Web Page Performance Test (2016)

4.2.4.4 Pruebas de carga.

La prueba de carga es el proceso de someter una aplicación a un nivel de trabajo que se aproxima a los límites de sus especificaciones. Las pruebas de carga pueden realizarse bajo condiciones de laboratorio controladas, con el propósito de identificar con precisión las capacidades de los sistemas (Search Software Quality, 2016).

Objetivo:

- Forzar el sistema al punto máximo para determinar los límites de funcionamiento normal de la solución desarrollada.

Estrategia: A través del uso de herramientas de software se someterá al sistema de manera que se pueda determinar la capacidad máxima correspondiente a las funcionalidades de consulta, descarga, y envío de información desde las estaciones meteorológicas; específicamente cuando existe un gran número de usuarios o estaciones conectados simultáneamente al subsistema de Gestión de Estaciones Meteorológicas y Variables medioambientales. Por consiguiente, se ejecutará la simulación con una cantidad considerable de usuarios conectados, la misma que será incrementada gradualmente con el fin de descubrir capacidades máximas, y determinar el escenario en el cual la plataforma colapsa. La tabla 22, muestra el detalle de las herramientas a utilizar para realizar las pruebas de carga.

Tabla 22: Herramientas de software utilizadas para la ejecución de pruebas de carga

Herramienta	Descripción
Mqtt.fx	Herramienta de software que permite la publicación/suscripción de tópicos a través del protocolo MQTT.
Gatling	Diseñado para ser utilizado como una herramienta de prueba de carga para analizar y medir el rendimiento de una variedad de servicios, con un enfoque en las aplicaciones web.

Robomongo	Es una herramienta de software que nos provee de una interfaz gráfica para poder administrar una base de datos NoSQL en nuestro caso MongoDB.
flood.io	Flood IO permite ejecutar pruebas de carga distribuidas en todo el mundo, utilizando scripts de Gatling como insumo de entrada.

Fuente: Elaborada por los autores.

Escenarios: Para este tipo de pruebas se diseñó tres escenarios correspondientes a consulta, descarga, y envío de información desde las estaciones meteorológicas. A continuación, el detalle de los escenarios:

- *Escenario 1:* Consulta de variables medioambientales

El usuario consulta las estaciones meteorológicas, selecciona una estación meteorológica de su interés, y visualiza las variables medioambientales de aquella estación. Luego, el usuario selecciona variable determinada dando clic en ella, con el fin de visualizar el detalle correspondiente al histórico de datos obtenidos a través de los sensores. En la figura 37, se muestra el diagrama que representa este escenario.

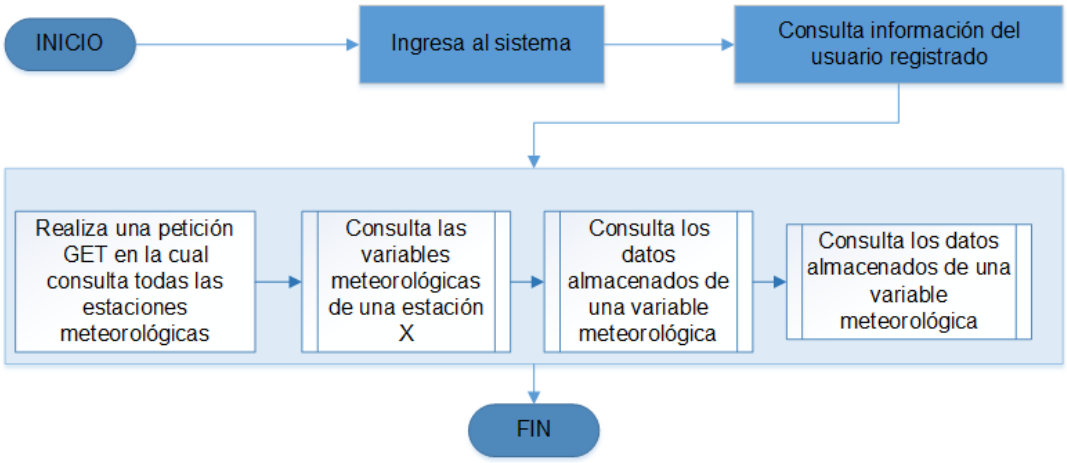


Figura 37: Pruebas de carga - diagrama de flujo para consulta de variables medioambientales
Fuente: Elaborada por los autores.

Una vez diseñado el escenario, con la ayuda de herramientas se ha procedido a realizar varias ejecuciones, acorde a la tabla 23, en la que se define los diferentes casos a ejecutar. Además, se visualiza los resultados de las métricas definidas tanto del lado del cliente como del lado del servidor.

Tabla 23: Casos de ejecución – escenario 1 – pruebas de carga: consulta de variables medioambientales

DATOS DE PARTIDA Configurados		CLIENTE			SERVIDOR			
		Tiempo de respuesta (ms)						
Tiempo (minutos)	**Usuarios**	**Mínimo**	**Media**	**Máximo**	**Usuarios concurrentes**	**Errores (%)**	**CPU usado (%)**	RAM usado (Gb)
1	100	294	303	335	82	0	11	5.9
1	200	293	310	335	156	0	14	5.9
1	300	294	363	399	237	0	20	5.9
1	400	339	1274	3266	339	0	31	5.9
1	500	346	3539	7743	421	0	34	5.9
1	600	331	3731	7745	491	13.05	36	5.9
1	700	145	3422	7800	559	23.15	36	5.9

* La RAM no varía en el servidor ya que ha sido configurada por defecto en 3GB de memoria para Tomcat Apache
Fuente: Elaborada por los autores.

- *Escenario 2:* Descarga de información de variables medioambientales

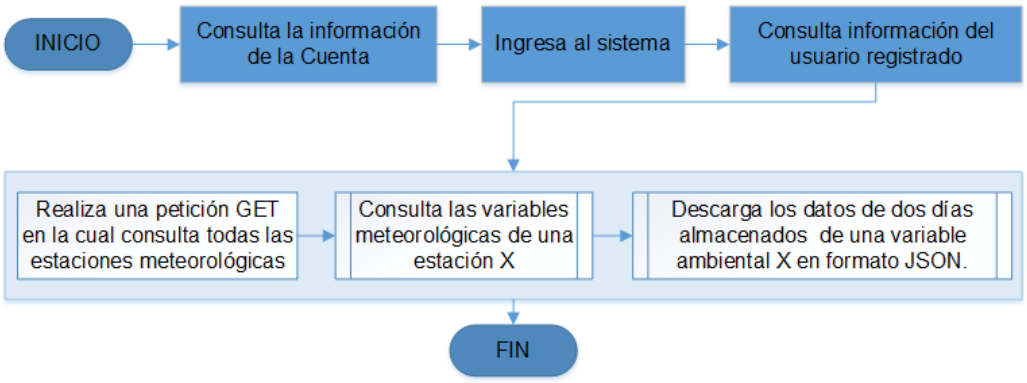


Figura 38: Pruebas de carga - diagrama de flujo para descarga de información de variables medioambientales
Fuente: Elaborada por los autores.

Un usuario requiere exportar los registros almacenados a un formato Json. Para lo cual se autentifica en el Subsistema de Gestión de Estaciones Meteorológicas y Variables medioambientales, selecciona una estación, selecciona una variable medioambiental sobre la cual descargar la información, y finalmente establece un rango de fechas del cual desea la información. Con todo ese proceso definido el sistema genera un archivo Json. El diagrama que representa este escenario es ilustrado en la figura 38.

Luego de haber sido el escenario definido, se procedió a realizar las pruebas de carga de acuerdo a los casos de prueba detallados en la tabla 24. La tabla contiene los resultados obtenidos al ejecutar los scripts de prueba tanto del lado del cliente como del servidor.

Tabla 24: Casos de ejecución – escenario 2 – pruebas de carga: descarga de información de variables medioambientales

DATOS DE PARTIDA Configurados		CLIENTE			SERVIDOR			
		TIEMPO DE RESPUESTA (milisegundos)						
Tiempo (minuto)	Usuarios	Mínimo	Media	Máximo	Usuarios concurrentes	Errores (%)	CPU usado (%)	RAM usado (GB)
1	100	151	219	235	69	0	3	5.7
1	200	222	236	240	154	0	11	5.8
1	300	223	235	242	226	0	17	5.8
1	400	225	234	243	291	0	22	5.8
1	500	233	974	1976	363	0	28	5.8
1	600	232	1858	3461	447	5	35	5.8
1	700	146	1593	3746	550	12.54	42	5.8

* La RAM no varía en el servidor ya que ha sido configurada por defecto en 3GB de memoria para Tomcat Apache
Fuente: Elaborada por los autores.

- *Escenario 3*: envío de información desde las estaciones meteorológicas.

En este escenario se simula que las estaciones meteorológicas enviando N mensajes por segundo durante una hora, para así determinar el correcto funcionamiento del sistema de almacenamiento, así como su capacidad máxima.

Una vez diseñado el escenario, con la ayuda de herramientas se ha procedido a realizar varias ejecuciones, acorde a la tabla 25, en la que se define los diferentes casos a ejecutar.

Tabla 25: Casos de ejecución – escenario 3 – pruebas de carga: envío de información desde las estaciones meteorológicas

DATOS DE CONFIGURACIÓN		SERVIDOR		
DURACIÓN DE LA PRUEBA (hora)	MENSAJES POR SEGUNDO	ERRORES (%)	CPU USADO (%)	RAM USADO (GB)
1	10	0	5%	5.7
1	20	0	10%	5.8
1	30	0	20%	5.9
1	40	25%	30%	6

Fuente: Elaborada por los autores.

Resultados: A continuación, la tabla 26 muestra los resultados correspondientes a las pruebas de carga, en las cuales fueron validados tres escenarios.

Tabla 26: Resultados de pruebas de carga

Escenario	Funcionalidad	Caso Óptimo	Caso Crítico
1	Visualización	A través de los diferentes casos de ejecución, se pudo determinar que la plataforma cumple con los tiempos de respuesta adecuados hasta los 500 usuarios conectados dentro de un 1 minuto.	La plataforma presenta errores a partir de los 491 usuarios conectados concurrentemente, ya que en dicho caso presenta una tasa de error de 13.05%.
2	Descarga	La plataforma cuando los usuarios descargan información soporta simultáneamente 291 usuarios sin problemas.	A partir de los 363 usuarios concurrentes se empieza a evidenciar errores a través de las diferentes peticiones.
3	Envío de mensajes	El sistema de almacenamiento puede procesar sin problemas hasta 30 tramas por segundo.	A partir de las 40 tramas por segundo se pone en evidencia un porcentaje de error del 25%.

Fuente: Elaborada por los autores.

4.2.5 Análisis global de resultados.

Luego de haber diseñado y ejecutado el plan de pruebas se logró realizar las validaciones correspondientes, se puede concluir que el software desarrollado ha superado exitosamente las pruebas correspondientes a la integridad de datos, funcionalidad, interfaz de usuario, y carga. La tabla 27, resume los objetivos alcanzados a través de la aplicación del plan de pruebas.

Tabla 27: Resumen de pruebas

Pruebas	Objetivo	Escenarios Validados	Resultado Validación
Pruebas de Integridad de Datos	Asegurar que las tramas de datos provenientes de las estaciones meteorológicas se almacenen de manera íntegra en las colecciones de MongoDB, de acuerdo al estándar SOS dentro de 52North (PostgreSQL).	4	✓
Pruebas de Funcionalidad	Validar que los requerimientos funcionales se cumplan a cabalidad.	21	✓
Pruebas de Interfaz de Usuario	Verificar la compatibilidad con navegadores y dispositivos. Validar que tanto el tiempo de carga en un navegador, así como el tamaño de los objetos del subsistema de visualización de datos y subsistema de gestión de estaciones meteorológicas y variables medioambientales estén alineados a los estándares de calidad recomendados.	6	✓
Pruebas de Carga	Forzar el sistema al punto máximo para determinar los límites de funcionamiento normal de la solución	3	✓

Fuente: Elaborada por los autores.

CONCLUSIONES

- Se ha desarrollado una plataforma para la recolección, almacenamiento y visualización de variables medioambientales obtenidas de una red de sensores. Asimismo, se ha subdividido la solución en tres subsistemas: subsistema de almacenamiento, subsistema de gestión de estaciones meteorológicas y variables medioambientales, y el subsistema de visualización de datos para el público en general. Mediante la utilización de herramientas, protocolos y estándares de código abierto.
- Al elaborar el estado de arte, se realizó un compendio teórico de trabajos similares relacionados con el tema de sistemas de almacenamiento y visualización de datos medioambientales; lo que nos permitió concluir que en la actualidad existen varias plataformas orientados a la IoT como: Ubidots, Amazon Web Services, y IBM Watson. Sin embargo, todas estas plataformas ofrecen características limitadas en aspectos relacionados con capacidad de almacenamiento, número de terminales conectados concurrentemente, número de mensajes enviados; todo esto con respecto a la versión gratuita. Asimismo, no todas las plataformas disponen de herramientas para la visualización e interpretación de los datos.
- Con la utilización de tecnologías de código abierto para los componentes de back-end y front-end, los cuales interactúan activamente para brindar un correcto funcionamiento al usuario. Para el back-end, la parte del lado del servidor, se ha seleccionado el lenguaje de programación Java por ser confiable y robusto al momento de desarrollar soluciones de software. Mientras que para el front-end, la parte del lado del cliente, se ha seleccionado el lenguaje de programación Java Script a través del framework AngularJs debido a que está basado en el patrón “modelo-vista-vista-modelo” (MVVM); que proporciona controladores, servicios y directivas para organizar el proyecto.
- Con el fin integrar a la plataforma interoperabilidad entre otros sistemas, se integra el estándar SOS en el almacenamiento de información proveniente de las estaciones meteorológicas. Al ser SOS un estándar aprobado por el OGC, provee una interfaz estandarizada para la distribución de datos; lo cual permite a otros sistemas geoespaciales de características heterogéneas el acceso a la información recolectada.
- Se aplica de MQTT como protocolo de comunicación entre los sensores de recolección de datos y el sistema de almacenamiento, debido a sus características de: bi-direccionalidad, sencillez, y calidad de servicio. MQTT permite establecer una

comunicación bidireccional ya que sigue el modelo de publicación/subscripción. Por otro lado, el formato de MQTT es liviano, lo que es ideal para redes con un bajo ancho de banda, alta latencia, o en conexiones frágiles, altos costos de servicio; y para dispositivos con limitaciones en la capacidad de procesamiento, memoria y consumo de energía. Asimismo, MQTT es fiable porque ofrece tres niveles de calidad de servicio, particularmente, en la presente solución se ha implementado el primer nivel “envía y olvida”, debido a que el envío de mensajes correspondiente a una medición de variable medioambiental se lo realiza cada 10 minutos, y la pérdida de algún mensaje no constituiría un problema significativo para el sistema.

- Para asegurar la calidad de software en la solución desarrollada, se ha ejecutado un plan estructurado por cuatro tipos de pruebas: de integridad de datos, de funcionalidad, de interfaz gráfica de usuario y de carga. A través de las pruebas de integridad de datos se pudo asegurar que las tramas de datos provenientes de las estaciones meteorológicas se almacenan de manera íntegra en las colecciones de MongoDB y 52North (Postgresql). En cambio, con las pruebas de funcionalidad se validó que los requerimientos funcionales se cumplan a cabalidad. Por otro lado, las pruebas de interfaz gráfica de usuario nos permitieron verificar la compatibilidad con navegadores y dispositivos; así como el tamaño de los objetos del sistema estén alineados a los estándares de calidad recomendados. Finalmente, las pruebas de carga permitieron forzar el sistema al punto máximo para determinar los límites de funcionamiento normal de la solución.

RECOMENDACIONES

- Debido a que SOS es un estándar con un nivel de complejidad alta de implementación en la IoT; se recomienda tener como referencia para futuros trabajos al OGC SensorThings API (STA).
- Debido a la evolución de los lenguajes de programación y frameworks de desarrollo; se recomienda la utilización para trabajos futuros de los frameworks es ReactJS y Meteor que están basados en la construcción de apps en tiempo real.
- Utilizar el subsistema de Gestión de Estaciones Meteorológicas y Variables medioambientales dentro de diferentes ámbitos que no sean necesariamente de tipo variables medioambientales; como puede ser nivel de caudal en ríos, consumo de energía eléctrica.
- Para futuras actualizaciones al subsistema de Gestión de Estaciones Meteorológicas y Variables medioambientales se recomienda seguir utilizando el patrón MVVM (Model View View-Model) de Angular js.

BIBLIOGRAFÍA

52°North Initiative for Geospatial Open Source Software GmbH. (Agosto de 2016). *Sensor Web, Geoprocessing, Security & GeoRM, Geostatistics, Semantics, 3D, Ilwis, Earth Observation*. Obtenido de Seismic Observations in SOS, Midterm Report: http://blog.52north.org/2013/07/31/seismic-observations-in-sos-midterm-report/

Amazon Web Services. (Diciembre de 2016). *Precios de AWS IoT*. Obtenido de https://aws.amazon.com/es/iot-platform/pricing/

Bosch Software Innovations and Mongo DB. (Agosto de 2015). *IoT and Big Data.* Obtenido de A Joint Whitepaper by Bosch Software Innovations and MongoDB: https://s3.amazonaws.com/info-mongodb-com/MongoDB_BoschSI_IoT_BigData.pdf?utm_campaign=Int_OC_Internet%20of%20Things%20with%20Bosch%20White%20Paper_WW%20-%20Autoresponder&utm_medium=email&utm_source=Eloqua

Bröring, A., Stasch, C., & Echterhoff, J. (16 de Abril de 2012). *Open Geospatial Consortium.* Obtenido de https://portal.opengeospatial.org/files/?artifact_id=47599

Camacho, R. R. (Septiembre de 2014). Infraestructura de Eventos para la Internet de las Cosas. Alcazar de San Juan, España.

Castells, M. (2011). *INTERNET Y LA SOCIEDAD RED.* (U. O. Catalunya, Ed.) Obtenido de Conferencia de Presentación del Programa de Doctorado sobre la Sociedad de la Información y el Conocimiento: http://s3.amazonaws.com/academia.edu.documents/34314728/INTERNET_Y_LA_SOCIEDAD_RED.pdf?AWSAccessKeyId=AKIAJ56TQJRTWSMTNPEA&Expires=1470785808&Signature=b5%2FNd8LNMvaFSYO6mxR3W9mvNvg%3D&response-content-disposition=inline%3B%20filename%3DINTERNET_Y_LA_SOCI

Chen S, Xu H, Liu D, Hu B , & Wang H. (2014). A Vision of IoT: Applications, Challenges and Opportunities With China Perspective. *IEEE Internet of Things Journal 1*, 349-359.

CSIRT-CV. (2007). *SEGURIDAD EN INTERNET DE LAS COSAS.* Obtenido de http://www.csirtcv.gva.es/sites/all/files/downloads/%5BCSIRT-CV%5D%20Informe-Internet_de_las_Cosas.pdf

Deitel, P., & Deitel, H. (2012). *Como Programar Java.* FINANCIAL TIMES/PRENTICE HALL.

Evans, D. (2011). *Internet of Things La próxima evolución de Internet lo está cambiando todo.*

Gartner. (2014). *Newsroom. Gartner's 2014 Hype Cycle for Emerging Technologies.*

Ghahrai, A. (22 de Enero de 2009). *Testing Excellence, learn software testing*. Obtenido de What is Data and Database Integrity Testing?: http://www.testingexcellence.com/what-is-data-and-database-integrity-testing/

GidHub. (2016). *Sensor Widgets*. Obtenido de http://sensors.fonts.cat/

Google. (2016). *AngularJS*. Obtenido de https://angularjs.org/

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (24 de Febrero de 2013). *Future Generation Computer Systems.* Obtenido de Internet of Things (IoT): A vision, architectural elements, and future directions: http://www.sciencedirect.com/science/article/pii/S0167739X13000241

Heidloff, N. (14 de Junio de 2016). *Analyzing Data with IBM Watson Internet of Things*. Obtenido de http://heidloff.net/article/analyzing-data-ibm-watson-internet-of-things

IBM. (Diciembre de 2016). *IBM Knowledge Center*. Obtenido de Envío de un mensaje a un cliente

MQTT: https://www.ibm.com/support/knowledgecenter/es/SSFKSJ_7.5.0/com.ibm.mq.pro.doc/q002890_.htm

IBM. (Diciembre de 2016). *IBM Watson Internet of Things*. Obtenido de The Emerging Technology 'Web' for Our Live: http://www.ibm.com/internet-of-things/

IBM. (s.f.). *http://www-01.ibm.com*. Obtenido de http://www-01.ibm.com/support/knowledgecenter/SS9D84_1.0.0/com.ibm.mm.tc.doc/tc60340_.htm?lang=es

IOTpreneur. (10 de 10 de 2016). *Internet de las Cosas y el Ciclo de sobre expectación de Gartner (Parte II)*. Obtenido de http://www.iotpreneur.com/internet-de-las-cosas-y-el-ciclo-de-sobre-expectacion-de-gartner-ii/

ISTQBExamCertification. (Diciembre de 2016). *ISTQB Exam Certification*. Obtenido de What is Functional testing (Testing of functions) in software?: http://istqbexamcertification.com/what-is-functional-testing-testing-of-functions-in-software/

ITU. (2005). *ITU Internet Reports.* Geneva.

