Universidad Andrés Bello

UNIVERSIDAD ANDRES BELLO

FACULTAD DE INGENIERÍA

INGENIERÍA EN COMPUTACIÓN E INFORMÁTICA

“PROTOTIPO DE OBSERVATORIO DE EVENTOS HIDROMETEREOLÓGICOS ROBUSTO A FALLAS EN LOS SENSORES PARA LA ZONA DE ATACAMA”

Trabajo de título para optar a título de Ingeniería en Computación e Informática

Autor:

Álvaro Cabrera Montino

Profesor Guía:

Romina Torres

Agradecimiento

El proceso de aprender nunca es cómodo ni fácil, el esfuerzo necesario para llegar a la meta puede ser bastante difícil, pero no menos satisfactorio. Es por eso que quisiera agradecer primero a mi esposa, mi compañera de vida, que me ha dado su apoyo incondicional durante este proceso, sin ella todo esto no hubiera sido posible. Segundo a mi profesora guía que me ayudo a abrir la mente al mundo de la investigación y al verdadero profesionalismo. Por último, a mis mascotas, en especial a mi pequeña Padme que se nos fue, ellos siempre estaban dispuestos a regalar un momento de alegría cuando el estrés supera los ánimos.

Gracias a todos, por su apoyo, por presionarme para dar lo mejor de mí, gracias a ustedes me siento satisfecho por este logro.

Índice

Tabla de contenido

Índice de figuras

Índice de tablas

Resumen

Debido a las condiciones meteorológicas y geográficas de Chile, el País es susceptible a eventos catastróficos como sismos o desastres ocurridos a partir de eventos meteorológicos. Si bien estos eventos, en la mayoría de los casos, no se pueden predecir es posible obtener datos de comportamiento de diferentes fuentes que permiten tomar decisiones que permitan disminuir los daños tanto humanos como materiales ante este tipo de evento.

Este documento busca afrontar una problemática específica para la III región de Atacama, Chile, donde ocurrió un evento hidrometeorológico que causó vastos daños en las cercanías de los cauces de los ríos en la región, debido a desbordes y consiguientes aludes debido a la magnitud del evento. Uno de los problemas a los que se afrontó el equipo de la Oficina Nacional de Emergencias del Ministerio del interior ONEMI, fue que no contaron con la información que les permitiera a las autoridades tomar decisiones de acción oportunas, con el objetivo de disminuir las perdidas debido a este evento.

Esta tesis busca proponer una solución a esta problemática construyendo una herramienta tecnológica que permita entre otras cosas entregar información necesaria para las autoridades de manera que estos puedan tomar decisiones ante un evento similar además de generar los informes actualmente utilizados por la ONEMI Atacama

Abstract

Due the geographic and meteorological conditions in Chile, the country is susceptible to catastrophic events like earthquakes or disasters from meteorological events. While this kind of events, in most of the cases, can't be predicted it is feasible to obtain behavioral data from different sources that allow to take decisions to prevent or decrease structural damage or human loss.

This thesis faces a specific problem on the III Region de Atacama in Chile, where on 2015 a hydro-meteorological event provoked vast damage on populated zones near the rivers in the region due river overflows and avalanches caused by the magnitude of the event. One of the issues that the National Office of Emergencies of the Interior Ministry (ONEMI by its Spanish acronym) has to deal was the fact that they didn't have an up-to-date information to allow to the authorities to take prompt decisions to decrease the damage from the event.

The objective of this project is to propose a solution for this problem is to build a ICT tool to act as a hub for the data sourced from different monitoring stations to show real time data to offer a tool to the authorities to make decisions based on the results of the data collected.

CAPÍTULO 1: Introducción

Introducción

Chile, debido a sus características geográficas, está propenso a sufrir diferentes tipos de desastres naturales en diferentes partes del territorio nacional por la diversidad de atributos que este posee. En particular sus formaciones tectónicas únicas hacen que Chile sea uno de los países con más actividad sísmica del mundo. Además de las características sismográficas de Chile, el país posee numerosas fuentes de agua como ríos y lagos. En especial los ríos del norte de Chile pueden ser afectados por eventos meteorológicos que, debido a diferentes causas, como desviaciones de cauces e intervención humana, eventualmente pueden causar crecidas, desbordamientos y eventualmente aluviones.

En marzo 2015, ocurrió el mayor evento hidrometeorológico en el norte del país en más de 75 años. Según estadísticas provenientes de la Oficina Nacional de Emergencia del Ministerio del interior, en adelante ONEMI, el evento provocó 18 aluviones que generaron 31 muertos, 16 desaparecidos, 35.086 damnificados, 2.071 viviendas destruidas y 6.253 con daño mayor. Los daños causados en este evento, en parte, fueron causados por la falta de información oportuna que permitiera a las autoridades tomar decisiones como medidas de prevención ante catástrofes o situaciones de emergencia.

Aunque actualmente la ONEMI cuenta con estaciones de monitoreo en la región de Atacama, que entregan datos sobre precipitaciones, temperatura y caudal de los ríos de la región y se cuenta con la tecnología para obtener datos sobre el comportamiento de los cauces. Estos datos deben ser recopilados por el personal del Comité Científico Técnico, en adelante CCT, de la ONEMI para después elaborar un informe sobre lo ocurrido.

A pesar de que se cuentan con los datos y posteriores informes de lo ocurrido, la ONEMI no cuenta con tecnología con la que puedan visualizar datos en tiempo real ni monitorear el estado de los sensores actuales, lo que no les permite actuar de manera oportuna antes situaciones como la antes mencionada.

Debido a estos antecedentes se propone un sistema que permita:

1. Mostrar los datos de las diferentes estaciones de forma centralizada
2. Generar informes bajo demanda que permita automatizar el proceso de recopilación de datos
3. Mantener monitoreadas las estaciones de manera de detectar fallas el equipamiento y además proveer los medios de tolerancia a fallos para el sistema actual.

La siguiente tesis se divide en los siguientes capítulos:

Capítulo 1 Introducción: Este capítulo se da explicación al contexto de origen para esta tesis

Capítulo 2 Fundamentación: Identifica el contexto y la identificación de la problemática completar el desarrollo de esta tesis. Además, se identifican todos los elementos que conforman parte de la fundamentación para llevar a cabo el proyecto.

Capítulo 3 Materiales y métodos: Prepara e identifica la metodología de producto y desarrollo. Define los materiales que se utilizaran como lenguajes de programación, control de versiones y cambios, gestión de riesgo y diseño preliminar de la aplicación.

Capítulo 4 Resultado y discusión: Resumen del desarrollo de la aplicación, esto incluye la planificación, liberación de incrementos, pruebas y resultado del incremento.

Capítulo 5 Conclusiones: Port mortem del proyecto y conclusiones relevantes

CAPITULO 2: Fundamentación

2.1 Contexto

La ONEMI en conjunto con la DGA[1] en la región de Atacama, mantiene un sistema de monitoreo sobre las fuentes de aguas de la región, que incluyen estaciones de monitoreo que han implementado sensores que permiten obtener datos de caudal, humedad y temperatura de la zona en la que se encuentra.

Los datos entregados por las diferentes estaciones no son entregados de manera estandarizada debido a posibles diferencias en el funcionamiento de las estaciones y/o sensores específicos. Por esta razón los miembros del CCT[2] encargados de elaborar los informes de comportamiento de las fuentes de agua deben recolectar los datos de las estaciones en ocasiones de manera manual.

Una vez obtenidos los datos se elabora un informe donde, además, los miembros del CCT deben realizar el cálculo manual para los datos que se representan en el informe entregado.

[1] Dirección General de aguas
[2] Comité Científico Técnico

Tabla Datos Hidrometeorológicos Cuenca Río Copiapó

ESTACIONES SATELITALES O GPRS		Q INST (m3/seg)	T (C°)	H° (%)	PRECIPITACIONES (mm) ENE	FEB	MARZ	ABR 19	ABR 20	TOTAL ACUM FECHA	MISMA FECHA 2015
N°	CUENCA HIDROGRÁFICA RÍO COPIAPÓ										
1	Río Pulido en vertedero	1,051	12,44	81,0	-	-	-	0,00	0,00	0,00	-
2	Río Copiapó en Pastillo	1,53	-	-	0,00	0,00	0,00	s/t	s/t	0,00	65,00
3	Río Copiapó en Lautaro	0,329	-	-	-	-	-	-	-	-	70,00
4	Río Copiapó en la Puerta	s/t	s/t	s/t	s/t	s/t	s/t	s/t	s/t	s/t	s/t
5	Qda. Paipote en Pastos Grandes	-	8,1	73,0	0,00	0,00	0,00	0,00	0,00	0,00	86,10
6	MET-QP1 (Cuesta Codoceo)	-	8,0	-	0,00	0,00	0,00	0,00	0,00	0,00	-
7	MET-QP2 (El Chulo)	-	23,0	-	0,00	0,00	0,00	0,00	0,00	0,00	-
8	Copiapó en La Ciudad	-	13,9	83,7	0,00	0,00	0,00	0,00	0,00	0,00	23,50

Tipología estaciones monitoreo:
Fluviométrica
Meteorológica

s/t = sin transmisión

Figura 1: ejemplo de informe generado por el CCT

2.2 Definición del problema

Según los estudios realizados por el CCT de la ONEMI, debido a que no se contaba con información en el momento oportuno durante el evento hidrometeorológico ocurrido en marzo 2015 en la región de Atacama produjo considerables pérdidas a nivel humano y estructural.

La problemática que tratará este proyecto es el tiempo que demora la ONEMI en detectar una posible catástrofe asociado a la zona de los ríos de la región de atacama

A continuación, se realiza el análisis de las causas que ocasionan esta problemática utilizando la técnica de 5 por qué.

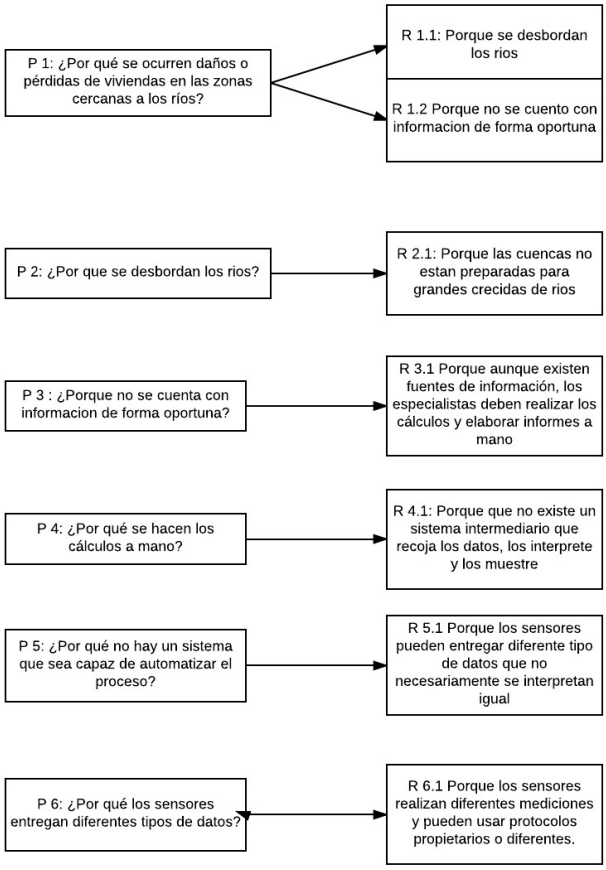


Figura 2 : 5 por que

2.3 Análisis de causas

Basado en este análisis de las causas basado en los 5 por qué. A continuación, se profundiza en las causas utilizando un diagrama de Ishikawa.