Kontio, M. (2005). *Architectural manifesto: Designing software architectures, Part 5.*

Kruchten, P. (1995). *The "4+1" view model of software architecture.*

Lampkin, V., Leong, Weng Tat, Olivera, Leonardo, Rawat, Sweta, Subrahmanyam, Nagesh, & Xiang, Rong. (2012). *Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry.* IBM Redbooks.

Layton , M. (2015). *10 Key Benefits of Scrum.* Wiley Brand.

Lu Zeyong, G. G., & Jun, J. (2011). *Internet of things security analysis. In Internet Technology and Applications.* Obtenido de http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6006307&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6006307

Matamala, M. (2008). *Lenguajes de Programación.* Obtenido de http://www.mauriciomatamala.net/ED/lenguajes/lenguajes_de_programacion.pdf

meteo.apb.es. (2016). Obtenido de http://sensors.fonts.cat/examples/meteo.apb.es/#

Meyer, G. G. (2009). *Intelligent Products: A survey.*

Meyer, G. G. (2009). *Intelligent Products: A survey.* Obtenido de http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.159.5049

MICROSOFT EDGE. (Diciembre de 2016). *Browser screenshots*. Obtenido de Microsoft Developer Technologies: https://developer.microsoft.com/en-us/microsoft-edge/tools/screenshots/?url=http%3A%2F%2F190.15.141.114%2F%23%2F

Milan Antonovic , M. (2015). *Introduction to the SOS standard*. Obtenido de http://istsos.org/en/trunk/doc/intro.html

MQTT.ORG. (2016). *MQTT (MQ Telemetry Transport)*. Obtenido de http://mqtt.org/

Nicholas, S. (31 de mayo de 2012). *Power Profiling: HTTPS Long Polling vs. MQTT with SSL, on Android*. Obtenido de http://stephendnicholas.com/posts/power-profiling-mqtt-vs-https

OASIS. (12 de Diciembre de 2013). *docs.oasis-open.* Obtenido de http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/csprd01/mqtt-v3.1.1-csprd01.pdf

Open Geospatial Consortium. (2011). *OGC Network*. Obtenido de How to model your observation data in SOS 2.0?: http://www.ogcnetwork.net/sos_2_0/tutorial/om

OSGeo-Live. (2016). *52°North SOS.* Obtenido de https://live.osgeo.org/en/overview/52nSOS_overview.html

Paper, R. S. (s.f.). *www.ibm.com.*

RabbitMQ. (15 de 01 de 2016). *RabbitMQ official page*. Obtenido de http://www.rabbitmq.com/

(s.f.). *RFID and the Inclusive Model for the Internet of Things.*

Rodríguez González, D. (Diciembre de 2013). *Arquitectura y Gestión de la IoT*. Obtenido de http://revistatelematica.cujae.edu.cu/index.php/tele/article/viewFile/119/115

Rouse, M. (Febrero de 2007). *Software Development Fundamentals*. Obtenido de Development environment: http://searchsoftwarequality.techtarget.com/definition/development-environment

Sampalo de la Torre, M., Leyva Cortés, E., Garzón Villar, M., & Prieto Tinoco, J. (2003). *Informática*. España: Mad, S.L.

Saravia, R. (Enero de 2014). *FUNDAMENTOS DE BASES DE DATOS UNIDAD I*. Obtenido de http://www.academia.edu/11156604/FUNDAMENTOS_DE_BASES_DE_DATOS_UNIDAD_I

Search Software Quality. (December de 2016). *Load testing* . Obtenido de Software Testing Methodologies: http://searchsoftwarequality.techtarget.com/definition/load-testing

Solano, L. F. (2013). *Análisis de los Sistemas de Gestión de Bases de Datos actuales como soporte para las tecnologías de Internet de las Cosas.* (U. P. Valencia, Ed.) Obtenido de https://riunet.upv.es/bitstream/handle/10251/43325/Thesis%20Loanny%20Solano.pdf?sequence=1

Sundmaeker, H., & Guillemin, P. (2010). *Vision and Challenges for Realising the Internet of Things.*

Svennerberg , G. (13 de Enero de 2009). *Handling Large Amounts of Markers in Google Maps*. Obtenido de MarkerClusterer - The new kid in town: http://www.svennerberg.com/2009/01/handling-large-amounts-of-markers-in-google-maps/

Tech Target. (2016). *GUI testing (graphical user interface testing)* . Obtenido de http://whatis.techtarget.com/definition/GUI-testing-graphical-user-interface-testing

TIOBE software BV. (2016). *TIOBE's programming language ranquing of 2016*. Obtenido de http://www.tiobe.com/tiobe-index//

Tutorials Point. (Diciembre de 2016). *Learn software testing terms*. Obtenido de What is data integrity testing: https://www.tutorialspoint.com/software_testing_dictionary/data_and_database_integrity_testing.htm

Ubidots, C. (2016). *Flexible pricing to take your project from prototype to production*. Obtenido de https://ubidots.com/pricing

Web Page Performance Test. (Diciembre de 2016). *Web Page Performance Test*. Obtenido de http://www.webpagetest.org/result/170125_QN_3378/

Web Page Performance Test. (Diciembre de 2016). *Web Page Performance Test* . Obtenido de http://www.webpagetest.org/result/170125_7A_3VV/

Xamarin Inc. (Noviembre de 2015). *Introduction to Continuous Integration with Xamarin*. Obtenido de Integrating Xamarin Projects into a Continuous Build Workflow: https://developer.xamarin.com/guides/cross-platform/ci/intro_to_ci/

Zhang, L. (12 de Agosto de 2011). *https://www.facebook.com.* Obtenido de https://www.facebook.com/notes/facebook-engineering/building-facebook-messenger/10150259350998920

ANEXOS

ANEXO # 1: Formato de mensaje MQTT

El protocolo MQTT trabaja intercambiando mensajes denominados MQTT Control Packets. Un Control Packet se encuentra conformado por tres partes en el siguiente orden: primero una *cabecera fija (Fixed Header)*, presente en todos los MQTT paquetes; segundo una *cabecera variable (Variable Header)*, presente en algunos paquetes y su contenido depende del tipo de paquete; y tercero *Payload* incluido en algunos paquetes (OASIS, 2013).

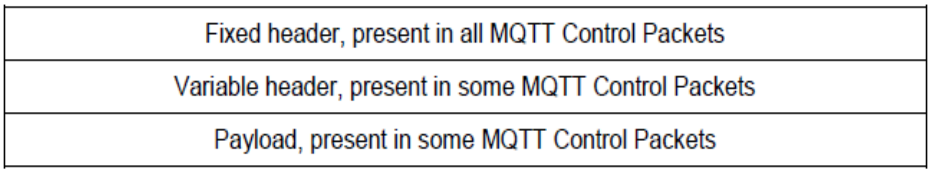


Figura 39: Formato de mensajes MQTT
Fuente: Tomada de MQTT Versión 3.1.1**.** (OASIS, 2013)

1. Cabecera fija (*fixed header*)

La cabecera fija de cada mensaje MQTT contiene información correspondiente a tipo de mensaje, DUP, el nivel de calidad (QoS), *retain*, y *remaining length*. El tamaño de la cabecera fija puede variar de 2 a 5 bytes. El primer byte contiene 8 bits, de los cuales los 4 bits corresponden al tipo de mensaje (message type), el siguiente bit corresponde al *DUP Flag*, los dos siguientes bits al QoS, y en el último bit a la bandera *retain*. Mientras que el segundo byte contiene información correspondiente a *remaining lenght*, como se ilustra en la figura 40.

Figura 40: Estructura de la cabecera fija
Fuente*:* Tomada de MQTT Versión 3.1.1**.** (OASIS, 2013)

Tipo de mensaje (*message type*)

El *message type* es un campo de 4 bit, que contiene información acerca del tipo de mensaje. Los valores posibles están listados en la tabla 28, a continuación:

Tabla 28: Tipos de mensajes MQTT

Mensaje	Dirección	Numeración	Descripción
CONNECT	Cliente al bróker	1	El cliente solicita conectarse al servidor.
CONNACK	Bróker al cliente	2	Confirmación de conexión.
PUBLISH	Cliente al bróker o Bróker al cliente	3	Publica un mensaje.

PUBACK	Cliente al bróker o Bróker al cliente	4	Confirmación de publicación de mensaje.
PUBREC	Cliente al bróker o Bróker al cliente	5	Publicación recibe (QoS = 2)
PUBREL	Cliente al bróker o Bróker al cliente	6	Publicación libera (QoS = 2)
PUBCOMP	Cliente al bróker o Bróker al cliente	7	Publicación se completa (QoS = 2)
SUBSCRIBE	Cliente al bróker	8	El cliente solicita subscribirse
SUBACK	Bróker al cliente	9	Confirmación de suscripción.
UNSUBSCRIBE	Cliente al bróker	10	El cliente solicita cancelar suscripción
UNSUBACK	Bróker al cliente	11	Confirmación de cancelar suscripción
PINGREQ	Cliente al bróker	12	PING request
PINGRESP	Bróker al cliente	13	PING response
DISCONNECT	Cliente al bróker	14	Solicitud de desconexión del cliente
RESEVED	Prohibida	15	Reservado para uso futuro.

Fuente*:* Tomada de (OASIS, 2013)

Duplicate delivery of control packet (DUP Flag)

El indicador DUP se establece cuando el cliente o el servidor intenta re-enviar un mensaje PUBLISH, PUBREL, SUBSCRIBE o UNSUBSCRIBE, este indicador se utiliza sólo en los mensajes que tienen QoS > 0.

Calidad de servicio (*Quality of Service QoS*)

Este campo es denominado en español “nivel de calidad de servicio”, que se utiliza para ajustar el nivel de garantía de entrega de mensajes al PUBLICAR. A continuación, en la tabla 29 se detalla cada uno de los cuatro valores posibles de QoS.

Tabla 29: Tipos de calidad de servicio

Valor QoS	Bit 2	Bit 1	Descripción
0	0	0	Como máximo una vez se entrega (envía y olvida)
1	0	1	Como mínimo una vez (reconoce la entrega)
2	1	0	Exactamente una vez (asegura la entrega)
3	1	1	Reservada

Fuente: Tomada de (OASIS, 2013)

Una de las ventajas que posee MQTT como protocolo de la capa de aplicación es que facilita la distribución de sus mensajes de tipo PUBLISH de una manera garantizada. Es decir, el QoS nos permite indicar si se requiere o no recibir garantías en la entrega de los mensajes que se han enviado. Actualmente, se usan tres niveles de calidad de servicio, los mismos que serán descritos

a continuación.

QoS 0 – Como máximo una entrega (envía y olvida)

Cuando el QoS es igual a cero, el mensaje se entrega como máximo una vez o no se entrega. El mensaje es entregado de acuerdo a las capacidades de la red. Con este nivel no se efectúa el acuse de recibo de una entrega por parte del receptor, es decir ninguna respuesta es enviada al emisor. Asimismo, en este nivel de seguridad el mensaje no se almacena, por lo que el mensaje puede perderse si el cliente se desconecta o si falla el servidor (ver figura 41).

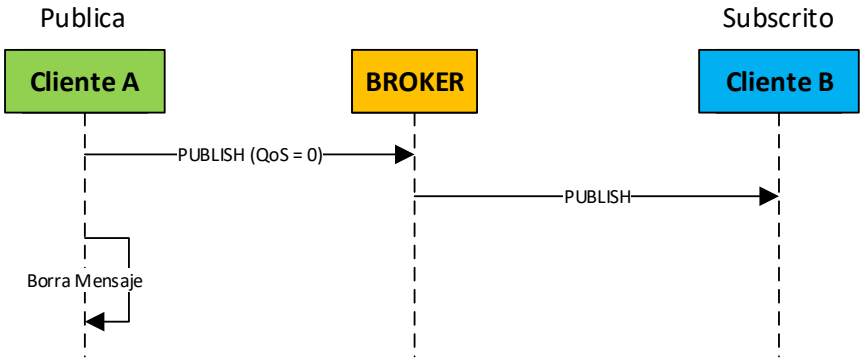


Figura 41: Diagrama de flujo QoS=0
Fuente: Elaborada por los autores

QoS 1 - Como mínimo una entrega (entrega con acuse de recibo).

En este nivel existe garantía en la distribución de mensajes. Cuando el QoS es igual a 1 el mensaje siempre se entrega, como mínimo, una vez. Si el emisor no recibe un acuse de recibo, el mensaje se envía de nuevo con el distintivo DUP establecido hasta que se reciba un acuse de recibo. Como consecuencia, se puede enviar al receptor el mismo elemento varias veces, y que se procese varias veces. El mensaje debe almacenarse localmente en el emisor y el receptor hasta que se procese. El mensaje se suprime del receptor después de que se haya procesado. Si el receptor es un intermediario, el mensaje se publica a sus suscriptores. Si el receptor es un cliente, el mensaje se entrega a la aplicación de suscriptor. En lo referente a la eliminación del mensaje el emisor y receptor lo hacen de la siguiente manera: (a) el receptor borra el mensaje una vez que se envíe un acuse de recibo al emisor, y (b) el emisor borra el mensaje después de que se haya recibido el acuse de recibo por parte del receptor (ver figura 42).

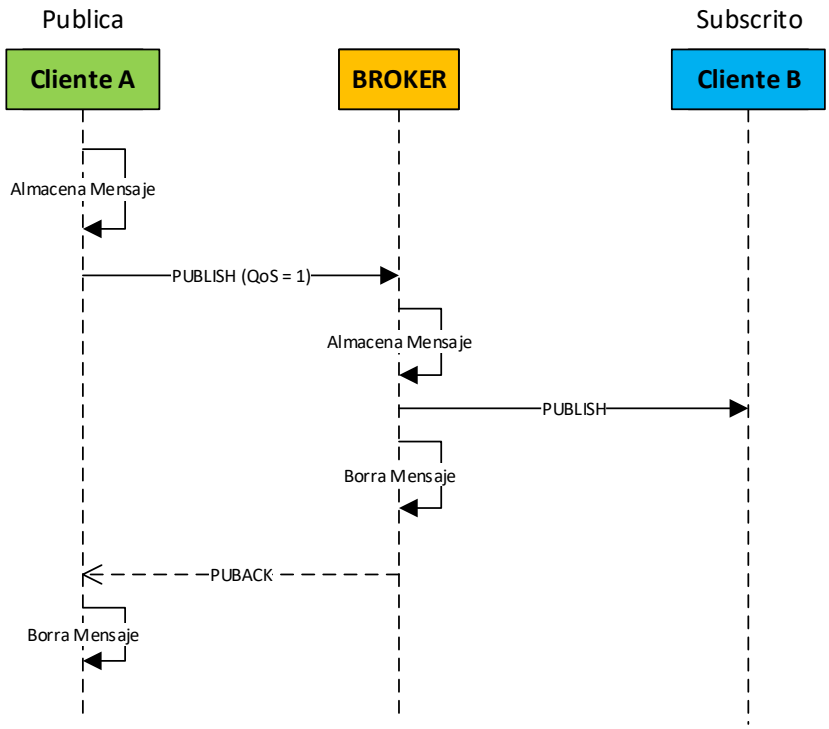


Figura 42: Diagrama de flujo QoS=1
Fuente: Elaborada por los autores

QoS 2 - Exactamente una entrega (entrega garantizada).

Este nivel añade una garantía más del QoS 1, asegurándose que los subscriptores no reciban mensajes duplicados; este es el nivel más alto de garantía que proporciona MQTT, pero la más lenta. Por lo tanto, solo se lo debe usar cuando el suscriptor no deba recibir publicaciones duplicadas bajo ninguna circunstancia.

La figura 43 describe el mecanismo de QoS=2 de una manera gráfica. Con este nivel de seguridad, el mensaje debe almacenarse localmente en el emisor y el receptor hasta que se procese. Asimismo, debe realizarse como mínimo dos pares de transmisiones entre el emisor y el receptor antes de que el mensaje pueda suprimirse de la parte del emisor. El mensaje puede procesarse en el receptor tras la primera transmisión. En el primer par de transmisiones, el emisor transmite el mensaje y obtiene acuse de recibo del receptor que ha almacenado el mensaje. Si el emisor no recibe un acuse de recibo, el mensaje se envía de nuevo con el distintivo DUP establecido hasta que se reciba un acuse de recibo. En el segundo par de transmisiones, el emisor comunica al receptor que puede completar el proceso del mensaje "PUBREL". Si el emisor no recibe acuse de recibo del mensaje "PUBREL", el

mensaje "PUBREL" se envía de nuevo hasta que se recibe aquél. El emisor suprime el mensaje que guardó al recibir el acuse de recibo para el mensaje "PUBREL". El receptor puede procesar el mensaje proporcionado en la primera o segunda fase, no tiene que volver a procesarlo. Si el receptor es un intermediario, publica el mensaje a los suscriptores. Si el receptor es un cliente, el mensaje se entrega a la aplicación de suscriptor. El receptor devuelve un mensaje de finalización al emisor para comunicarle que ha terminado de procesar el mensaje.

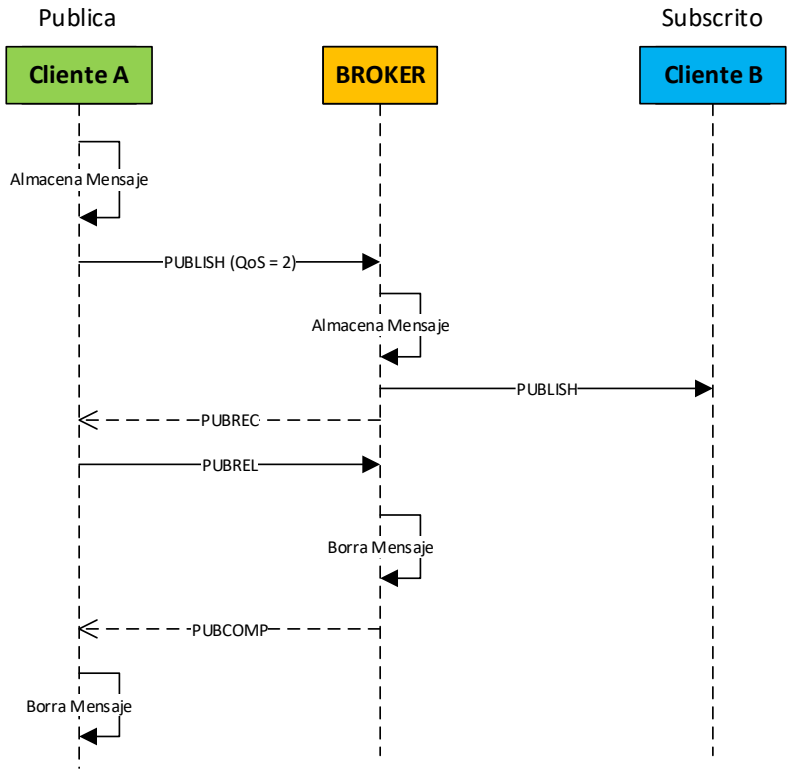


Figura 43: Diagrama de flujo QoS=2
Fuente: Elaborada por los autores

Retain

Es un indicador usado únicamente cuando se publica un mensaje, es decir en mensajes tipo PUBLISH. Este indicador tiene únicamente dos valores posibles 0 y 1. La dinámica de este indicador hace que cuando un cliente envíe un mensaje al bróker, si el indicador RETAIN = 1, el bróker deberá mantener el último mensaje que ha sido enviado a los actuales subscriptores; permitiendo de esta manera que cuando exista una nueva subscripción para el tópico, el último mensaje retenido debería ser enviado al nuevo subscritor. En cambio, cuando el indicador RETAIN = 0, ningún mensaje es enviado ante el evento de una nueva subscripción.

Remaining length

Es un campo que comienza en el segundo byte de la cabecera fija. Este campo puede tener una longitud de 1 a 4 bytes, el cual indica el número de bytes restantes en el mensaje; incluido las longitudes de la cabecera variable (variable header) y la del payload.

2. Cabecera variable (variable header)

El contenido de la cabecera variable depende del tipo de mensaje, es decir solo algunos paquetes MQTT tienen este componente (ver figura 44). El contenido de Variable Header reside entre Fixed Header y Payload. Asimismo, es importante mencionar que el contenido de esta componente varía dependiendo del tipo de paquete (Packet type); sin embargo, un campo llamado “Packet Identifier”, es común para todos los tipos de paquetes (OASIS, 2013).

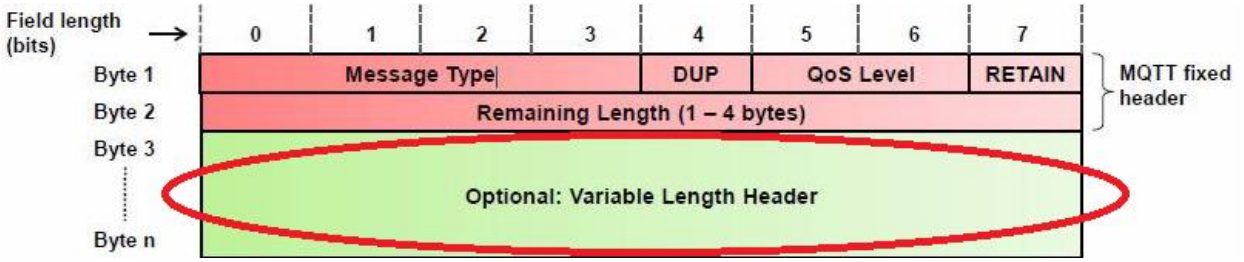


Figura 44: Estructura de la cabecera variable
Fuente: Tomada de MQTT Versión 3.1.1**.** (OASIS, 2013)

3. Payload

Este componente constituye el contenido del mensaje. Es posible enviar imágenes, textos en cualquier codificación, datos encriptados, e incluso en binario (ver figura 45).

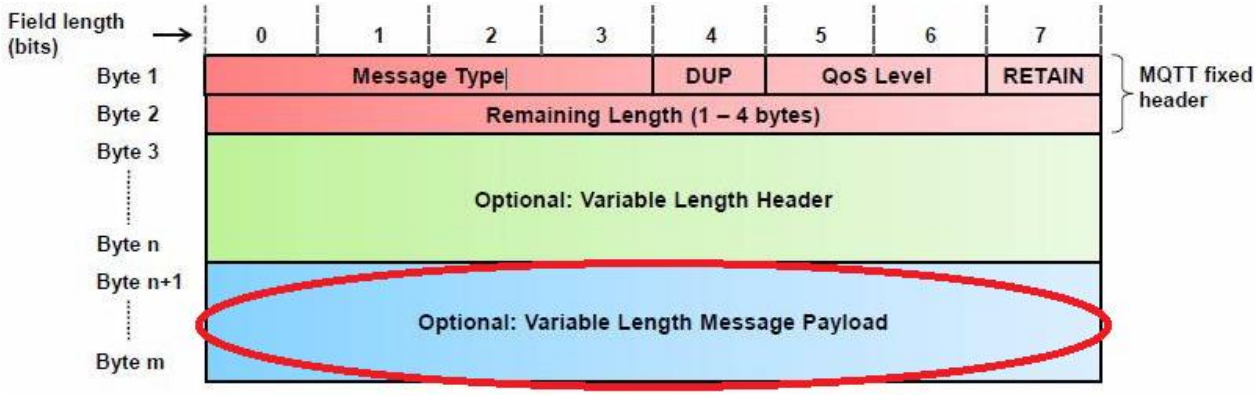


Figura 45: Estructura del *payload*
Fuente: Tomada de MQTT Versión 3.1.1**.** (OASIS, 2013)

ANEXO # 2: Planificación de la solución

Versión 1.0

1. Producto *BackLog*

A continuación, se detalla la primera versión del *Product Backlog*, conforme se vaya avanzando con el proyecto y completando las historias de usuario este podrá sufrir modificaciones en cada iteración, en la tabla 30, se muestra las tareas definidas para este *Backlog*.

Tabla 30: Tareas definidas para el *product backlog*

Identificador de Objetivo	Descripción	Prioridad	Sprint
O1	Conexión con bróker	Media	1
O2	Evaluación de tramas y deserialización	Media	1
O3	Almacenamiento de tramas en base de datos	Alta	1
O4	Definir y crear una interfaz grafica	Alta	2
O5	Autentificación de usuarios	Alta	2
O6	Crear usuarios	Alta	2
O7	Dar de baja usuarios	Media	2
O8	Asignar roles	Media	2
O9	Crear estaciones meteorológicas	Alta	3
O10	Crear variables medioambientales	Alta	3
O11	Descarga de historial de variable medioambiental	Media	3
O12	Visualizador de datos históricos	Media	3
O13	Crear alertas	Media	4
O14	Visualizador de alertas	Media	4
O15	Notificador de alertas	Alta	4
O16	Definir y crear una interfaz grafica	Media	5
O17	Clusterización de estaciones	Media	5
O18	Consumo de servicio 52North	Media	5

Fuente: Elaborado por los autores.

2. Roles y responsabilidades

A continuación, se describen las principales responsabilidades de cada uno de los puestos en el equipo *scrum* durante el desarrollo de las iteraciones, de acuerdo con los roles que desempeñan *scrum*, en la tabla 31, se presenta la descripción de cada rol, así como quien es responsable de ejecutarlo.

Tabla 31: Roles y responsabilidades en *scrum*

Rol	Responsabilidades	Responsable
Interesados	Brindar los requisitos del sistema	Manuel Quiñones Víctor González Luis Santiago Quiñones
Dueño de producto (*product owner*)	Definir los objetivos del producto o proyecto.	Manuel Quiñonez
Jefe de proyecto (*Scrum master*)	Realizar el seguimiento de los procesos.	Víctor González
Equipo de desarrollo	Realizar el seguimiento de los	Felipe Quiñonez

(*Development*)	procesos. Grupo de profesionales con los conocimientos técnicos necesarios y que desarrollan el proyecto de manera conjunta. Ejecutar buenas prácticas.	Byron Figueroa

Fuente: Elaborado por los autores.

3. Desarrollo

En esta sección se va a detallar como el proyecto fue afrontado, se presenta las historias de usuario por cada una de las iteraciones que se fueron implementadas a lo largo del proyecto.

Iteración 0

La primera iteración denominada “Iteración 0” tiene como objetivo realizar los preparativos previos a comenzar el desarrollo, se elaborará la definición de los requisitos del producto desde la perspectiva del usuario, en la tabla 32, se muestra las tareas que se realizaron en este Sprint.

Tabla 32: Tareas de las Iteración 0

Identificador	Tipo	Descripción	Responsable	Estimación
1	Investigación	Elección de tecnologías y herramientas	Felipe Quiñonez Byron Figueroa	20%
2	Investigación	Definición de requisitos técnicos	Felipe Quiñonez Byron Figueroa	20%
3	Investigación	Toma de decisiones	Felipe Quiñonez Byron Figueroa	20%
4	Implementación	Especificación de requerimientos	Felipe Quiñonez Byron Figueroa	40%

Fuente: Elaborado por los autores.

Iteración 1: subsistema de almacenamiento

Esta iteración consiste en el desarrollo del subsistema de almacenamiento de variables medioambientales (SAVA), teniendo como objetivo almacenar los valores censados provenientes de las estaciones meteorológicas. Las funcionalidades que se deben implementar son: conexión al bróker, evaluar trama, convertir trama a sensorML, almacenar en MongoDB y almacenar en postgreSQL.

Planificación

Del Anexo 3. Especificación de Requerimientos, obtenida durante el Iteración 0, se selecciona las funcionalidades correspondientes al módulo de almacenamiento que serán implementadas en este Iteración, en la tabla 33, se muestra las historias de usuario que se realizaron en esta iteración.

Tabla 33: Historias de usuario - iteración 1 subsistema de almacenamiento

Identificador	Historia de usuario	Responsable	Descripción	Estimación
1	Conexión del sistema con el bróker	Felipe Quiñonez Byron Figueroa	Es necesario que el sistema pueda conectarse al bróker y subscribirse a el tópico establecido para que pueda recibir las tramas.	10%
2	Evaluación de tramas y deserialización	Felipe Quiñonez Byron Figueroa	Cuando el sistema reciba una trama de las estaciones metrológicas debe ser capaz de evaluarla a través de un patrón definido para que comience a deserializar la trama.	10%
3	Almacenar variable medioambiental	Felipe Quiñonez Byron Figueroa	Es necesario que la trama deserializada se almacene en una base de datos mongoDB.	40%
4	Almacenar Observación	Felipe Quiñonez Byron Figueroa	Es necesario que la trama se convierta al formato sensorML y a continuación se envié al servicio 52North a través de una petición POST.	40%

Fuente: Elaborado por los autores.

El modelo de datos implementado para este *sprint* está disponible en el Anexo 4 en la sección llamada Vista lógica de datos.

Iteración 2: módulo de administración

Se tiene como objetivo implementar las funcionalidades requeridas para la administración de recursos utilizados por el subsistema de gestión de estaciones meteorológicas y variables medioambientales, esto es: crear, eliminar, modificar usuarios y perfiles para controlar el acceso a los diferentes módulos del sistema.

Planificación

Para la organización de esta iteración, se llevó a cabo una reunión con los interesados. En esta reunión se realiza un análisis de los procesos y las funcionalidades que serán implementadas. En la tabla 34 se presenta las historias de usuario que se siguieron en esta iteración.

Tabla 34: Historias de usuario - iteración 2 módulo de administración

Identificador	Historia de usuario	Responsable	Descripción	Estimación
1	Creación de perfiles	Felipe Quiñonez Byron	Es necesario manejar niveles de acceso para los usuarios, y para ellos es importante manejar	20%

		Figueroa	usuarios agrupados en perfiles.	
2	Iniciar sesión en el sistema	Felipe Quiñonez Byron Figueroa	Consiste en brindar seguridad al acceso de la aplicación, permitiendo que únicamente usuarios autorizados puedan tener acceso al mismo.	30%
3	Administración de usuarios	Felipe Quiñonez Byron Figueroa	El sistema requiere de un número variable de usuarios con diferentes funciones, por lo que es necesario administrar los usuarios y agruparlos en perfiles con diferentes niveles de acceso a la aplicación.	30%
3	Creación de perfiles	Felipe Quiñonez Byron Figueroa	Es necesario manejar niveles de acceso para los usuarios, y para ellos es importante manejar usuarios agrupados en perfiles.	20%

Fuente: Elaborado por los autores.

El modelo de datos implementado para este sprint lo puede consultar en el Anexo 3 en la sección llamada Vista lógica de datos.

Iteración 3: módulo de gestión de estaciones y variables medioambientales

Se tiene como objetivo implementar las funcionalidades requeridas para la gestión de estaciones meteorológicas, esto es: crear estaciones meteorológicas, crear variables medioambientales, localizar estación meteorológica, descargar datos históricos de variable medioambiental y presentar datos actuales de una estación.

Planificación

Del Anexo 3. Especificación de Requerimientos Funcionales obtenida durante el Iteración 0, se selecciona las funcionalidades correspondientes al módulo de gestión que serán implementadas en este Iteración. En la tabla 35, se muestra las historias de usuario que se realizaron en esta iteración.

Tabla 35: Historias de usuario - iteración 4 módulo de gestión de estaciones

Identificador	Historia de usuario	Responsable	Descripción	Estimación
1	Crear estaciones meteorológicas,	Felipe Quiñonez Byron Figueroa	Consiste en crear una nueva estación metrológica, estableciendo parámetros necesarios como: nombre, código, descripción.	20%
2	Crear variables medioambientales.	Felipe Quiñonez Byron Figueroa	La plataforma requiere que se establezca cuáles son las variables medioambientales (sensores) que van a capturar las	20%

			estaciones, para lo cual se deben establecer ciertos parámetros obligatorios (nombre, código, unidad, tipo)	
3	Localizar estación meteorológica.	Felipe Quiñonez Byron Figueroa	La plataforma permitirá visualizar la localización de la estación, donde se encuentre ubicada.	10%
4	Descargar datos históricos de variable meteorológica - JSON	Felipe Quiñonez Byron Figueroa	Consiste en poder descargar los datos almacenados de una variable medioambiental (sensor), estableciendo un periodo de tiempo. Los datos se descargarán en formato *json*.	5%
5	Descargar datos históricos de variable meteorológica - CSV	Felipe Quiñonez Byron Figueroa	Consiste en poder descargar los datos almacenados de una variable meteorológica (sensor), estableciendo un periodo de tiempo. Los datos se descargarán en formato csv.	5%
6	Visualizador de datos actuales.	Felipe Quiñonez Byron Figueroa	La plataforma, podrá presentar los últimos datos que han sido capturados por la estación meteorológica, y establecer cuáles son los valores máximos y mínimos que han sido capturados por cada variable medioambiental.	20%
7	Visualizador de datos históricos	Felipe Quiñonez Byron Figueroa	La plataforma, podrá presentar un histórico de los datos que han sido capturados por la estación meteorológica en un periodo de tiempo.	20%

Fuente: Elaborado por los autores.

Iteración 4: módulo de notificación

La cuarta Iteración tiene como objetivo implementar las funcionalidades requeridas para la gestión de alertas, esto es; crear, borrar, editar alerta, administrar alerta.

Planificación

Del Anexo 3, Especificación de Requerimientos Funcionales, obtenida durante el Iteración 0, se selecciona las funcionalidades correspondientes al módulo de gestión que serán implementadas en este Iteración, en la tabla 36, se muestra las historias de usuario que se realizaron en esta iteración.

Tabla 36: Historias de usuario - iteración 4 módulo de notificación

Identificador	Historia de usuario	Responsable	Descripción	Estimación
1	Crear alerta	Felipe	Es necesario que los usuarios	10%

		Quiñonez Byron Figueroa	puedan establecer alertas, dependiendo de las condiciones que ellos crean convenientes. Como además de ser notificados por el medio que ellos definan.	
2	Visualizar alertas	Felipe Quiñonez Byron Figueroa	Permite observar que alertas han sido creadas para cada variable medioambiental.	10%
3	Notificador Email	Felipe Quiñonez Byron Figueroa	El sistema notificara al usuario cuando una alerta ha sido activada, enviándole un correo electrónico	20%
4	Notificador SMS	Felipe Quiñonez Byron Figueroa	El sistema notificara al usuario cuando una alerta ha sido activada, enviándole un SMS.	20%
5	Consola de Alertas.	Felipe Quiñonez Byron Figueroa	Es necesario que el sistema disponga de una funcionalidad que permita evaluar con frecuencia las distintas alertas.	40%

Fuente: Elaborado por los autores.

Iteración 5: subsistema de visualización de datos

Esta iteración cubre el desarrollo del subsistema de visualización de datos, teniendo como objetivo la construcción de un sitio web donde los usuarios puedan ver los datos almacenados en el sistema. Las funcionalidades requeridas para esto son: cauterización de estaciones, consumo de servicios de 52North y consumo de servicio *rest*. En la tabla 37, se muestra las historias de usuario que se realizaron en esta iteración.

Tabla 37: Historias de usuario - iteración 5 subsistema de visualización de datos

Identificador	Historia de usuario	Responsable	Descripción	Estimación
1	Consumo de servicio *rest*	Felipe Quiñonez Byron Figueroa	Permite obtener las estaciones meteorológicas almacenadas en el sistema.	40%
2	Consumo de servicios 52North	Felipe Quiñonez Byron Figueroa	Permite obtener los datos almacenados en formato SOS, para su posterior visualización mediante una serie de tiempo.	40%
3	Clusterización de	Felipe	Es necesario que las estaciones	20%

	estaciones	Quiñonez Byron Figueroa	meteorológicas que se presentan en pantalla se agrupen acorde al lugar donde se encuentran.	

Fuente: Elaborado por los autores.

ANEXO # 3: Especificación de requerimientos de software

Especificación de Requerimientos Software

Plataforma Web para el Monitoreo de Estaciones Meteorológicas

HISTORIA DEL DOCUMENTO

Fecha	Versión	Comentarios	Autor
	0.1	Creación del documento de especificación de requerimientos	Felipe Quiñones, Byron Figueroa
	1.0	Corrección del documento de especificación de requerimientos	Byron Figueroa
	1.1	Corrección del documento de especificación de requerimientos	Felipe Quiñones
	1.2	Versión Final del Documento	Felipe Quiñones, Byron Figueroa

1. Introducción

El presente documento tiene como propósito definir las especificaciones funcionales, y no funcionales y del sistema para la implementación de una aplicación WEB que permitirá administrar y consultar la información de estaciones meteorológicas, utilizadas por el personal de la UTPL.

2. Alcance

Diseño, desarrollo e implantación del sistema SesMeter (Sistema de Estaciones Meteorológicas). El SesMeter será una aplicación que funcionará en un entorno WEB que permitirá administrar y consultar la información de estaciones meteorológica, administradas por la UTPL. Esta aplicación dará apoyo a los siguientes procesos:

- Administrar estaciones meteorológicas.
- Administrar variables medioambientales
- Gestión de Alertas.
- Gestión de Descargas.
- Administrar usuarios del sistema.

3. Resumen de requerimientos software

	Requerimientos Funcionales
RF01	Autenticación de usuarios.
RF02	Gestión de estaciones meteorológicas.
RF03	Gestión de variables medioambientales.
RF04	Visualizar información meteorológica.
RF05	Gestión de descargas.
RF06	Gestión de alertas.
RF07	Almacenar tramas de estaciones meteorológicas
RF08	Gestión de Notificaciones de Alertas.
RF09	Envió de observaciones al sistema SOS.

	Requerimientos No Funcionales
RF01	Rendimiento
RF02	Seguridad
RF03	Disponibilidad
RF04	Portabilidad

RF05	Fiabilidad
RF06	Mantenibilidad

4. Detalle de Requerimientos funcionales

RF01. Autenticación de usuarios

Declaración de función: el acceso al sistema deberá ser restringido a usuarios autorizados. Para ello, el sistema deberá solicitar el usuario y contraseña del usuario para validar su ingreso (ver tabla 38).

Tabla 38: Especificación de RF01: autenticación de usuarios.

	Especificaciones
Entradas	
1	Registro de Usuario.
2	Usuario (string)
3	Contraseña (string)
Proceso	
1	Validar las credenciales ingresadas (usuario, contraseña) del usuario, contra la base de datos del sistema.
2	Presentar las funcionalidades al usuario, dependiendo del rol de usuario al que pertenezca.
Salidas	
1	Si la validación de usuario y contraseña es correcta, el usuario podrá ingresar al sistema, además las funcionalidades se presentarán de acuerdo al rol que representa.
2	Si la validación de usuario y contraseña es invalida, se presenta un mensaje de error, el cual comunica que las credenciales ingresadas son incorrectas.

Fuente: Elaborado por los autores.

RF02. Gestión de estaciones meteorológicas (crear, modificar, eliminar)

Declaración de función: las estaciones meteorológicas deben ser registradas en el sistema, además se deberá permitir la asignación de las variables medioambientales a una estación. Además, se deberá permitir la actualización de los campos ingresados (ver tabla 39).

Tabla 39: Especificación de RF02: gestión de estaciones meteorológicas.

	Especificaciones
Entradas	
1	Nombre (string)
2	Código (string)
3	Descripción (string)
4	Latilud (string)
5	Longitud (string)
6	Variables medioambientales [object]
Proceso	
1	Se registra la información de la estación meteorológica, • Se realiza una comprobación de los datos ingresados antes de que se almacenen, • Almacena la información de la estación meteorológica en la base de datos.
2	Se modifica la información de la estación Meteorológica. • Se realiza una comprobación de los datos ingresados antes de que se almacenen, el campo código debe ser único así que no se lo puede volver a repetir. • Almacena la información de la estación meteorológica en la base de datos.