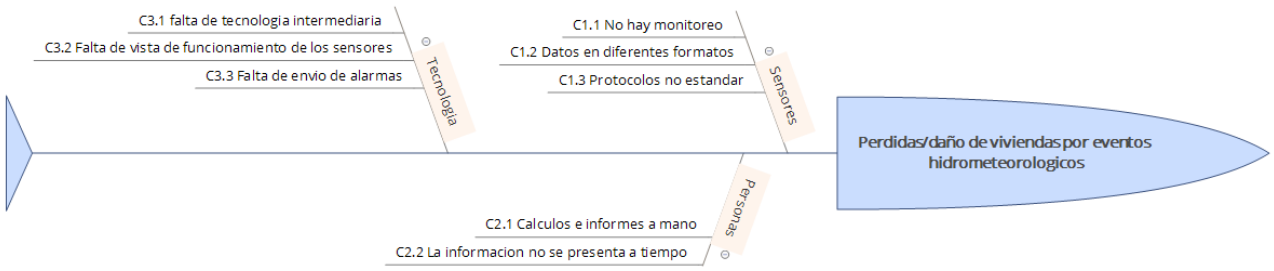


Figura 3 : Diagrama de ishikawa

Según el análisis realizado, se identifican tres causas principales que corresponde a como se están tomando los datos de las diferentes estaciones de monitoreo, la primera causa identificada es que a pesar de que existen estaciones de monitoreo en cada cauce de rio estos no entregan los datos de forma estándar, esto debido a que, los sensores entregan los datos en diferentes formatos o que los protocolos utilizados no son estándar.

Las siguientes causas están relacionadas a como se generan los reportes ya que una de las causas de la problemática identificada tiene relación a que los operadores del CCT deben recoger los datos manualmente y después generar reportes basados en cálculos que el mismo equipo realiza. Esto lleva a que la confección de los reportes demora más tiempo del que debieran con la consecuencia de que no se tiene suficiente tiempo para la toma de decisiones ante algún evento catastrófico.

Resumen de Causas identificadas

C1- Los sensores no entregan datos de forma estándar

C1.1 Debido a falta de monitoreo

C1.2 Debido a entrega de datos en diferentes formatos

C1.3 Debido a protocolos de comunicación no estándar

C2 – Las personas no reaccionar a tiempo

C2.1 Debido a que los cálculos deben realizar a mano

C2.2 Debido a que la información no se entrega a tiempo

C3 – No se posee la tecnología para automatizar el proceso

C3.1 Debido a falta de software intermediario que recoja e interprete los datos

C3.2 Falta de una vista que muestre el estado y datos de los sensores

C3.3 Falta de envío de alarmas/alertas.

2.4 Objetivos

A continuación, se definen los objetivos del proyecto.

2.4.1 Objetivo general

Disminuir el tiempo en que la ONEMI demora en detectar una catástrofe asociada a los ríos de la región de Atacama (Disminuir el tiempo en un 50% en que las autoridades puedan tener información suficiente que les permitan tomar decisiones de acción.)

Donde los posibles eventos podrían ser:

- Desbordamientos de ríos de la región

-Datos erróneos debido a fallas en los sensores.

2.4.2 Objetivos específicos

- **OE1:** Aumentar en un 80% la velocidad de elaboración de informes parte de las autoridades al automatizar el cálculo y presentación de informes para los datos entregados por las estaciones de monitoreo
- **OE2:** Aumentar los tiempos de respuesta en un 60% de parte del equipo técnico al monitorear el estado y entrega de datos de las estaciones de monitoreo y sensores individuales instalados en ellas
- **OE3:** Disminuir la probabilidad de que los datos utilizados para la toma de decisiones estén erróneos

Situación actual	Objetivos específicos	Situación esperada	Métrica	Criterio de éxito
No existe monitoreo de los sensores en las diferentes estaciones	Monitorear el estado y la entrega de datos de los sensores	Obtener datos de estado de los sensores y que estos sean precisos	Obtener datos de estado de los sensores al menos del 90% de su tiempo de uptime	Obtener estadísticas de uptime hasta con 5% de error en las transmisiones
No existe automatización en el cálculo y elaboración de informes	Automatizar el cálculo y muestra de información de manera que esta sea en tiempo real	Visualizar el estado de los cauces de los ríos en tiempo real	Obtener información en tiempo real del estado de los cauces de los ríos con un margen de +-5% de error	Información con un margen de error del 5%.
No existe una plataforma que permita mostrar el estado completo de las estaciones y agregar nuevas si es necesario	Disminuir la probabilidad de que los datos utilizados para la toma de decisiones estén erróneos	Visualizar el estado de las estaciones de monitoreo y la entrega de datos con métricas que permitan determinar si están funcionando correctamente	95% en el éxito de visualizar el estado de las estaciones de monitoreo.	Obtención de datos con un 95% de exactitud.