3	Se elimina la información de la variable medioambiental. • Se selecciona la variable medioambiental que se desea eliminar. • Elimina la estación Meteorológica en la base de datos.
Salidas	
1	Si los datos fueron ingresados de una forma correcta, habilitar el botón ***Guardar*** el cual permitirá almacenar los datos, luego presentar en pantalla un menaje indicando que la estación ha sido creada exitosamente; posteriormente se podrá observar a la nueva estación meteorológica en la pantalla.
2	Si los datos fueron ingresados de una forma incorrecta o el campo ***Código*** se encuentra repetido, se deshabilitará el botón ***Guardar*** y se presentara en pantalla el mensaje "el Código ya se encuentra registrado"; únicamente se habilitará el botón guardar cuando se cambie el código o se ingrese el campo marcado de acuerdo a lo establecido.
3	Cuando se haya eliminada la variable medioambiental, presentara en pantalla un mensaje de confirmación de que realizado esta acción.

Fuente: Elaborado por los autores.

RF03. Gestión de variables medioambientales (crear, modificar, eliminar)

Declaración de función: las variables medioambientales deben ser registradas en el sistema, así como pueden ser actualizados sus campos. Además, las variables medioambientales podrán ser asociadas a determinada estación meteorológica (ver tabla 40).

Tabla 40: Especificación de RF03: gestión de variables medioambientales.

	Especificaciones
Entradas	
1	Nombre (string)
2	Código (string)
3	Unidad (string)
4	Tipo (VariableTipo)
5	Descripción (string)
6	Icono (string)
Proceso	
1	Se registra la información de la variable ambienta, • Se realiza una comprobación de los datos para que cumplan condiciones establecidas para cada campo antes de que se almacenen, el campo ***código*** debe ser único así que no se lo puede volver a repetir. • Almacena la información de la variable medioambiental, en la base de datos.
2	Se modifica la información de la estación Meteorológica. • Se realiza una comprobación de los datos ingresados antes de que se almacenen, el campo código debe ser único así que no se lo puede volver a repetir. • Almacena la información de la estación meteorológica en la base de datos.
3	Se elimina la información de la variable Meteorológica. • Se selecciona la variable Meteorológica que se desea eliminar. • Elimina la variable Meteorológica en la base de datos.
Salidas	
1	Si los datos fueron ingresados de una forma correcta, se habilitará el botón ***Guardar*** el cual permitirá almacenar los datos, luego presentara en pantalla un menaje indicando que la variable medioambiental, ha sido creada exitosamente y se podrá observar a la nueva variable medioambiental, en la pantalla.
2	Si los datos fueron ingresados de una forma incorrecta o el campo ***Código*** se encuentra repetido, se deshabilitará el botón ***Guardar*** y se presentara en pantalla el mensaje "el Código ya se encuentra registrado"; únicamente se habilitará el botón guardar cuando se cambie el código o se ingrese el campo marcado de acuerdo a lo establecido.

3	Cuando se haya eliminada la variable Meteorológica, presentara en pantalla un mensaje de confirmación de que realizado esta acción.

Fuente: Elaborado por los autores.

RF04. Visualizar información meteorológica

Declaración de función: el sistema ofrecerá al público en general la visualización de la información referente a las variables medioambientales almacenadas en las diferentes estaciones meteorológicas (ver tabla 41).

Tabla 41: Especificación de RF04: visualizar información meteorológica.

	Especificaciones
Entradas	
1	Estación Meteorológica.
Proceso	
1	Se selecciona una estación meteorológica, la cual se requiera visualizar.
Salidas	
1	Al seleccionar una estación meteorológica, se presenta en pantalla todas las variables medioambientales con los últimos datos que están han sido recopilados.

Fuente: Elaborado por los autores.

RF05. Gestión de descargas.

Declaración de función: el sistema debe permitir realizar la descarga del histórico almacenado de una variable medioambiental asociada determinada estación meteorológica. Para lo cual es necesario especificar el rango de las fechas de las cuales se quiere obtener los datos (ver tabla 42).

Tabla 42: Especificación de RF05: gestión de descargas.

	Especificaciones
Entradas	
1	Código de estación (string)
2	Código de Variable (string)
3	Desde (date)
4	Hasta (date)
5	Formato (string)
Proceso	
1	Se selecciona una estación meteorológica, la cual se requiera descargar su historial. Luego, se debe seleccionar el rango de fechas de las cuales se requiera obtener los datos, y finalmente se escoge el tipo de formato el cual se necesite descargar.
Salidas	
1	Se presenta en pantalla un mensaje comunicando que se están descargando los datos, y dependiendo del formato que se haya seleccionado se descargará un archivo con la extensión de ese formato.
2	Si los rangos de las fechas se encuentran mal establecidos, presentar un mensaje comunicando que se ingrese correctamente estos campos.

Fuente: Elaborado por los autores.

RF06. Gestión de alertas.

Declaración de función: las alertas deben ser registradas en el sistema, así como se puede

actualizar la condición establecida (ver tabla 43).

Tabla 43: Especificación de RF06: gestión de alertas.

	Especificaciones
Entradas	
1	Tipo (EventoTipo)
2	Condición (EventoCondicion)
3	Tiempo Inactividad (BigInteger)
4	Resultado (string)
Proceso	
1	Se registra la información de alerta y se establece el tipo de alerta que se desee. • Si la alerta es ***Condicional***, se presentará dos nuevos campos que deberán ser llenados, estos son *condición* y *resultado*. • Si la alerta es por ***Inactividad***, se presentará un nuevo campo llamado tiempo de inactividad el cual deberá ser completado.
2	Se modifica la información de la alerta. • Se realiza una comprobación de los datos cumplan condiciones establecidas para cada campo antes de que se almacenen.
3	Cuando se haya eliminada la Alerta, presentara en pantalla un mensaje de confirmación de que realizado esta acción.
Salidas	
1	Si los datos fueron ingresados de forma correcta, habilitara el botón ***Guardar*** el cual nos permitirá almacenar los datos, luego presentara en pantalla un menaje indicando que la Alerta (evento), ha sido creada exitosamente y podremos observarlo en el listado que se presenta con todas las alertas establecidas.
2	Si los datos fueron ingresados de una forma incorrecta, se deshabilitará el botón Guardar y presentara en pantalla cual campo falta por completar o esta incorrecto, solo se habilitará el botón guardar hasta que se ingrese correctamente los campos de acuerdo a lo establecido.

Fuente: Elaborado por los autores.

RF07. Gestión de notificación de alertas.

Declaración de función: el sistema deberá ser capaz de enviar notificaciones vía mail o SMS cuando una alerta ha cumplido las condiciones establecidas, solo las notificaciones que son vía Mail están activadas para todos los usuarios mientras que las notificaciones por SMS solo el administrador de la plataforma será capaz de activarlas para un usuario especifico (ver tabla 44).

Tabla 44: Especificación de RF07: gestión de notificaciones de alertas.

	Especificaciones
Entradas	
1	Alerta, condición establecida
Proceso	
1	El sistema revisa constantemente todas las alertas creadas.
2	Cuando compara que una de estas alertas ha cumplido las condiciones establecidas.
3	El sistema envía un mensaje al usuario que estableció la alerta.
Salidas	
1	Si el usuario solo tiene activado las notificaciones por Email, le llegara a un correo electrónico indicándole cuando se cumplió la alerta.
2	Si el usuario solo tiene activado las notificaciones por SMS, le llegara al celular un mensaje indicándole cuando se cumplió la alerta.

3	Si el usuario tiene activado notificaciones por SMS y EMAIL, le llegara a su celular, como a su correo electrónico, un mensaje indicándole cuando se cumplió la alerta.

Fuente: Elaborado por los autores.

RF08. Almacenar tramas de estaciones meteorológicas.

Declaración de función: el sistema SAVA deberá ser capaz de conectarse al bróker MQTT y suscribirse al tópico necesario para poder recibir las tramas, cuando este se haya subscrito deberá comenzar a recibir las tramas que envían cada una de las estaciones meteorológicas, cuando llegue una trama deberá evaluarla para comprobar si el patrón es correcto, si es así deberá de-serializarla y almacenarla en la base de datos mongoDB.

Tabla 45: Especificación de RF08: Almacenar tramas de estaciones meteorológicas.

	Especificaciones
Entradas	
1	Trama de estación meteorológica
Proceso	
1	El sistema se conecta al bróker MQTT.
2	El sistema se subscribe al tópico definido.
3	El sistema recibe las tramas entrantes.
4	El sistema evalúa la trama para ver si tiene el patrón correcto.
5	Si la trama es correcta esta es deserealizada y envía cada variable para que se almacene
Salidas	
1	Si la trama no cumple con el patrón establecido no se almacenará.
2	Si la trama cumple el patrón establecido se almacenará.

Fuente: Elaborado por los autores.

RF09. Almacenar tramas de estaciones meteorológicas al sistema SOS.

Declaración de función: el sistema SAVA deberá ser capaz de conectarse al bróker MQTT y suscribirse al tópico necesario para poder recibir las tramas, cuando este se haya subscrito deberá comenzar a recibir las tramas que envían cada una de las estaciones meteorológicas, cuando llegue una trama deberá evaluarla para comprobar si el patrón correcto, si es así deberá convertirla al lenguaje sensorML y enviarla a través de una petición POST al sistema 52North.

Tabla 46: Especificación de RF09: Almacenar tramas de estaciones meteorológicas en SOS.

	Especificaciones
Entradas	
1	Trama de estación meteorológica
Proceso	
1	El sistema se conecta al bróker Mqtt.
2	El sistema se subscribe al tópico definido.

3	El sistema recibe las tramas entrantes.
4	El sistema evalúa la trama para ver si tiene el patrón correcto.
5	El sistema convierte la trama al lenguaje sensorML.
	El sistema envía una petición POST al sistema 52North en la cual está la trama convertida
Salidas	
1	Si la trama no cumple con el patrón establecido no se almacenará.
2	El sistema 52North responderá con OK si la trama ha sido enviada correctamente.

Fuente: Elaborado por los autores.

5. Detalle de Requerimientos no funcionales

RNF01: Rendimiento

- Se debe garantizar que las consultas u otros procesos no afecte el desempeño de la base de datos.

RNF02: Seguridad

- Uso de contraseña para cada usuario, esto permitirá el ingreso al sistema únicamente a personas autorizadas.
- Registros del ingreso al sistema.
- Creación de roles de usuarios.
- El sistema se debe proveer mecanismos de seguridad que impidan la vulnerabilidad de los datos y garanticen la integridad de la información.

RNF03: Disponibilidad

- El sistema debe ser desarrollado tomando en cuenta las necesidades, requerimientos y objetivos de UTPL, por lo que deberá estar disponible el 100% del tiempo del día.

RNF04: Portabilidad

- Al utilizar herramientas de software libre se está garantizando la portabilidad, además se utilizará el lenguaje Java y base de datos MongoDB.

RNF05: Fiabilidad

- El sistema debe tener una interfaz de uso intuitiva y sencilla.
- El sistema deberá ser capaz de responder ante todo tipo de incidente.

RNF05: Mantenibilidad

- El sistema debe disponer de una documentación que permita realizar operaciones de mantenimiento con el menor esfuerzo posible.

ANEXO # 4: Documento de arquitectura de software

Versión 1.0

Historial de Revisión

Fecha	Versión	Descripción	Autor
01/04/20016	1.0	Creación del Documento de Arquitectura	Quiñones Cuenca Felipe David Figueroa Sarmiento Byron Rafael

1. Introducción

El presente documento describe el diseño de la solución a construir para el presente trabajo de titulación, específicamente enfocado en tener un sistema que permita recolectar la información en tiempo real desde una red de estaciones meteorológicas distribuidas en algunas regiones del país. Se requiere que el sistema sea diseñado y estructurado de manera que sea fácil e intuitiva para el usuario final, así como que tenga un enfoque de escalabilidad. Concretamente, el contenido de este documento presenta la arquitectura del sistema a desarrollar. Para lo cual se subdividió el sistema en tres grandes subsistemas: subsistema de almacenamiento, subsistema de gestión de estaciones meteorológicas y variables medioambientales, y el subsistema de visualización de datos para el público en general.

Propósito

Este documento se propone mostrar todos los aspectos y las características que constituyen la arquitectura del sistema. Este modelo 4+1 tiene una gran importancia, ya que está estrechamente relacionado con los interesados del proyecto de acuerdo al rol dentro del desarrollo del proyecto.

Definiciones, acrónimos y abreviaciones

- RUP: Rational Unified Process.
- SOS - Sensor observation service.
- OS - Operating System.
- API - Application Programming Interface
- SAVA – Sistema de almacenamiento de variables medioambientales.
- FrontEnd: Es la parte del desarrollo web que se dedica de la parte frontal de un sitio web, en pocas palabras del diseño de un sitio web, desde la estructura del sitio hasta los estilos como colores, fondos, tamaños hasta llegar a las animaciones y efectos.

2. Restricciones de la arquitectura

A continuación, se describe las tecnologías y *software* base a utilizar.

Plataformas tecnológicas a utilizar

Bootstrap 3.0: Es un framework para CSS, HTML y Js (javascript); el cual permite el desarrollo de páginas con un diseño adaptativo (responsive design). Esto facilita la maquetación del sitio, así como que el sitio web se despliegue correctamente en toda clase de dispositivos, evitando rediseñar el sitio web.

Angular 1.4.5: Es un framework de Javasript el cual permite crear y mantener la aplicación web en una sola página. Este framework amplía el HTML tradicional para servir el contenido de forma dinámica; con ello se reduce la cantidad de código JavaScript necesario para hacer la aplicación web funcional.

Grunt 0.4.5: Esta librería basada en Javasript permite realizar tareas automatizadas, lo cual ayuda a ahorrar tiempo en desarrollo y despliegue de la aplicación web.

Spring: Es un framework para el desarrollo de aplicaciones basadas en Java. Este framework tiene dos características importantes: es de código abierto también y posee diversos módulos que permiten agregar funcionalidades a la aplicación.

Mongose: Es una librería de Node.js que permitir realizar la conexión entre Node.js y MongoDB.

Software base a utilizar

A continuación, se describe el software base sobre el cual se ejecutará el sistema.

Sistema Operativo Windows server 2008 R2: Aquí se alojará la plataforma web y el servicio de almacenamiento.

Mongo 3.2: Se utilizará esta base de datos como repositorio del sistema de almacenamiento de variables medioambientales.

PostgreSQL: Es un sistema de gestión de bases de datos relacional orientado a objetos, este sistema de base de datos será usada para la implementación del estándar SOS.

Apache Server: Se utilizará para alojar el Subsistema de Visualización de Datos.

Tomcat Server 8.0: Se utilizará para alojar la plataforma de estaciones meteorológicas.

52 North: Es un servicio web que permite obtener datos espaciales de sensores y almacenarlos bajo el estándar SOS.

Herramientas de desarrollo a utilizar

Las herramientas que se utilizaron para el desarrollo de la plataforma de estaciones meteorológicas son las siguientes:

- Red Hat JBoss Developer Studio 8.1.0.GA: entorno de desarrollo integrado y basado en Eclipse, que combina las herramientas con el tiempo de ejecución.
- Jenkins: software de integración continua para el desarrollo de software escrito en Java.

- NodeJs V6.9.2: entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación ECMAScript, asíncrono, con I/O de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google.

3. Representación arquitectónica

El sistema será representado a través del modelo de vistas 4+1, propuesto por Krutchen. Este modelo utiliza el siguiente conjunto de vistas para representar la arquitectura:

Vista de Casos de Uso: Lista los casos de uso o escenarios del modelo que representen funcionalidades centrales del sistema final; específicamente aquellas que requieran una gran cobertura arquitectónica o que impliquen algún punto especialmente delicado en lo concerniente a la arquitectura.

Vista Lógica: Describe las partes arquitectónicamente significativas del modelo de diseño; es decir detalla su composición en capas, subsistemas o paquetes.

Vista de desarrollo: Describe la estructura general del modelo de implementación, mapeo de los subsistemas, paquetes y clases de la vista lógica, y componentes de implementación.

Vista de Procesos: Esta vista describe los procesos que hay en el sistema y la forma en la que se comunican; esta vista puede incluir el diagrama de actividad de UML.

Vista de física: Describe uno o más escenarios de distribución física del sistema, es decir describe cómo es instalada la aplicación y cómo se ejecuta en una red de computadores tomando en cuenta requerimientos no funcionales.

Además de las vistas mencionadas anteriormente, se incluirá una vista adicional a la que propone el modelo 4+1, esta es la vista de lógica de datos.

Vista de lógica de datos: En esta vista se describe la estructura de datos de los cuales está compuesto el sistema, se debe tener en cuenta los requerimientos analizados junto con la Información suministrada por los encargados del proyecto.

4. Vista de casos de uso

Teniendo en cuenta los requerimientos funcionales, se ha elaborado un diagrama de casos de uso; lo que muestra una perspectiva global de las funcionalidades del sistema. En la figura 46, se

representan los casos de uso significativos y los actores relacionados a ellos.

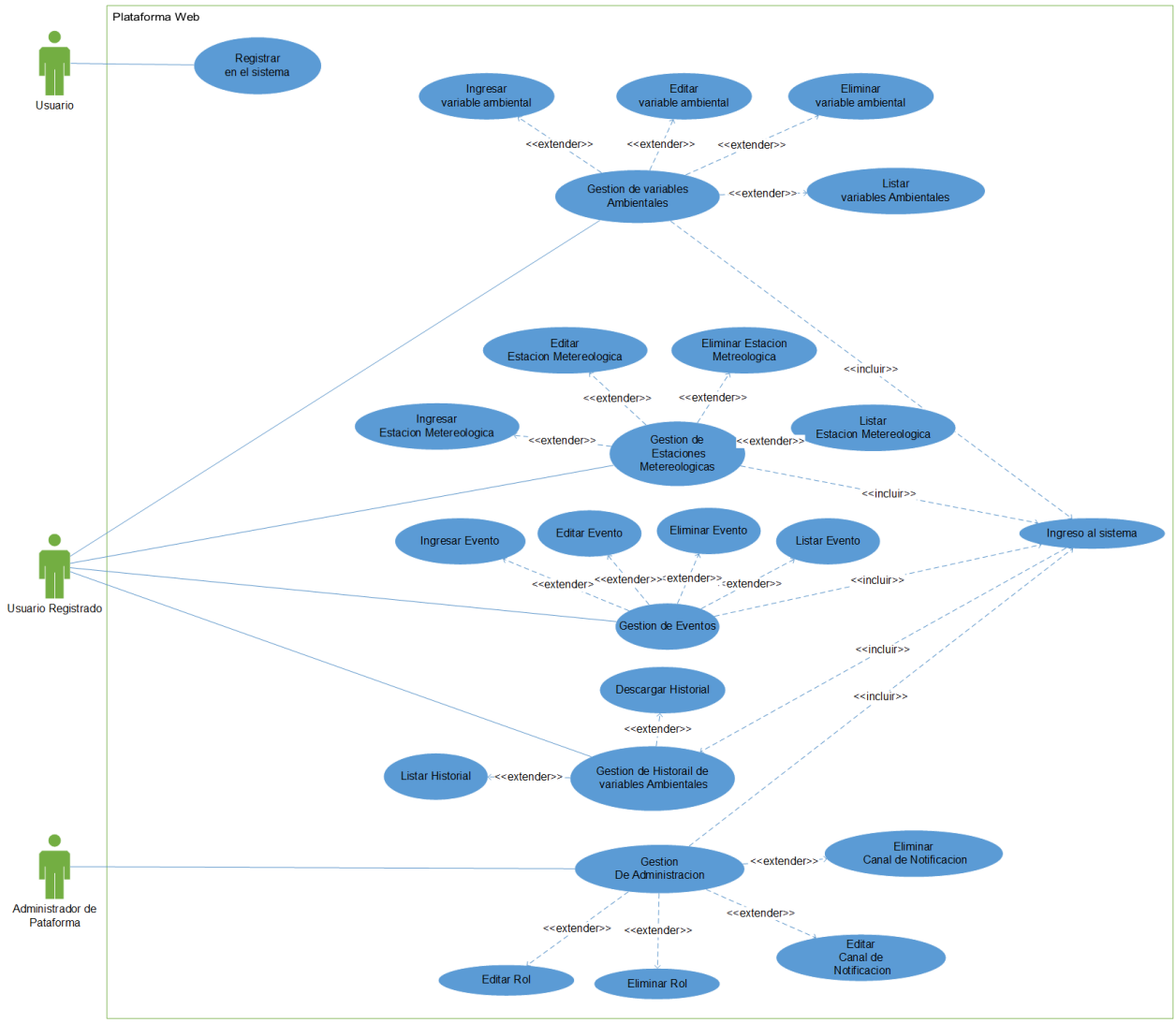


Figura 46: Diagrama general de casos de uso – subsistema de visualización de datos
Fuente: Elaborado por los autores.

Para un mayor entendimiento de los casos de uso se los ha segregado por componentes. A continuación, se detalla cada uno de estos componentes.

Casos de uso – Componente de Gestión de variables

Figura 47: Diagrama de casos de uso – componente de gestión de variables medioambientales
Fuente: Elaborado por los autores.

Diagrama de casos de uso – componente de gestión de estaciones meteorológicas

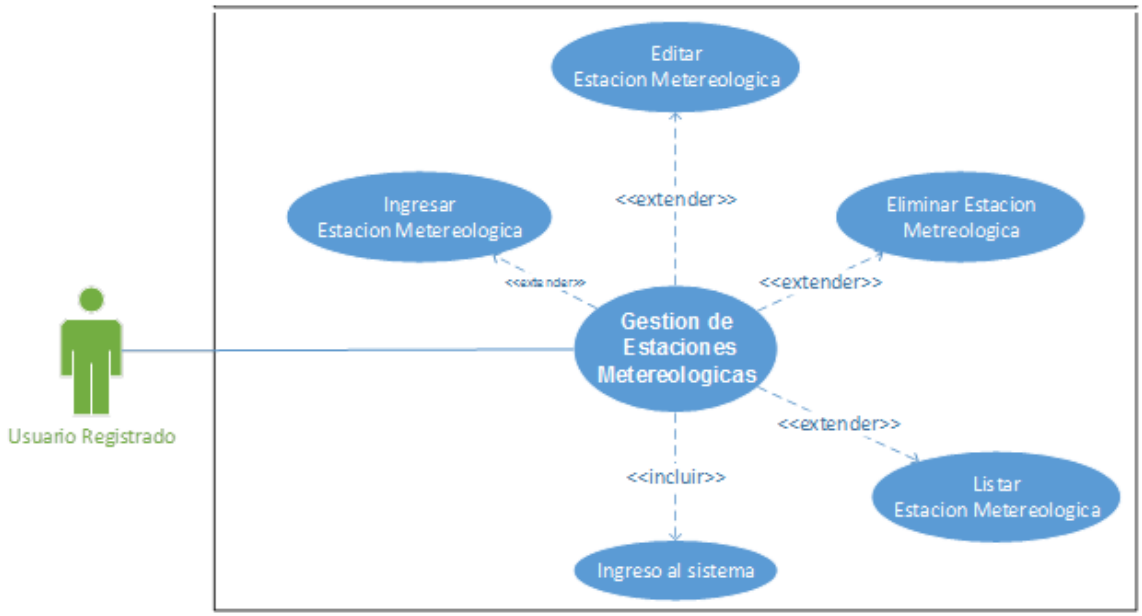


Figura 48: Diagrama de casos de uso – componente de gestión de estaciones meteorológicas
Fuente: Elaborado por los autores.

Diagrama de casos de uso – componente de eventos

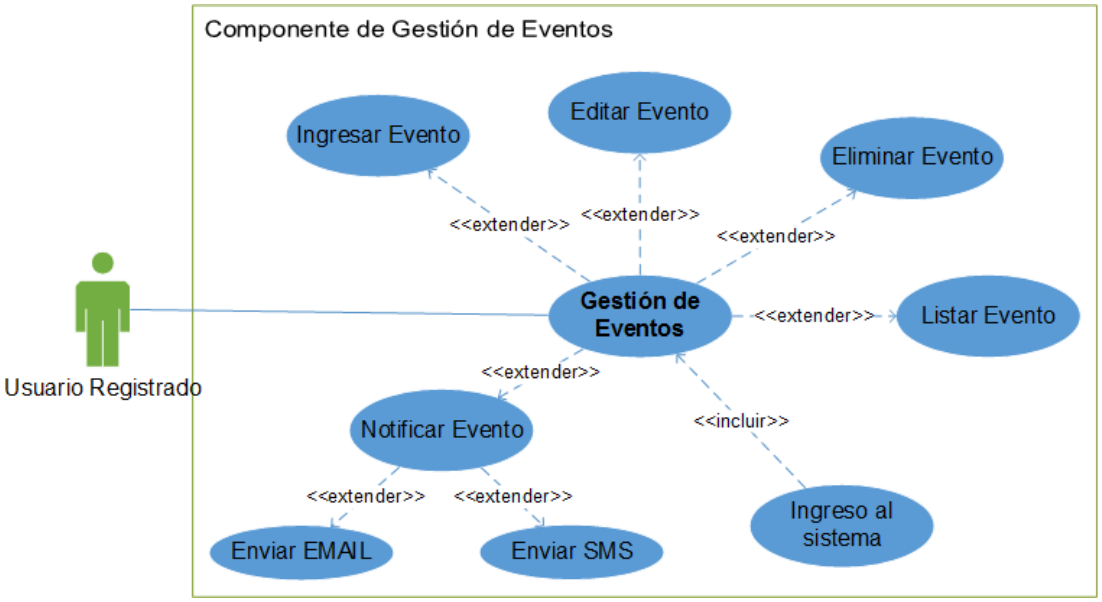


Figura 49: Diagrama de casos de uso - componente de gestión de alertas
Fuente: Elaborado por los autores.

Diagrama de casos de uso – componente de administración

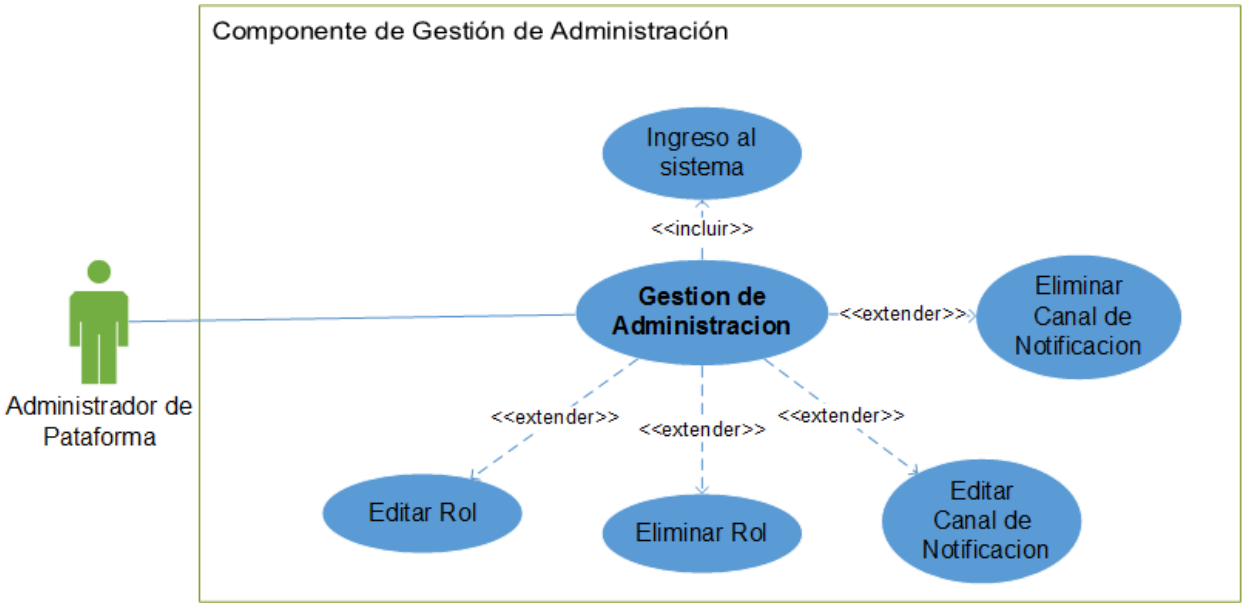


Figura 50: Diagrama de casos de uso - componente de administración
Fuente: Elaborado por los autores.

Diagrama de casos de uso – componente de gestión de historial

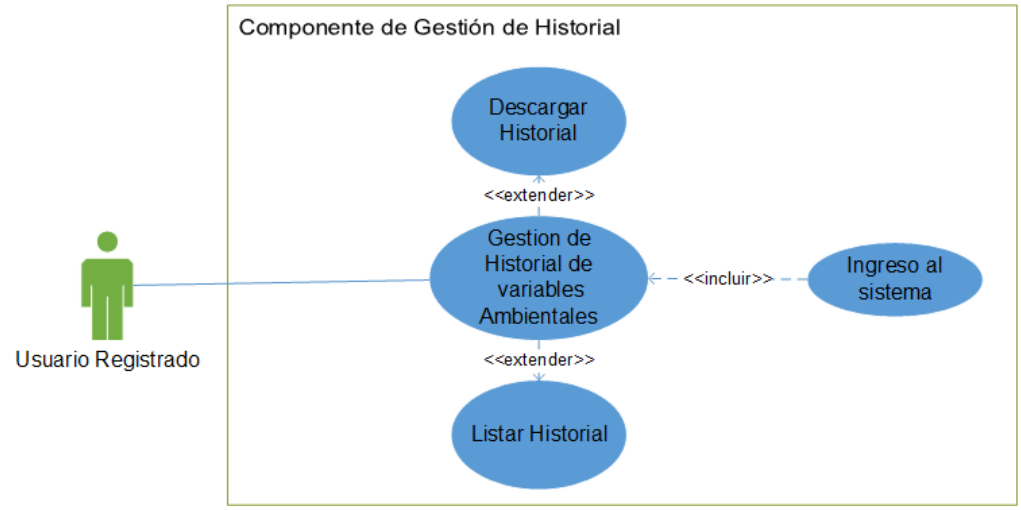


Figura 51: Diagrama de casos de uso - componente de historial
Fuente: Elaborado por los autores.

Diagrama de casos de uso – SAVA

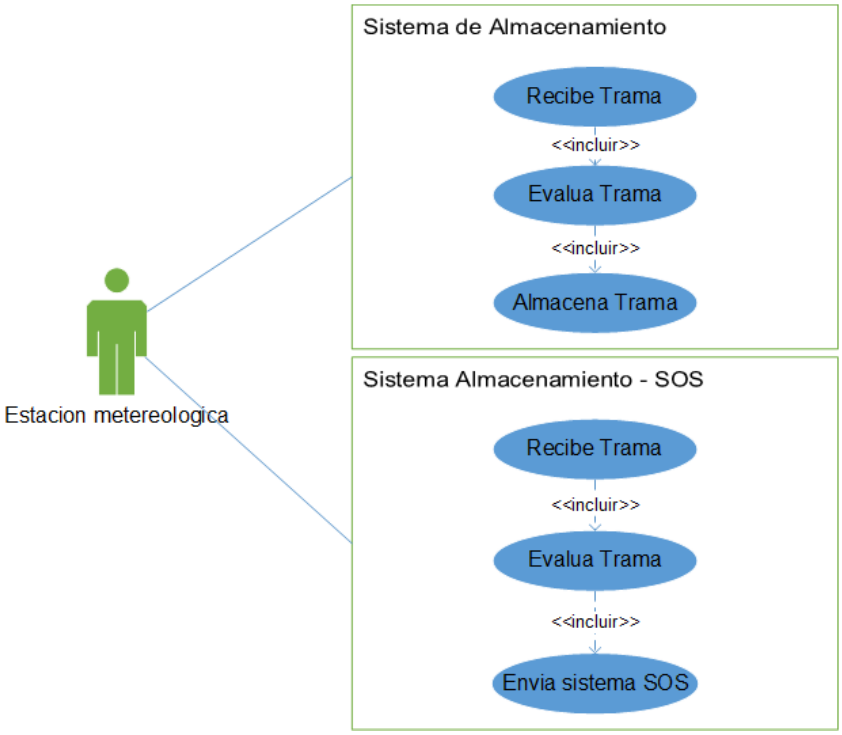


Figura 52: Diagrama de casos de uso – SAVA.
Fuente: Elaborado por los autores.

5. Vista Lógica

La vista lógica se encarga de representar los requerimientos funcionales del sistema. Esta sección describe las partes del diseño del modelo significativas para la arquitectura, tales como subsistemas y paquetes.

Capas Lógicas

Para la plataforma web se ha definido una arquitectura en tres capas. Este tipo de arquitectura es usado en la gran mayoría de sistemas. La ventaja principal que representa este estilo es que permite el desarrollo en varios niveles, y si es necesario algún cambio, solo se realiza modificaciones en el nivel requerido sin tener que revisar entre código mezclado. Las ventajas por las cuales se ha utilizado este tipo de arquitectura de software son las siguientes:

- Cada capa tiene una función específica y no interfiere con las demás, lo cual permite separar responsabilidades.
- La separación de roles en tres capas hace más fácil reemplazar o modificar a una, sin afectar a los módulos restantes.
- Capacidad de migrar el motor de Base de Datos sin tener grandes impactos para el resto de la plataforma.
- Es posible cambiar el FrontEnd sin afectar a la lógica o base de datos de la plataforma.

Para esta arquitectura se han definido las siguientes capas:

***Presentación**:* Esta capa estará basada de acuerdo al Patrón MVC y el Framework Angular que implementa dicho patrón; esta capa es en la que el usuario final interactúa.
Media: Se encargará de ejecutar la lógica de negocio y de brindar los servicios necesarios a la capa de presentación. Esta capa estará desarrollada con el framework Spring que manejará el núcleo de la aplicación.
Datos: Esta capa estará desarrollada utilizando el framework de Spring Data; el cual nos permite la conexión con la base de datos. Aquí se maneja la interacción de las transacciones con la Base de Datos.

Para el sistema SAVA se ha definido una arquitectura basada en eventos, en este tipo de modelo se promueven la producción, detección, consumo de, y reacción a eventos. Un evento es el cambio de un estado, el consumidor tiene la responsabilidad de llevar a cabo una reacción tan pronto como el evento esté presente. El sistema SAVA activa condiciones ante los eventos

entrantes como mensajes provenientes de las estaciones.

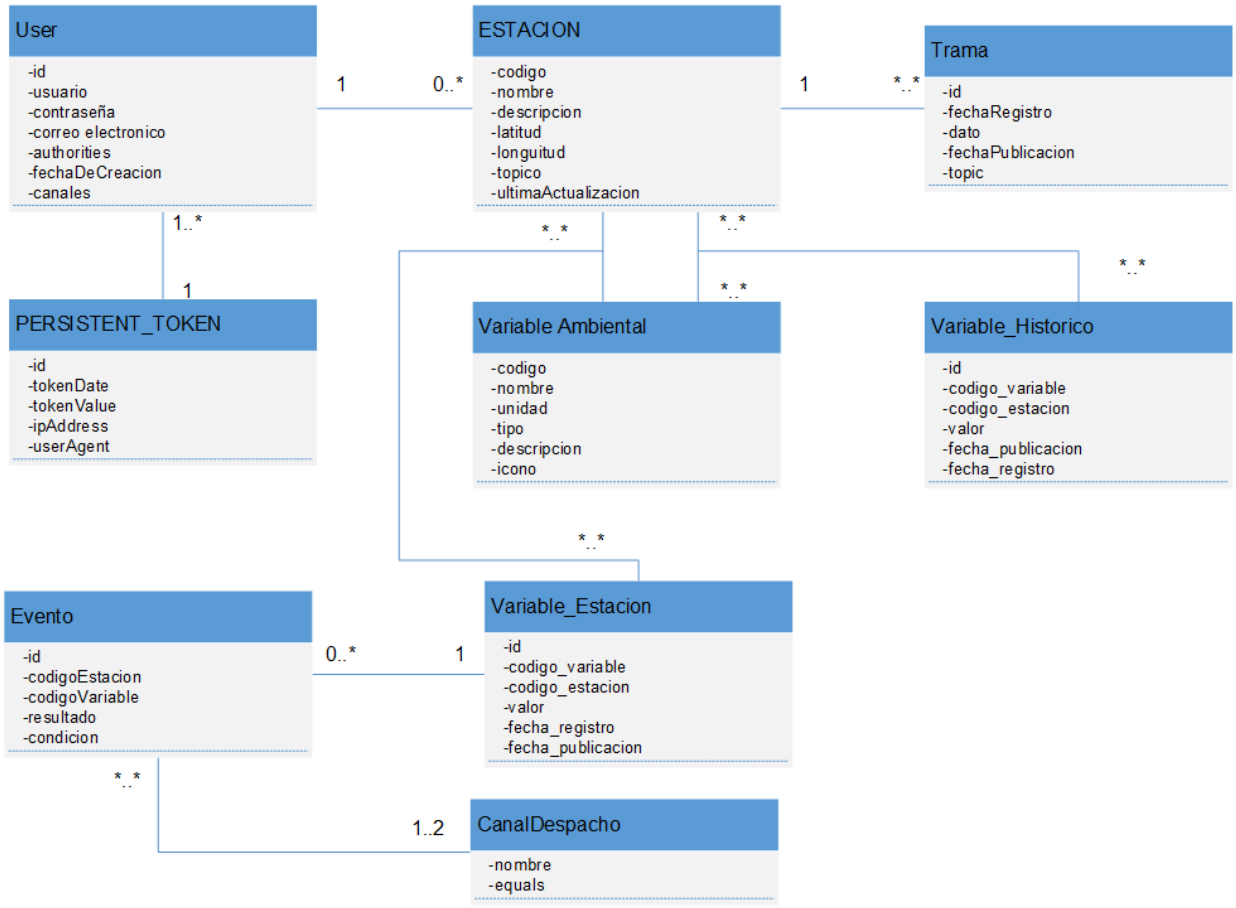


Figura 53: Diagrama de clases – subsistema de visualización de datos
Fuente: Elaborado por los autores.

Esquemas de comunicación con otros sistemas

La plataforma web y el servicio de almacenamiento (SAVA) se comunican con servicios externos; a continuación, se describe esta comunicación.

Sistema de envió de mensajes SMS

Para poder realizar el aviso de una alerta SMS se utiliza un servicio externo. Esto se lo hace enviando en una petición POST en HTTP, en la cual se adjunta un mensaje que contiene el numero celular y el mensaje que se quiere transmitir. El servicio externo que facilita esto es Twilio, una plataforma para el envío de SMS y llamadas telefónicas. La figura 54, ilustra el diagrama de comunicación entre el sistema SAVA y Twillio.

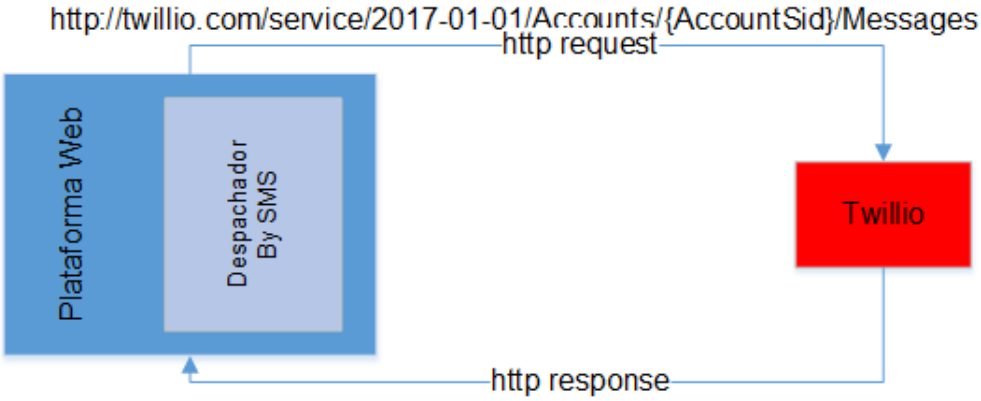


Figura 54: Comunicación SAVA con Twillio.
Fuente: Elaborado por los autores.

Sistema SOS

Todos los mensajes que lleguen al servicio de almacenamiento, deberán ser almacenados bajo el estándar SOS. Para esto se envía una petición HTTP a la implementación de este estándar llamado 52 North, como se puede ver en la figura 55.

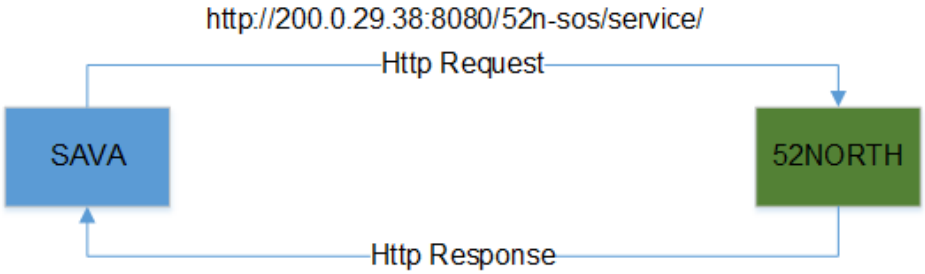


Figura 55: Comunicación SAVA con 52North
Fuente: Elaborado por los autores.