Tabla 1: Tabla de objetivos

	C1	C2	C3
OE1		**X**	
OE2	**X**		
OE3			**X**

Tabla 2: Trazabilidad de objetivos y causas

2.5 Alcance

- La aplicación al ser un prototipo se ejecutará según las especificaciones del desarrollador
- Se considerará que la aplicación se deberá ejecutar en un dispositivo PC o MAC compatible con html5 y Javascript.
- La aplicación está diseñada para ser operada por personal de la ONEMI Atacama
- La aplicación utilizará API de mapas google maps durante la etapa de prototipo
- Se construirá un emulador de estaciones y se utilizarán sus datos mientras no se cuenten con sensores físicos.
- Los datos almacenados en la base de datos implementada para el proyecto serán de propiedad de ONEMI atacama

2.6 Restricciones

- Los datos que se utilizarán inicialmente se generarán desde un emulador de estaciones de monitoreo
- No se considerarán datos reales provenientes de estaciones de monitoreo debido a restricciones propias de la DGAC y ONEMI Atacama
- La aplicación web será construida en AngularJS como framework de Javascript y HTML
- La API de estaciones de monitoreo será construida en JAVA utilizando librerías JAX-RS e implementada en el servidor de aplicaciones TOMCAT

2.7 Alternativas de solución

A continuación, se presentan algunas alternativas para dar solución a la problemática.

Reforzar la contención de cauces en zonas cercanas a la población:

Se propone implementar medidas de contención en los cauces cercanos a zonas pobladas, es posible construir muros de contención para las zonas donde puedan ocurrir desbordes para prevenir la destrucción de viviendas cercanas.

Ventajas.

- Fácil implementación
- Reducción de tiempos de desborde o eventual superación de la emergencia.

Desventajas.

- Alto costo en recursos.
- Aún existe el riesgo de desbordamiento y destrucción de los muros de contención.

Inspección visual de los cauces en zonas habitadas

Se propone mantener un inspector para los cauces cada vez que ocurra un evento meteorológico importante.

Ventajas.

- Constante flujo de información entre el inspector y las oficinas de ONEMI

Desventajas.

- Gasto en HH

- Solución no calcula tiempos de desborde
- No garantiza que la información se entregue en tiempo oportuno.

Desvío de cauces

Se propone construir desvíos en los cauces de mayor riesgo de daño en zonas habitadas

Ventajas:

- Reducción de riesgo de desbordamiento de los cauces
- Reducción de riesgo de daños a zonas habitadas

Desventajas.

- Solución de alto costo
- No se miden consecuencias ambientales al desviar los cauces.

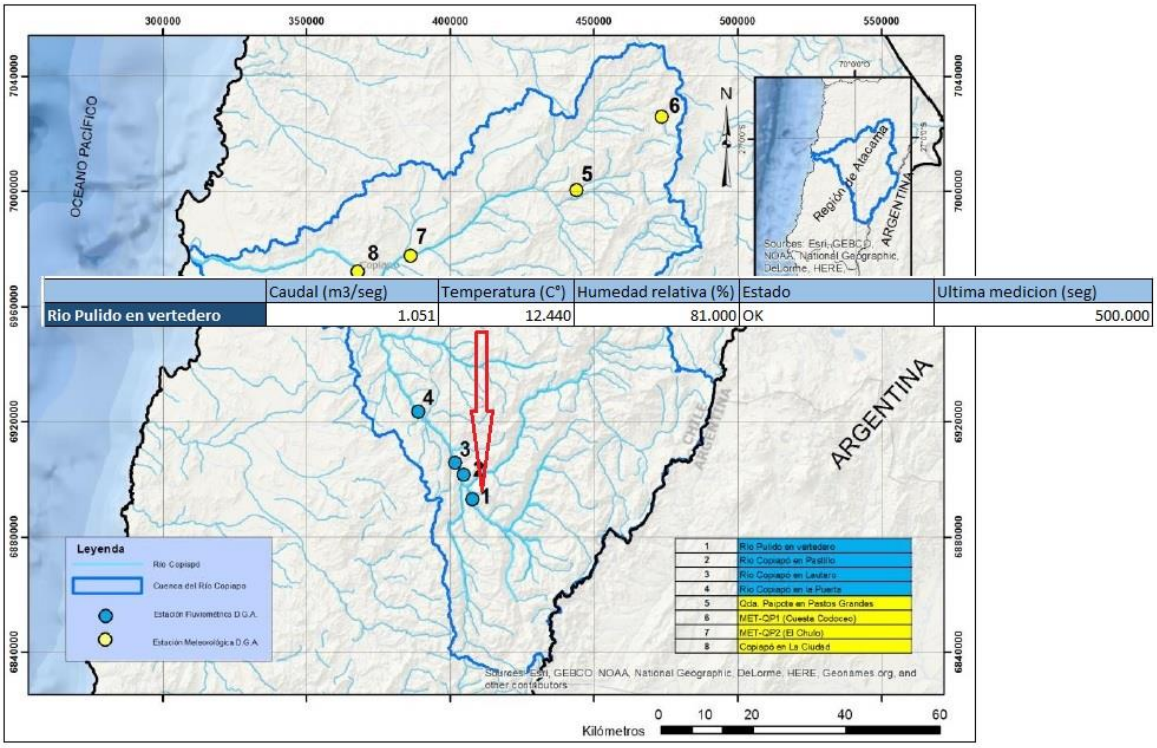
2.8 Propuesta.

Se propone desarrollar una plataforma Web que utilice API de mapas, que sea capaz de mostrar lo siguiente:

- Ubicación geográfica de las estaciones.
- Estado de los sensores, con sus respectivos datos en tiempo real
- Generación de informes utilizando los datos proporcionados por los sensores.