Diagramas de secuencias

En la figura 56, 57 y 58, se describen los principales diagramas de secuencia que se han definido para la solución planteada.

Almacenamiento de variables medioambientales – SAVA.

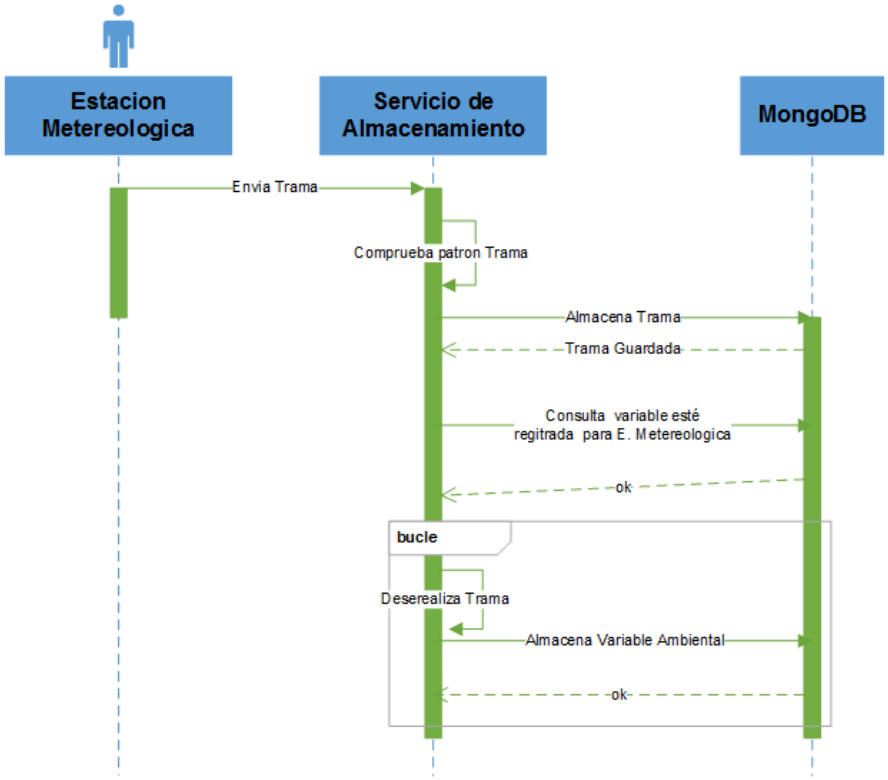


Figura 56: Diagrama de secuencia para SAVA
Fuente: Elaborado por los autores.

Almacenamiento SOS.

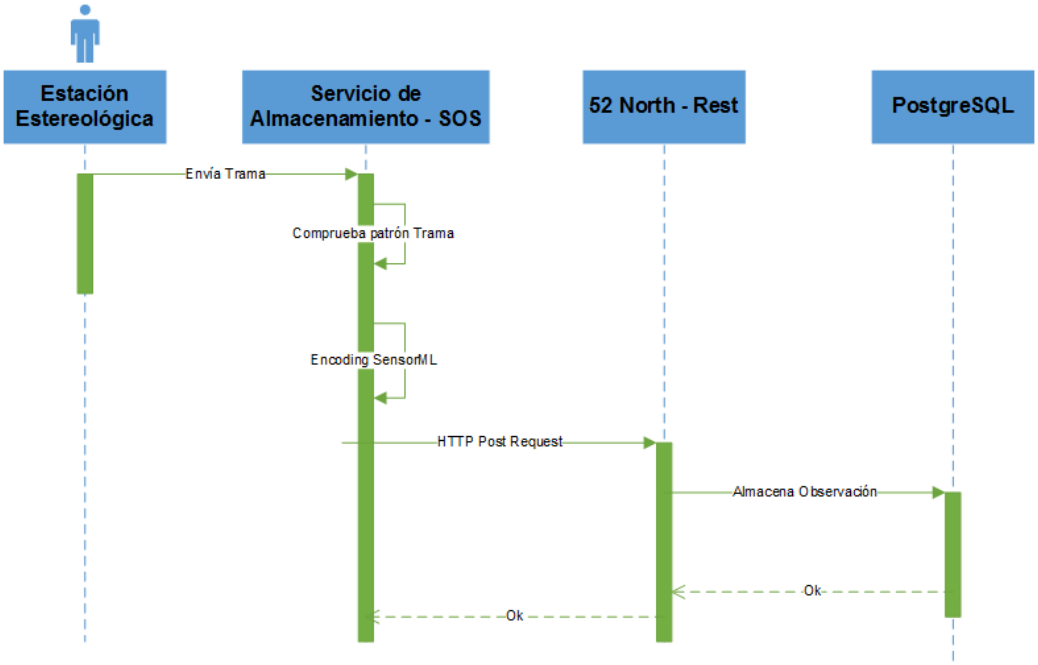


Figura 57: Diagrama de secuencia SOS
Fuente: Elaborado por los autores.

Descarga historial - plataforma web.

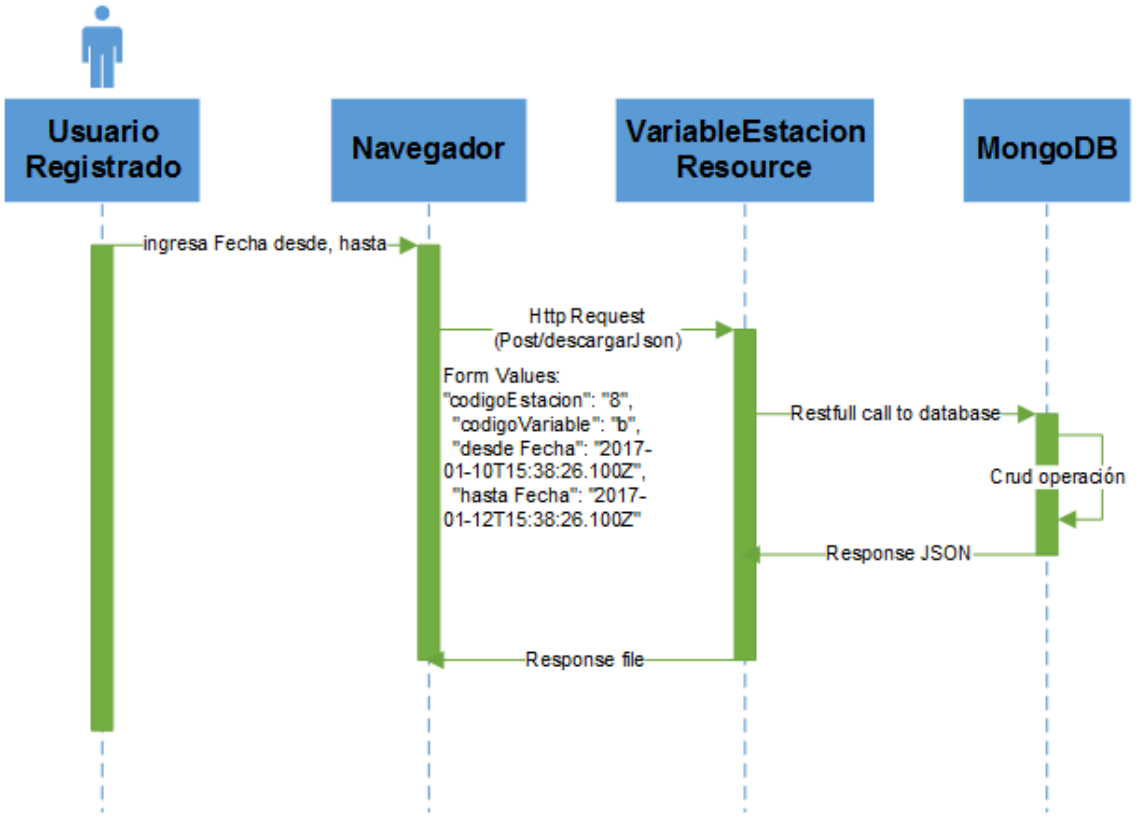


Figura 58: Diagrama de secuencia - descarga de archivos
Fuente: Elaborado por los autores.

6. Vista de desarrollo

La vista de desarrollo ilustra el sistema desde la perspectiva del programador, está enfocado en la administración de los artefactos de software. Esta vista también se conoce como vista de implementación, debido a que describe los componentes de sistema. El diagrama UML que se utiliza en la vista de desarrollo es el Diagrama de Paquetes de la figura 59.

Diagrama de paquetes – módulo de alertas / plataforma web

Figura 59: Diagrama de paquetes – componente de alertas
Fuente: Elaborado por los autores.

7. Vista de proceso

En esta vista se describirán los principales procesos que hay en el sistema y la forma en que se comunican estos procesos, para esta vista se ha incluido el diagrama de actividad de UML.

Diagrama de actividades

En la figura 60, se puede observar el diagrama de actividades para la notificación de eventos. Se debe destacar que en este proceso es necesario implementar el patrón de diseño *chain of responsibility* o "cadena de responsabilidades". Este patrón tiene como enfoque definir una serie de clases, las cuales van a recibir una petición, y cada una de ellas analizará si es su responsabilidad atender la petición o no. Esto se lo implementa cuando se desea evaluar si un evento es por condición o por inactividad. Cuando se lanza la consola de eventos y se desea evaluar el patrón del evento, ésta se propagará a través de la cadena hasta que un objeto manejador se haga responsable de procesarla.

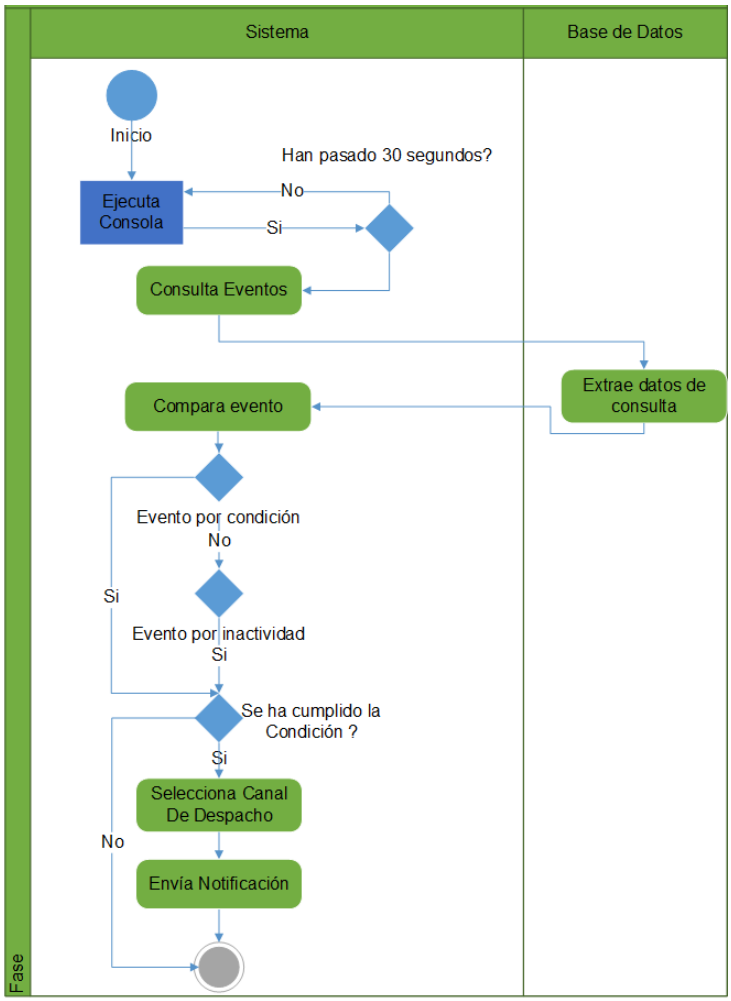


Figura 60: Diagrama de actividad - notificación de alertas.
Fuente: Elaborado por los autores.

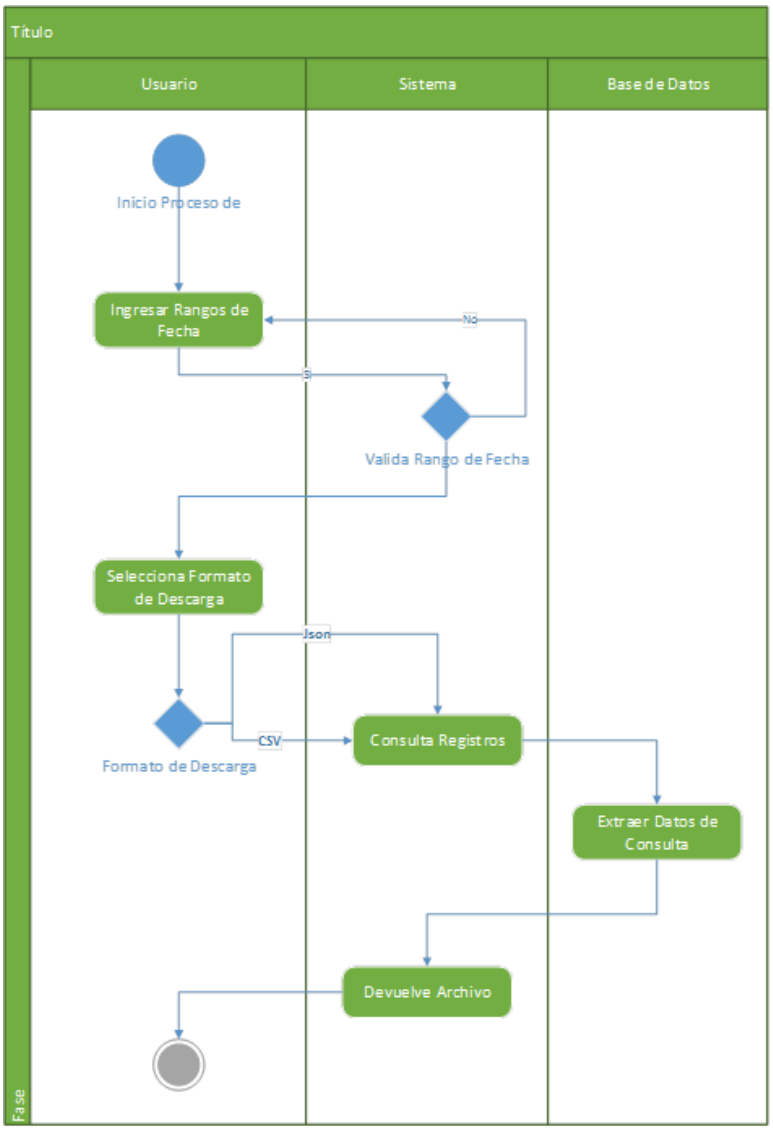


Figura 61: Diagrama de actividad - proceso de descarga
Fuente: Elaborado por los autores.

8. Vista física

Diagrama de Despliegue

Para el despliegue de los componentes de la plataforma web y el sistema de almacenamiento de variables medioambientales se utilizará la plataforma tecnológica que dispone UTPL. Para ello se hará uso del servidor que proveerá las características de la escalabilidad, rendimiento y disponibilidad requeridas por este proyecto. En la figura 62, se esquematiza la distribución de los diferentes recursos hardware que se utilizarán en el ambiente de producción.

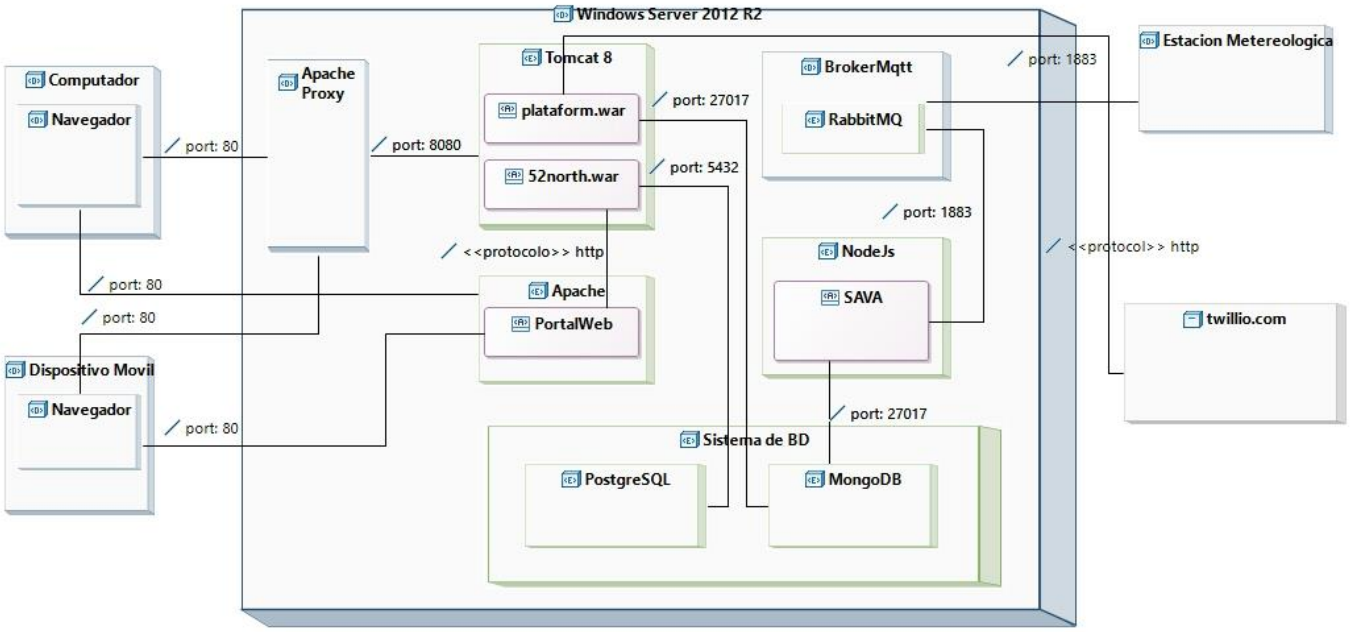


Figura 62: Diagrama de despliegue de la aplicación
Fuente: Elaborada por los autores.

A continuación, la tabla 47 describe las características del equipo físico donde se encuentra alojado el sistema.

Tabla 47: Características del equipo físico.

Características del Equipo	
Sistema Operativo	Windows Server 2012 R2
RAM	8 GB
Procesador	Intel Xeon 2.5 Ghz
Disco Duro	190 GB

Fuente: Elaborada por los autores.

9. Vista lógica de datos

Los diagramas de la figura 63 a la 66, representan los modelos de datos, en los cuales el sistema almacenara la información.

Componente de Administración

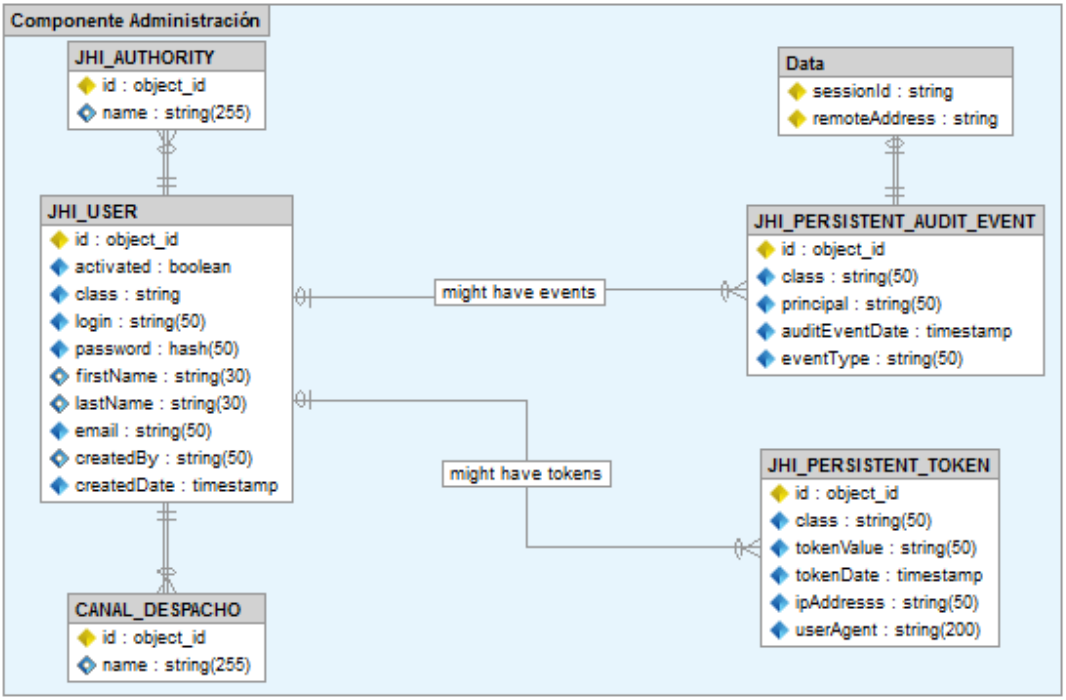


Figura 63: Modelo de datos - componente de administración
Fuente: Elaborado por los autores.