- Construcción de modelos ad-hoc para monitorear peligros considerando el envío de alertas ante eventos anómalos **Supuesto**

A continuación, se muestra un supuesto de una de las vistas de la aplicación:



	Caudal (m3/seg)	Temperatura (C°)	Humedad relativa (%)	Estado	Ultima medicion (seg)
Rio Pulido en vertedero	1.051	12.440	81.000	OK	500.000

Figura 4: Supuesto de la aplicación

En la figura se muestra lo siguiente:

- Mapa cartográfico de la región de Atacama
- Ubicación geográfica de las estaciones de monitoreo con sus sensores
- Información de la cuenta del rio como caudal, temperatura y humedad relativa.
- Estado del sensor, en este caso OK hace referencia a que el sensor se encuentra operativo.
- Última medición de datos provenientes del sensor.

2.10 Factibilidades

Factibilidad Económica: Ya que este proyecto es un prototipo según las necesidades observadas por la problemática identificada, no se cuenta actualmente con información que proporcione comportamiento de mercado que sirva como indicador económico para este proyecto, por lo que no se considerara la factibilidad económica para este proyecto.

Factibilidad Técnica: Aunque inicialmente no se cuenta con el equipamiento ni lo datos reales de las estaciones actualmente operativas, dentro de la planificación del proyecto se considera el desarrollo de un emulador de estaciones que permita generar datos que se puedan utilizar como prueba para las funcionalidades del sistema.

Las tecnologías que se utilizaran para el desarrollo del proyecto son de libre acceso y bien conocidas, por lo tanto, se dispondrá con tanto con soporte comunitario (foros especializados) y guía de expertos en el área.

Factibilidad operativa: ya que este proyecto se desarrollará con tecnologías libres, los usuarios finales o administradores de sistema, no tendrán problemas para operar y mantener la aplicación. Además, el equipo de desarrollo contara con diferentes fuentes de apoyo durante el avance del proyecto.

Al final del proyecto se entregará tanto la documentación de las aplicaciones y manuales de usuario de manera que el uso de la plataforma sea lo más sencilla para el usuario fina

2.11 Diseño de alto nivel.

2.11.1 Requerimientos de alto nivel

- Tener una vista que permita ver las estaciones de monitoreo georreferenciadas
- Visualizar las métricas entregadas por cada estación de monitoreo en un formato entendible
- Visualizar gráficos de comportamiento en el tiempo para cada estación.
- Definir una interfaz que permita fácilmente integrar nuevas estaciones de monitoreo.
- exponer toda la funcionalidad mediante una API para que pueda ser utilizada por otros sistemas.

2.11.2 Requisitos no funcionales del sistema.

Los principales requisitos no funcionales definidos para este proyecto son:

Nombre del RNF	Usabilidad
Descripción del RNF	La aplicación debe ser lo más intuitiva posible, de manera que el usuario u operador pueda con una rápida visualización verificar el estado de las estaciones de monitoreo junto a su comportamiento, además debe ser posible de georreferenciar las estaciones en una vista de mapas que también entregue información.
Requisito para la aprobación	El usuario debe ser capaz de utilizar la aplicación y obtener información útil con menos de dos horas de utilización o entrenamiento.

Tabla 3: RNF1

Nombre del RNF	Eficiencia
Descripción del RNF	Al ser una aplicación que eventualmente manejara altos volúmenes de datos, esta debe estar optimizada para solo realizar consultar por datos relevantes de manera que no consumir recursos innecesarios.
Requisito para la aprobación	El sistema debe responder a solicitudes a servicios web en menos de 5 segundos

Figura 5: RNF 2

Nombre del RNF	Mantenibilidad y portabilidad
Descripción del RNF	La aplicación se desarrollará en plataformas web, por lo que se debe ejecutar en browsers en diferentes sistemas operativos como Windows, MacOS y Linux.
Requisito para la aprobación	La aplicación se debe ejecutar sin problemas al menos en los 3 browsers más populares (Internet explorer, Firefox y Chrome)

Tabla 4: RNF 3

CAPITULO 3: Materiales y métodos

3.1 Metodología Seleccionada.

Para el desarrollo de este proyecto se seleccionaron en las siguientes metodologías considerando la naturaleza del proyecto y los tiempos asociados a este. Estos son:

- Metodología de gestión de proyecto: SCRUM
- Metodología de desarrollo: Iterativo/Incremental

3.1.1 Metodología de Gestión de proyecto

Se decidió utilizar SCRUM ya que para el tipo de producto que se propone, una metodología ágil es la más útil para este proyecto ya que el equipo de que desarrollara el proyecto será pequeño y además es flexible ya que se adapta fácilmente a los cambios de requerimientos de parte del cliente.

SCRUM organiza el desarrollo del proyecto y la entrega del producto en ciclos cortos llamados Sprints los que al final, hace posible entregar una parte del producto, este método permite reducir los riesgos ya que la planificación y revisión del avance puede cambiar al inicio y durante cada sprint. La duración de cada sprint de este proyecto será de 4 semanas.

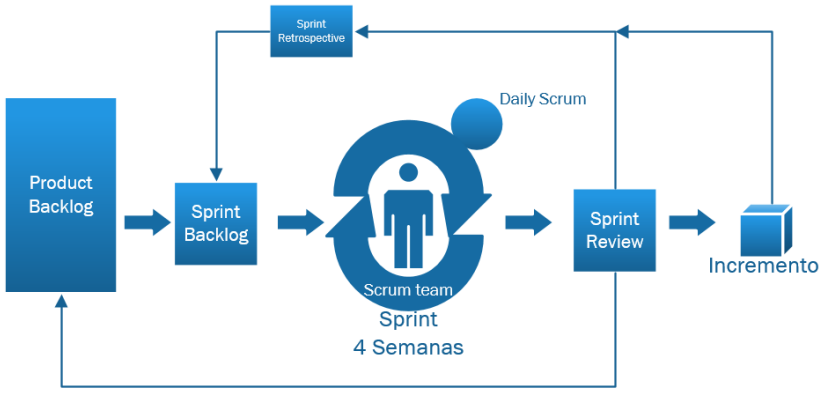


Figura 6: Metodología SCRUMRoles

SCRUM define roles para las personas que participaran durante el desarrollo del proyecto. Los roles se definen como se menciona a continuación:

- **Product Owner:** Romina Torres. Establecerá la priorización de los elementos o requisitos que se desarrollaran en cada iteración (Sprint para el caso de SCRUM) y realizara la traza de estos con el avance del proyecto.
- **Scrum Master:** Álvaro Cabrera: Encargado de llevar las historias de usuario a tareas y estimara el tiempo para cada una, confeccionara además los artefactos relacionados con esta etapa como el Product Backlog y Sprint Backlog y realizara el control y monitoreo del avance de estos.
- **Equipo de desarrollo:** Álvaro Cabrera. Realizará la ejecución de las tareas programadas para el sprint y se auto asignará cada una tomando en cuenta sus competencias e intereses.

3.1.2 Estimación de los Sprints.

Los requerimientos que se especificaran en el product backlog se estimaran según su dificultad, utilizando la serie de Fibonacci asignar un valor dependiendo de la complejidad del requerimiento, los números que se utilizaran son: 8, 5, 3, 2 donde 8 es el requerimiento de mayor dificultad y 2 el de menor dificultad respectivamente.

Para la priorización de las historias de usuario detalladas en el product backlog, se utilizará la técnica “MoSCoW”, donde el Product Owner asignará un valor de prioridad a cada H.U. Los valores que se utilizaran para este proyecto son:

- **Letra “M” (Must):** H.U que debe ser implementada ya que es de importancia vital para el funcionamiento del sistema.
- **Letra “S” (Should):** H.U que debería ser idealmente implementada, pero se puede prescindir de ella en el sistema.
- **Letra “C” (Could):** H.U que podría ser implementada, pero se puede prescindir de ella.
- **Letra “W” (Won’t):** H.U que no se necesita implementar inmediatamente, pero podría ser incluida en el futuro.

3.2 Composición de los Sprint

3.2.1 Entregables

Al término de cada sprint, se entregará una liberación del producto. Este incluirá una parte funcional del producto, el cual debe ser aceptado por el Product Owner, en base a los criterios de aceptación definidos al inicio del Sprint.

3.2.2 Plantillas y formatos

A continuación, se presenta la plantilla que se utilizara para registrar las historias de usuario con sus respectivos criterios de aceptación.

Product Backlog		Historias de usuario			
N°	Rol	Objetivo	Razon	Prioridad	peso
XX	Como admin	quiero tener	para generar	M	8
XX	como operador	quiero ver	para entregar	C	5
XX	como dueño	quiero desplegar	para decidir	S	3
XX	como gerente	quiero agregar	para notificar	W	2

Figura 7: Plantilla Product Backlog

Criterios de Aceptacion					
N° HU	N° Escenario	Criterio de aceptaci	Contexto	evento	Resultado/Comporatmiento esperado
A	1	escenario	En caso de<Conexto> y/o<Contexto>	cuando<evento>	el sistema <resultado/comportamiento>
	2	escenario	En caso de<Conexto> y/o<Contexto>	cuando<evento>	el sistema <resultado/comportamiento>
B	1	escenario	En caso de<Conexto> y/o<Contexto>	cuando<evento>	el sistema <resultado/comportamiento>
	2	escenario	En caso de<Conexto> y/o<Contexto>	cuando<evento>	el sistema <resultado/comportamiento>

Figura 8: Plantilla criterios de aceptación

3.3 Metodología de Desarrollo

La metodología iterativa incremental permite utilizar las etapas comunes del desarrollo de software como son el análisis, diseño, desarrollo y pruebas. Se compone de bloques definidos según las etapas antes descritas que iteran cada vez que termina el ciclo completo, al final de cada iteración se agreta un incremento al producto donde la idea principal perfeccionar el producto en cada iteración.

Esta metodología encaja con el tipo de producto a desarrollar ya que se recomienda utilizarla cuando los requerimientos están completamente definidos y que, al combinarlo con SCRUM, permite adaptarse a los cambios de requerimientos que puedan existe durante el transcurso del avance del proyecto.

Para este proyecto se combinarán ambas metodologías (SCRUM e iterativo incremental) donde la metodología de desarrollo se ejecutará dentro de cada sprint.

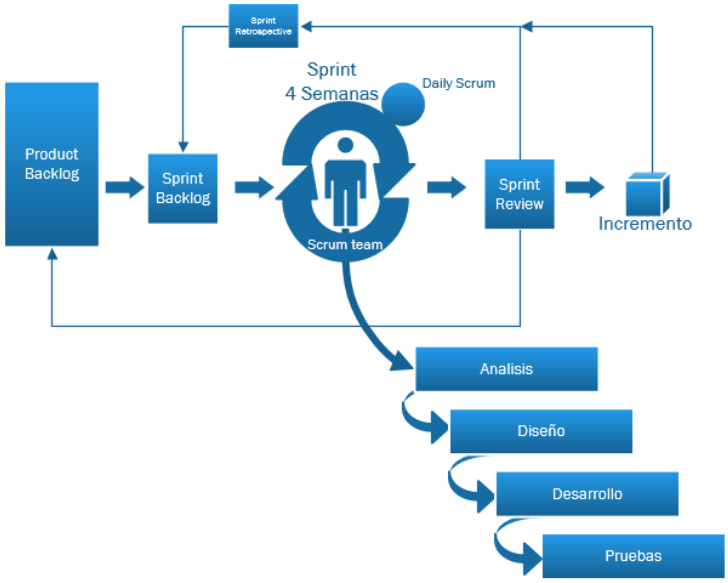


Figura 9: Metodología de desarrollo aplicada a la metodología de gestión

3.4 Plan de Proyecto

3.4.1 Product Backlog

A continuación, se presentan las historias de usuario que permitirán confeccionar el product backlog para el proyecto.