Componente de Gestión

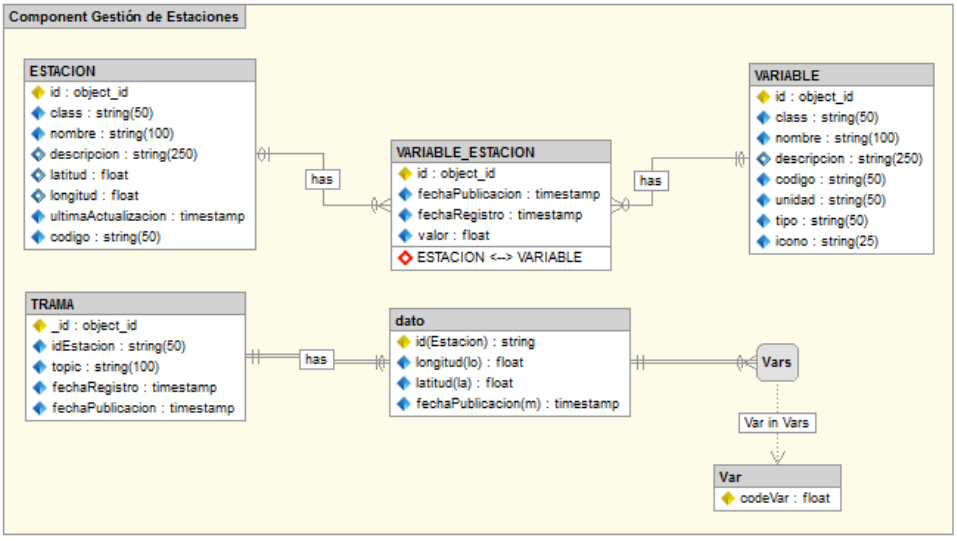


Figura 64: Modelo de datos - componente de gestión
Fuente: Elaborado por los autores.

Componente de notificación

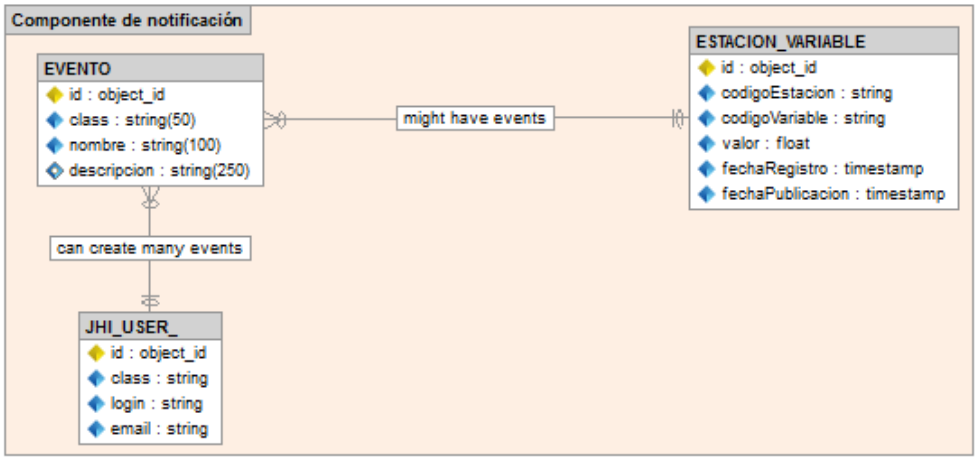


Figura 65: Modelo de datos – componente de notificación
Fuente: Elaborado por los autores.

Componente SOS

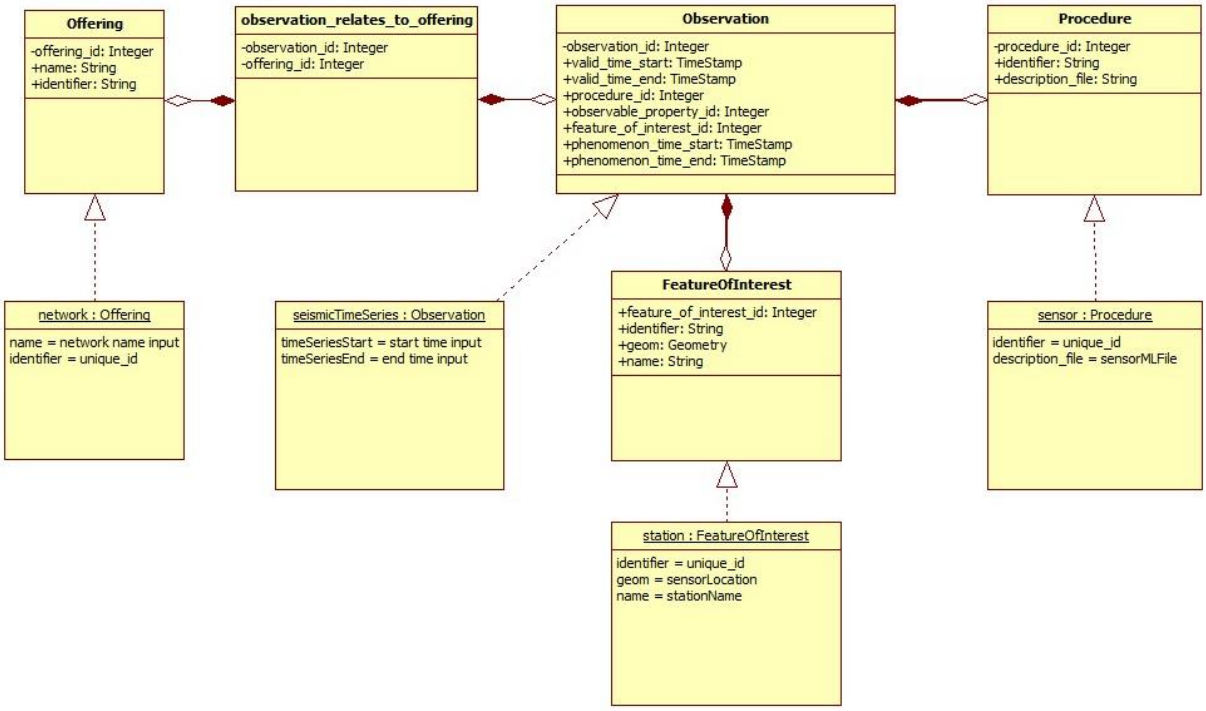


Figura 66: Diagrama de datos - componente SOS
Fuente: Tomado del sitio web 52North (52°North Initiative for Geospatial Open Source Software GmbH, 2016)

10. Tamaño y rendimiento

- La plataforma desarrollada permitirá el acceso concurrente de varios usuarios, de modo que puedan realizar múltiples transacciones.
- El tiempo de respuesta entre las peticiones que se realice no debe ser mayor de 5 segundos.

11. Calidad

- El sistema contará con una ayuda, es decir con la especificación de los diferentes componentes que forman del sistema; lo cual permitirá resolver cualquier interrogante del usuario con respecto a su funcionamiento.
- La interfaz del sistema estará diseñada de manera intuitiva, de manera facilite el uso al usuario que posea conocimientos básicos en informática. Por otro lado, también se tendrá en cuenta el diseño de las pantallas con la característica de *Responsive Design*, es decir tomando en cuenta las dimensiones de las pantallas en los dispositivos móviles.
- El sistema web podrá ejecutarse en diferentes navegadores y dispositivos móviles.
- El sistema estará disponible cada vez que un usuario lo requiera.
- El usuario deberá conocer el resultado de sus acciones, mediante mensajes de confirmación.
- El sistema ha sido desarrollado para que ingresen únicamente los usuarios que estén registrados en el mismo, por medio de usuario y contraseña.
- El sistema deberá ser flexible para que facilite su mantenimiento futuro.
- El sistema deberá ser intuitivo para los usuarios.
- El *front-end* de la aplicación web será adaptable a diferentes idiomas (inglés, español).

ANEXO # 5: Archivo de configuración SOS

Tabla 48: Archivo de configuración SOS

```
{
        "pathSos": "http://200.0.29.38:8080/52n-sos/service/",
        "estaciones": [
                {
                        "code": "8",
                        "configuration": {
                                "offering": "http://www.utpl.edu.ec/estacion/offering/1",
                                "observation": {
                                        "identifier": {
                                          "codespace": "http://www.opengis.net/def/nil/OGC/0/unknown"
                                        },
                                        "type": "http://www.opengis.net/def/observationType/OGC-
OM/2.0/OM_Measurement",
                                        "procedure": "http://www.utpl.edu.ec/estacion/procedure/1",
                                        "featureOfInterest": {
                                          "identifier": {
                                                        "value":
"http://www.utpl.edu.ec/featureOfInterest/8",
                                                        "codespace":
"http://www.opengis.net/def/nil/OGC/0/unknown"
                                          },
                                          "name":
                                                {
                                                  "value": "loja:loja:utpl:8",
                                                  "codespace":
"http://www.opengis.net/def/nil/OGC/0/unknown"
                                                },
                                          "sampledFeature": [

        "http://www.52north.org/test/featureOfInterest/world"
                                          ],
                                          "geometry": {
                                                        "type": "Point",
                                                        "coordinates": [
                                                          -3.987861111111111,
                                                          -79.19679722222223
                                                        ],
                                                        "crs": {
                                                          "type": "name",
                                                          "properties": {
                                                                        "name": "EPSG:4326"
                                                          }
                                                        }
                                          }
                                        }
                                }
                        },
                        "vars": [
                                {
                                        "codePlatform": "a",
                                        "uom": "km/h",
                                        "observedProperty":
"http://mmisw.org/ont/cf/parameter/wind_speed"
                                },
                                {
                                        "codePlatform": "h",
                                        "uom": "Cel",
                                        "observedProperty":
"http://mmisw.org/ont/cf/parameter/air_temperature"
                                },
```

Fuente: Elaborado por los autores.

ANEXO # 6: Manual de usuario

Manual de Usuario para la Plataforma Web de Estaciones Meteorológicas
Versión 1.0

Historial de Revisión

Fecha	Versión	Descripción	Autor
01/04/20016	1.0	Manual de Usuario	Quiñones Cuenca Felipe David Figueroa Sarmiento Byron Rafael

Manual de Usuario
Plataforma Web para el Almacenamiento y Visualización de Variables medioambientales

Al acceder a la plataforma web se presenta la pantalla home del sitio (ver figura 67), en ella se encuentran los siguientes elementos:

- Menú principal
  - Inicio.
  - Estaciones.
  - Variables medioambientales
  - Administración.
- Selección de Idioma: Se puede realizar el cambio de idioma al inglés, este y el idioma español son los únicos idiomas que se pueden cambiar.
- Mensaje de Bienvenida: Describe brevemente la plataforma web.

Figura 67: FrontPage del Subsistema de Gestión de Estaciones Meteorológicas
Fuente: Elaborado por los autores.

DEFINICIONES

Estación meteorológica:

Una estación meteorológica es una terminal que está destinada a medir y registrar regularmente diversas variables medioambientales, las cuales se encuentra ubicados en diferentes puntos del país. Todas las estaciones meteorológicas envían sus datos a través del protocolo MQTT, para que puedan ser recibidas por la Plataforma de Estaciones Meteorológicas.

Variables medioambientales:

Una variable medioambiental es el valor detectado por un sensor que se encuentran en una estación meteorológica. Dicho valor es enviado a la plataforma para su respectivo almacenamiento; por lo que se requiere que las variables medioambientales se registren previamente antes de comenzar la transmisión de datos hacia la plataforma. Únicamente de esa forma, la plataforma estará en la capacidad de interpretar los valores que recibe, y podrá presentar la magnitud con la que se midió un determinado fenómeno.

MQTT:

Es un protocolo de conectividad bidireccional (two-way) para redes inalámbricas orientado a paradigmas como "máquina a máquina" (M2M) o "Internet de las cosas" (IoT). Se trata de un protocolo de mensajería ligero que se encuentra situado en la capa de aplicación, y utiliza como base el protocolo TCP en la capa de transporte. MQTT sigue un modelo publicación/suscripción, donde hay un nodo que publica mensajes y otros nodos interesados que se "suscriben" al primer nodo para poder recibir estos mensajes. MQTT tiene como objetivos minimizar el uso del ancho de banda de red, optimizar los requerimientos de recursos del dispositivo; y asegurar un cierto nivel de fiabilidad. MQTT está destinado al mundo de los dispositivos interconectados y aplicaciones móviles; donde el ancho de banda y la energía de la batería son un bien escaso.

Alerta:

Una alerta es un aviso que el usuario establece para que se le notifique sobre una condición que ha establecido; existen dos tipos de alertas:

- *Condicional:* Este tipo de alerta será útil cuando se pretenda establecer condiciones que la variable medioambiental deba cumplir. Por ejemplo: en el caso que se requiera una notificación cuando supere o esté bajo un valor determinado.
- *Inactividad:* Este tipo de alerta sirve para avisar si se ha dejado de recibir datos provenientes de una variable medioambiental ubicada en una estación meteorológica, aquí se debe establecer el periodo de inactividad en el cual la variable medioambiental no ha emitido datos.

Las alertas se pueden recibir a través de correo electrónico y SMS.

Nota: para recibir alertas a través de SMS es necesario solicitar al administrador de la plataforma que se habilite esta opción.

ESTACIONES METEOROLÓGICAS

En esta sección se podrá encontrar las estaciones meteorológicas que has sido establecidas, las cual pueden ser visualizadas por todos quienes ingresen a la plataforma, en este menú se pueden realizar las siguientes acciones:

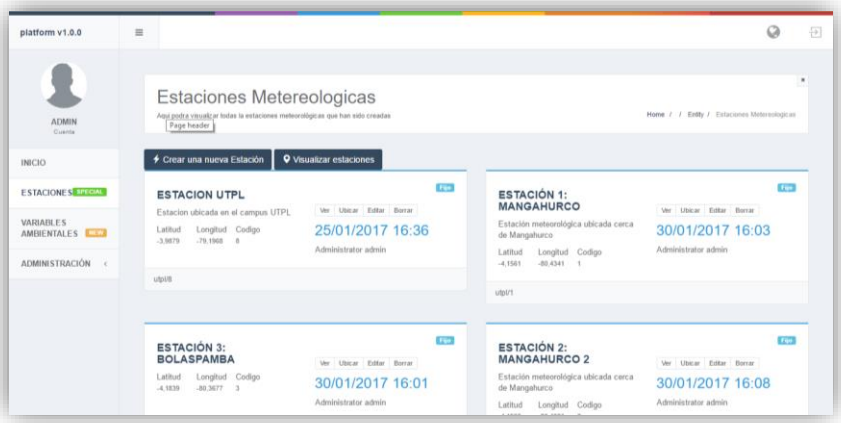


Figura 68: Menú estaciones meteorológicas
Fuente: Elaborado por los autores.

- **Crear Nueva Estación**

Cuando sea necesario agregar una nueva estación a la plataforma web, se deberá hacer clic sobre este botón, del cual emergerá una nueva ventana.

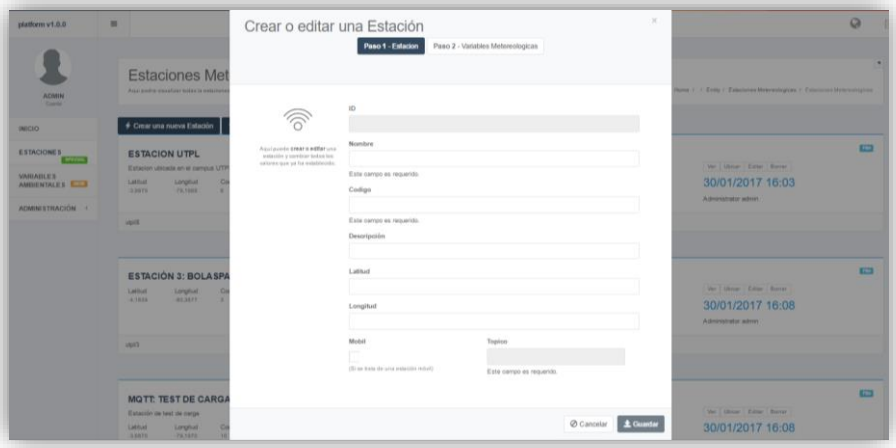


Figura 69: Ventana emergente - crear estación meteorológica
Fuente: Elaborado por los autores.

Cuando emerja esta ventana como primer paso deberá completar los siguientes campos.

- Nombre. (obligatorio)
- Código. (obligatorio)
- Descripción.
- Latitud
- Longitud.
- Móvil.

Como segundo paso para completar la creación de la estación meteorológica se debe definir las variables medioambientales que se encuentran asociadas a la estación. Para lo cual, el sistema presenta un listado de variables medioambientales que han sido creadas previamente, luego el usuario debe seleccionar una o varias variables que desee asociar a la estación creada; como se muestra en la siguiente figura.

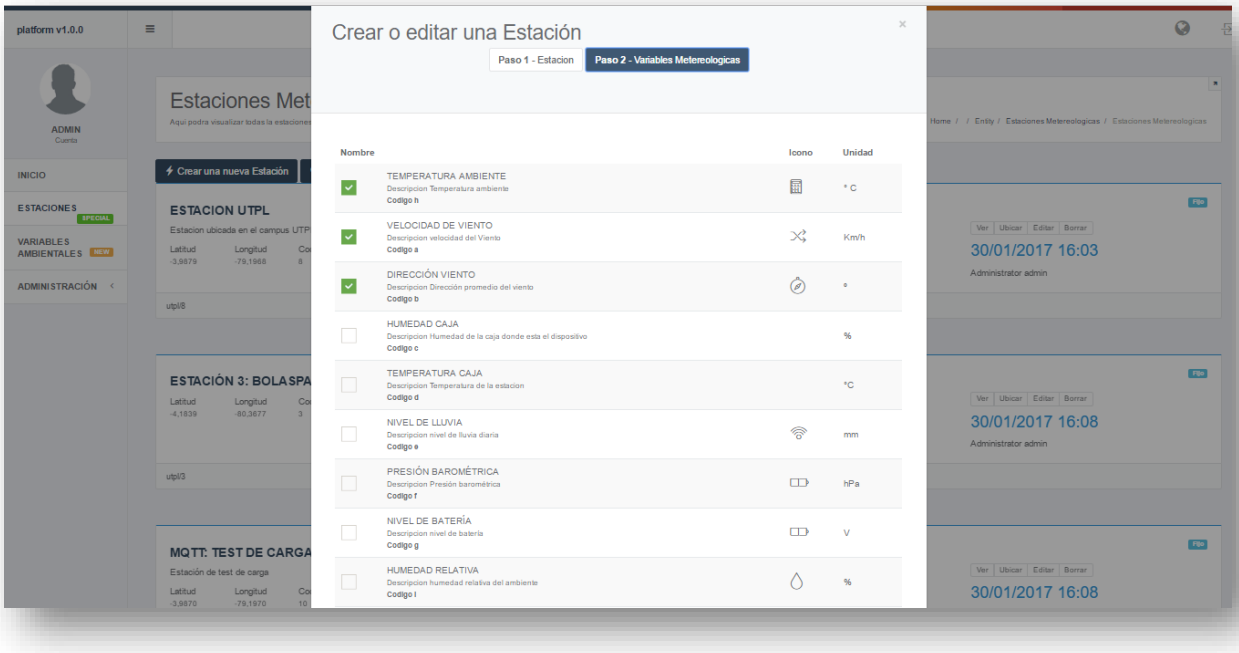


Figura 70: Ventana emergente - editar estaciones meteorológicas
Fuente: Elaborado por los autores.

Finalmente, luego de completar estos pasos, el usuario deberá hacer clic sobre el botón ***Guardar***, posteriormente podrá observarla entre las estaciones que han sido creadas.

- **Ver Detalle de Estación Meteorológica**

Con esta opción se puede visualizar el detalle de la estación meteorológica. La información desplegada a continuación corresponde al último valor que ha sido almacenado por cada variable medioambiental que una estación tenga asociada.

Figura 71: Detalle de estación meteorológica
Fuente: Elaborado por los autores.