Product Backlog

N°	Rol	Objetivo	Razon	Prioridad	Ptos H
1	Como operador	quiero visualizar las estaciones de monitoreo en un mapa	para tener una referencia rápida de las estaciones en funcionamiento	M	2
2	como operador	Como operador quiero ver en el mapa el estado de las estaciones de monitoreo y ver qué datos están entregando	para verificar que los datos obtenidos sean correctos	M	5
3	como operador	quiero obtener el estado de funcionamiento de las estaciones de monitoreo	para solicitar revisión o reparación en caso de que no esté funcionando correctamente	M	3
4	como operador	quiero obtener reportes automatizados y bajo demanda del comportamiento y estado de los caudales de las fuentes de aguas	para entregar la información de manera rápida a las autoridades en caso de algún evento.	M	5
5	como operador	quiero visualizar alarmas de estado de los caudales de agua	para alertar a las autoridades de manera que estos tomen decisiones rápidamente	M	5
6	como administrador	quiero gestionar los umbrales de las alarmas según los parámetros entregados por los expertos	para obtener alarmas precisar y así evitar falsos positivos	S	8
7	como operador	quiero ver gráficos en función del tiempo del comportamiento de los caudales de los ríos	para tener una estimación de estadística de la variación de su estado en ciertas épocas del año	S	5
8	como administrador	quiero configurar un modelo predictivo de crecimiento y desborde de caudales	para obtener alertas que permitan notificar a las autoridades de manera rápida ante una emergencia.	S	8
9	como administrador	quiero agregar nuevas estaciones de monitoreo al sistema	para mantener todo el sistema de monitoreo centralizado	S	5
10	Como desarrollador	quiero tener un ambiente de pruebas	para emular la conexion con las estaciones de monitoreo mediante servicios web	M	8

Tabla 5: Product Backlog

3.4.2 Sprint Backlog

Según Las historias de usuario descritas en el product backlog, se realizará el desglose de tareas que corresponderán a cada iteración, para la realización del primer sprint se completaran las H.U 1 y 2 que se documentaran en el anexo A: Sprint 1.

3.4.3 Burn Down chart

Durante el desarrollo de cada sprint se realizará seguimiento de avance para cada una de las tareas descritas en el Sprint Backlog. Para esta tarea se utilizará una plantilla que se llenará con las tareas asignadas para cada sprint con la estimación en horas y horas restantes para terminar la tarea/sprint.

La plantilla utilizada genera un gráfico que referencia el avance esperado con el avance real del proyecto. De esta manera será posible ajustar los tiempos de avance y estimación de horas de trabajo para cada tarea.

Además del ajuste del tiempo de avance, la medición de avance del proyecto en horas, permite una mejor estimación de la capacidad de avance real del equipo de desarrollo. De esta manera es posible hacer estimaciones cada vez más precisas para cada iteración del proyecto.

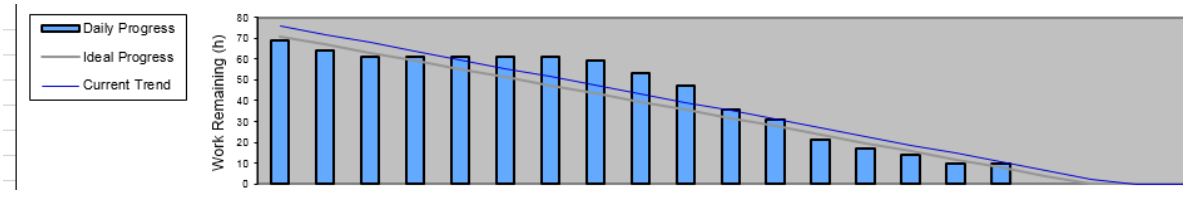


Figura 10: Ejemplo de sprint burndown

3.5 Planificación del proyecto

A continuación, se presenta la planificación del proyecto desde el punto de vista de la duración de los Sprint.