Adicionalmente, en la opción correspondiente a ***Detalle de la Estación*** se tiene tres operaciones que se pueden realizar sobre cada una de las variables medioambientales asociadas a la estación meteorológica seleccionada: ***Ver***, ***Descargar***, y ***Alerta***.

Figura 72: Operaciones sobre una variable medioambiental seleccionada
Fuente: Elaborado por los autores.

Operaciones sobre Variables medioambientales

- **Ver:** Al hacer clic sobre este botón ingresará a una nueva página donde podrá ver el historial de la variable medioambiental de la cual se seleccionó.

Figura 73: Historial de la variable medioambiental seleccionada
Fuente: Elaborado por los autores.

- **Descarga:** Esta acción abrirá una nueva página, aquí se puede descargar la información de la variable medioambiental estableciendo un rango de fechas. Una vez que se haya hecho esto, el usuario deberá hacer clic sobre el botón *Descargar*, esto hará que se presente una pantalla donde se debe seleccionar el formato de archivo que se desee descargar. Las opciones que actualmente el sistema ofrece son Csvy Json.

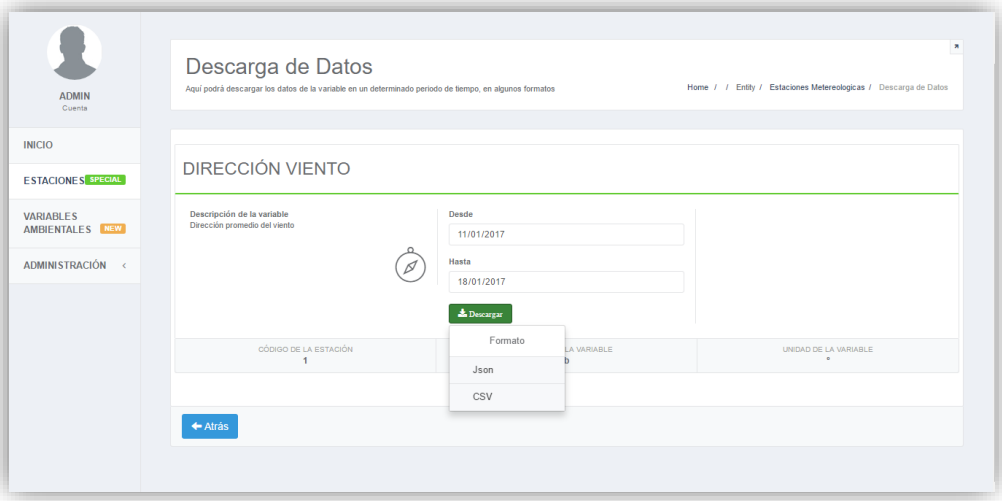


Figura 74: Descarga de datos- historial de variable medioambiental
Fuente: Elaborado por los autores.

- **Alerta:** Al hacer clic sobre este botón, se presentará una nueva pantalla, en la cual lo primero que se puede observar es la lista de alertas que han sido establecidas para la variable medioambiental.

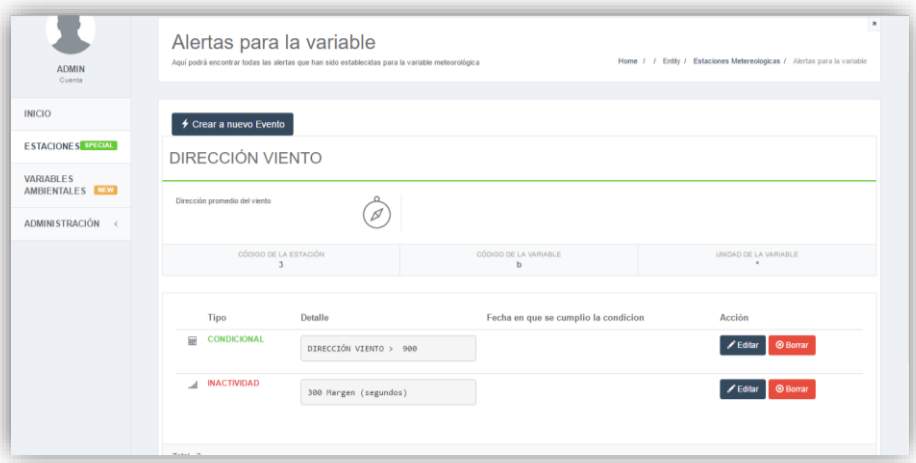


Figura 75: Menú configuración de alertas
Fuente: Elaborado por los autores.

Adicionalmente en esta sección se tiene la opción ***Crear nuevo evento***, al hacer clic sobre el botón, se presentará un formulario en el cual se debe seleccionar el tipo de alerta que se quiere establecer.

En el caso de que se seleccione la opción ***Alerta por Inactividad***, se presentara un nuevo campo denominado ***Tiempo de Inactividad***. Esto corresponde al periodo de tiempo en segundos en la cual cierta variable no emita datos; una vez que se haya completado este campo se tiene que hacer clic sobre el botón ***Guardar***. A continuación, se podrá observar nueva alerta incluida en la lista de alertas.

Figura 76: Menú crear o editar una alerta – tipo inactividad
Fuente: Elaborado por los autores.

Por otro lado, en el caso que se desee configurar una alerta tipo ***Condicional***, el sistema desplegará un formulario con dos campos relevantes. En el campo *Condición* se deberá seleccionar la condición que deseas establecer las cuales pueden ser: mayor, menor, igual, mayor igual que, y menor igual que. Mientras que el campo *Resultado* corresponde al valor con el que se compara el ultimo valor que emita la estación meteorológica para variable medioambiental seleccionada. Una vez que se haya completado esos campos se deberá hacer clic sobre el botón ***Guardar.*** A continuación, se podrá observar nueva alerta incluida en la lista de alertas.

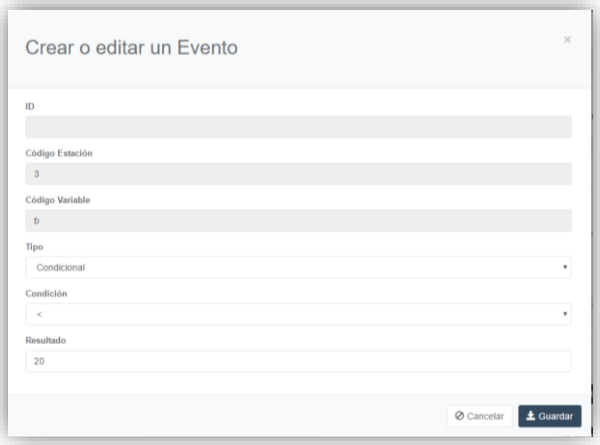


Figura 77: Menú crear o editar una alerta – tipo condicional
Fuente: Elaborado por los autores.

- **Ubicar Estación Meteorológica**

En esta página presentara la ubicación de la estación meteorológica que se seleccionó.

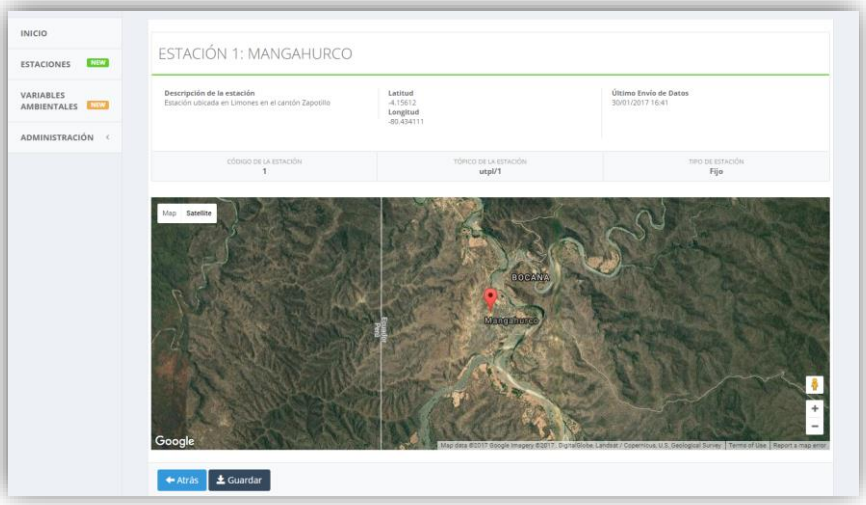


Figura 78: Ubicación geográfica de la estación meteorológica seleccionada

Fuente: Elaborado por los autores.

- **Editar Estación Meteorológica**

Aquí se puede editar la información que se estableció previamente cuando se creó la estación.

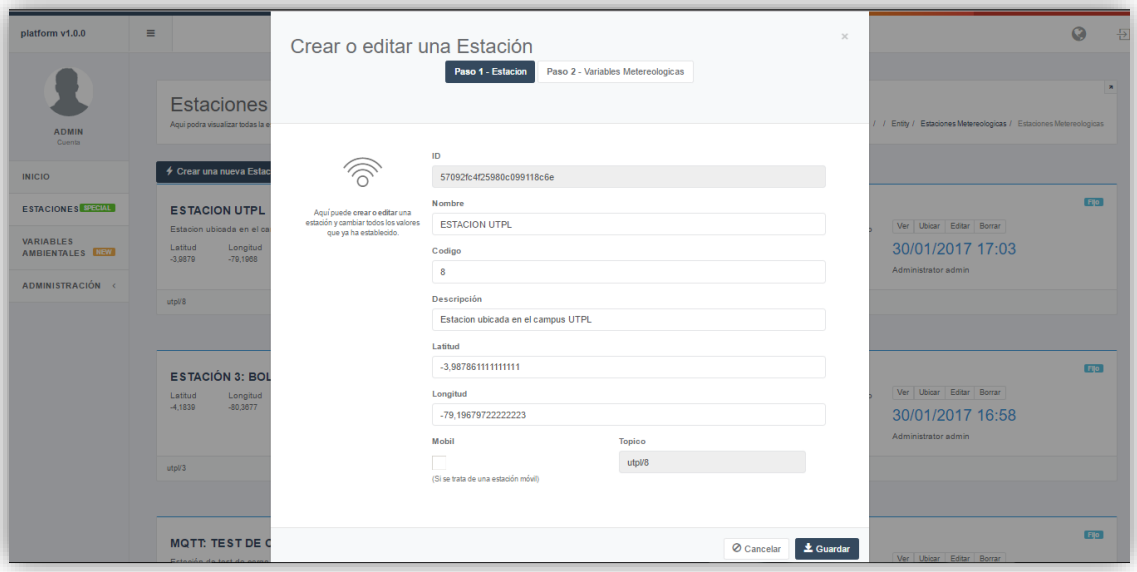


Figura 79: Editar una estación meteorológica
Fuente: Elaborado por los autores.

- **Borrar Estación Meteorológica**

Cuando se da clic sobre este botón aparece un mensaje de confirmación para eliminar la estación meteorológica.

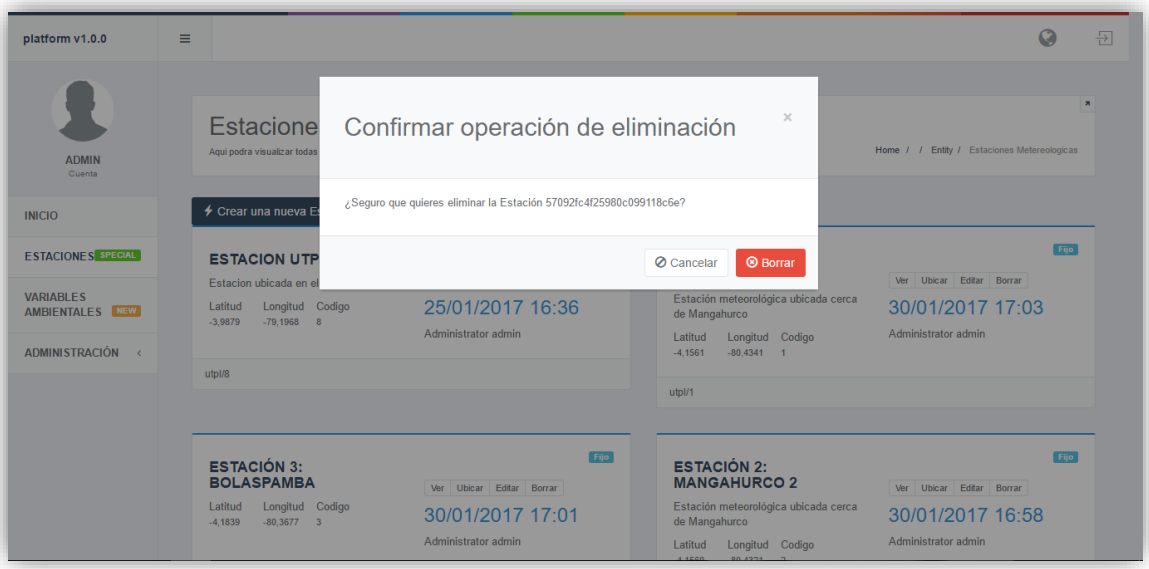


Figura 80: Eliminar estación meteorológica
Fuente: Elaborado por los autores.

- **Visualizar Mapa de Estaciones**

Haciendo clic sobre este botón podrá ver donde se encuentran ubicadas las estaciones meteorológicas previamente creadas, como se puede ver en la siguiente imagen.

Figura 81: Mapa de estaciones meteorológicas
Fuente: Elaborado por los autores.

VARIABLES MEDIOAMBIENTALES

En esta sección se puede definir una nueva variable medioambiental, así como se puede visualizar una lista de variables medioambientales que han sido creadas.

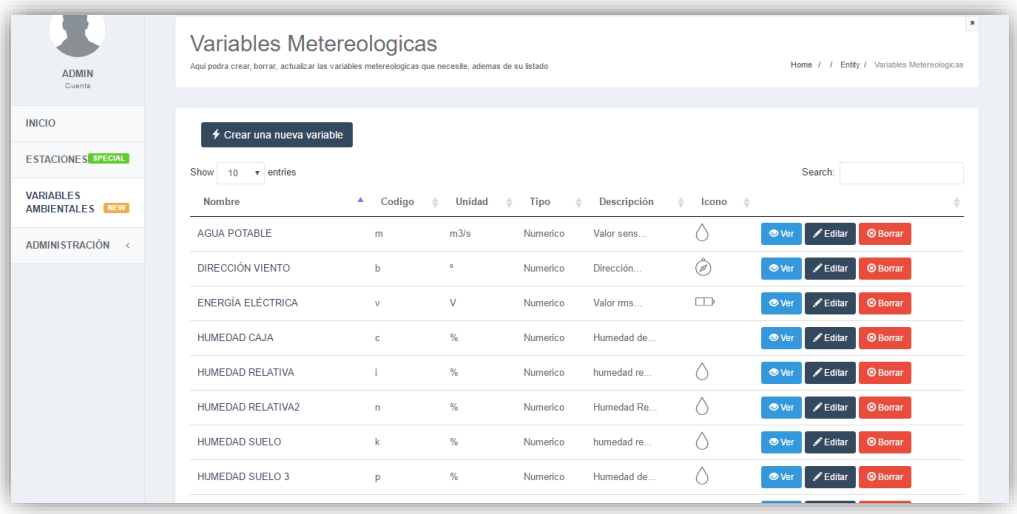


Figura 82: Lista de variables medioambientales asociadas a determinada estación meteorológica
Fuente: Elaborado por los autores.

Si es que fuera el caso y todavía la variable medioambiental no ha sido definida aún se deberá crear una, para lo que se debe hacer clic sobre el botón de la parte superior que se llama **Crear una nueva variable,** al realizar esta acción presentara una nueva pantalla, la cual se debe llenar los campos que se presentan, solo el campo descripción e icono no es obligatorio.

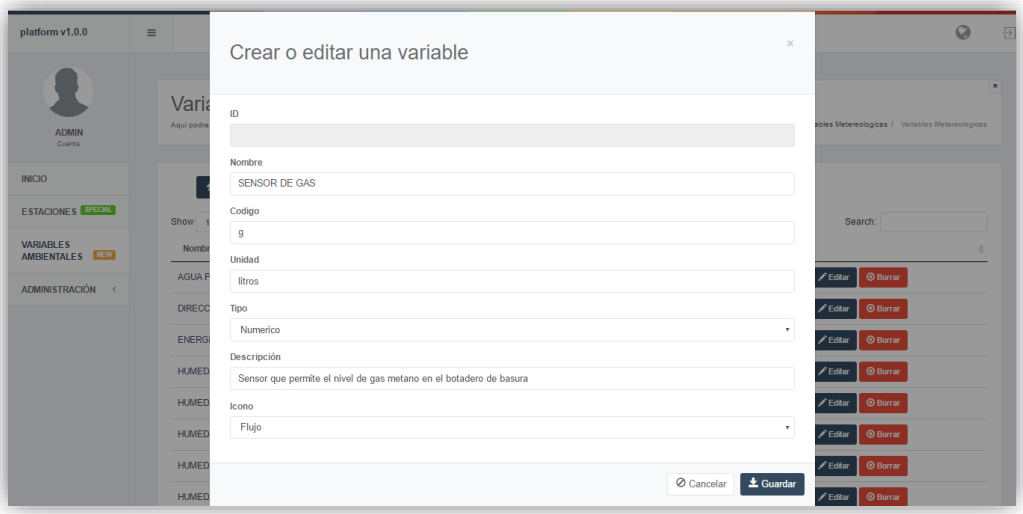


Figura 83: Menú crear o editar variable medioambiental
Fuente: Elaborado por los autores.

ENVIÓ DE DATOS

Para que el envío de mensajes se realice correctamente, es necesario tener en cuenta las siguientes configuraciones:

Tabla 49: Configuración para envío de datos

Dentro del campus universitario	Fuera del campus universitario
Protocolo: MQTT	Protocolo: MQTT
Puerto: 1883	Puerto: 1883
IP: 200.0.29.38.	IP: 172.16.60.29

* En el caso de requerir establecer comunicación dentro del campus universitario UTPL es necesario solicitar permisos al administrador de la red al puerto 1883, específicamente si se está utilizando una conexión inalámbrica.
Fuente: Elaborado por los autores.

Al ser MQTT el protocolo a usar, este protocolo maneja el concepto de ***Tópico***. Un tópico este es una cadena que se usa para filtrar mensajes, el tópico que se debe utilizar está definido de la siguiente manera:

`plataforma/sensor/utpl/"código de la estación"`

Donde "`código de la estación`", es el código que se le asignó a la estación en el momento de haberla creado. En el caso de no recordar este valor, se lo puede consultar ingresando en la sección de estaciones que se encuentra en el panel izquierdo de la plataforma web, como se puede ver en la siguiente imagen.

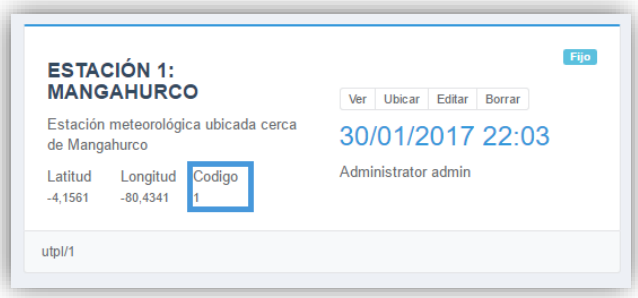


Figura 84: Estación meteorológica
Fuente: Elaborado por los autores.

Por otro lado, otro aspecto relevante para el envío de mensajes es dar formato al mensaje que se enviara a la plataforma, el mismo que tiene una estructura JSON y está conformado por cinco valores:

Tabla 50: Estructura de mensaje Json

Parámetro		Descripción	Obligatorio
1	id	Corresponde al ***código de la estación*** de la cual se trasmite el mensaje.	Si
2	m	Se refiere a la ***fecha*** en la que se registró el dato en la estación meteorológica. La fecha es transmitida en formato UTF (2016/10/09 12:23)	Si
3	la	Este campo se refiere a la ***latitud*** en la que se encuentra la estación meteorológica. La latitud debe estar bajo el formato EPSG:4326	No
4	lo	Se refiere a la ***longitud*** en la que se encuentra la estación meteorológica. La longitud debe estar bajo el formato EPSG:4326	No
5	var	Es una colección de valores en la cual se encuentran los valores de las variables medioambientales que han sido censadas.	Si

* El tercer y cuarto parámetro no son obligatorios, por lo que se puede obviar si el tipo de estación no es móvil,
Fuente: Elaborado por los autores.

La siguiente figura ilustra el formato del mensaje a transmitir.

Figura 85: Formato de mensaje
Fuente: Elaborado por los autores.