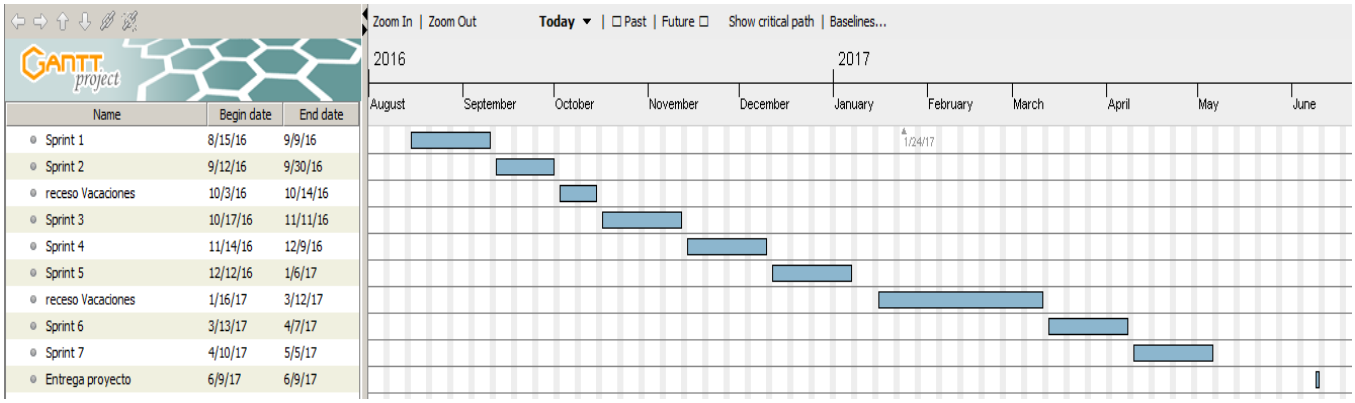


Figura 11: Carta Gantt desde el punto de vista del sprint

3.5.1 Sprint review

Al finalizar cada sprint, se realizará una reunión con el Product Owner, donde se entregará el producto resultante del sprint, que será trazado con el sprint backlog y los criterios de aceptación definidos en conjunto con el Product Owner.

El será el encargado de aprobar o rechazar el sprint o parte de él, se acordarán las tareas pendientes del sprint para la ejecución en el próximo sprint o en otro momento dentro del ciclo de vida del proyecto.

3.6 Plan general de pruebas

3.6.1 Responsables de las pruebas

Se asignarán testers a petición del P.O

3.6.2 Entorno de pruebas

Ambiente: Se desarrollarán en los laboratorios de la universidad, donde los testers tendrán acceso a la aplicación y al código del sistema.

Hardware:

Servidor: proporcionado por la universidad, donde se alojará la base de datos y la aplicación.

Pc de pruebas: proporcionado por la universidad en el laboratorio de informática.

Software:

- Pc con Windows 7 o superior
- Código fuente de la aplicación, con editores de texto o IDE para Java y AngularJS (Html, CCS, Javascript)
- Datos de prueba según especificación de etapas de testing.

3.6.3 Documentación de la fase de pruebas:

Los documentos que se generaran durante la fase de pruebas son:

- Scripts de pruebas y Casos de Prueba.
- Resultados de Pruebas siguiendo el formato especificado.
- Reporte consolidado de pruebas por módulo.

- Certificado de prueba para formalizar el hecho de que la aplicación en prueba ha pasado la prueba con éxito

3.7 Entorno de desarrollo

Se desarrollará en los laboratorios de informática de la universidad, donde se proporcionarán servicios en el datacenter de la institución en un ambiente controlado para alojar la aplicación.

3.7.1 Hardware:

- **Servidor:** proporcionado por el departamento de informática de la institución
- **Desarrollo:** Macbook Pro retina, core i5 2.5 Ghz, 8GB Ram, ejecutando Mac OSX El Capitan 10.11.6, proporcionado por el desarrollador
- **PC de pruebas:** proporcionado con el desarrollador.

3.7.2 Software:

Para el desarrollo de la aplicación web se utilizará:

- ATOM lastest
- NodeJS versión 7.10
-
- Apache web server versión 2.4.16

Para el desarrollo del emulador web-services se utilizará:

- Eclipse Neon
- Java JDK SE 1.8
- Oracle XE 11g
- Apache tomcat

3.7.3 Lenguaje de programación

Para el desarrollo del proyecto se utilizará el framework AngularJS para lograr presentar el producto como una aplicación web dinámica con integración de servicios web (web mashup), los lenguajes utilizados por este framework son:

- Angular.JS 1.5.x
- HTML 5
- Javascript
- Java
- SQL

3.7.4 Control de versión

Para el control de versión y documentación del código del proyecto se utilizará GIT, proporcionado por github.com

Github es una plataforma de desarrollo colaborativo que utiliza GIT que entre otras cosas permite el control de versiones para un determinado proyecto, este puede llevar registro de las versiones de código en la nube con la posibilidad de incluso editarlo en línea.

También se integra con el IDE que se utilizara en el proyecto por lo que actualizar el repositorio también se hace simple y eficaz.

El código se accederá en el siguiente enlace:

Aplicación web:

https://github.com/alvarock1985/onemimon:

API Estaciones de monitoreo:

https://github.com/alvarock1985/emusensor

Emulador de estaciones de monitoreo:

https://github.com/alvarock1985/clientEmu

- Durante el desarrollo se subirán los cambios en la plataforma utilizando **commits** que serán comentados con los cambios realizados en el código.
- Se trabajará con dos ramas de desarrollo (“branches”) que serán:
    - Master: almacenara el código final
- Para cada fin de Sprint se realizará una liberación de software en el directorio **Releases.**
- El esquema de versiones se realizará utilizando numeración de 3 dígitos donde:
    - El primer digito mencionara la versión final de software
    - El segundo mencionara cambios significativos a la versión
    - El tercero cambios menores a la versión.
- La versión 1.0 del sistema será la liberación del sistema al finalizar todos los sprint programados.

3.7.5 3.8 Registro de cambios de requerimientos

Los cambios de requerimientos o la agregación de nuevos se realizarán mediante la plataforma TRAC Project Manager.

Esta herramienta permita tanto controlar versiones, documentar y registrar cambios en los requerimientos, para este proyecto se utilizará para la documentación de los requisitos y la gestión de estos.

La plataforma permite el registro de tickets para solicitar los cambios y documentarlos. Esto permitirá llevar control del avance de los requisitos durante el ciclo de vida del proyecto.

3.8 Análisis y gestión de riesgos

Para el análisis y gestión de riesgos se realizará utilizando el siguiente método:

1. Planificación de la gestión de riesgo: se elabora un plan general según la identificación de riesgos y su mitigación
2. Identificación de los riesgos: Se incluye una lista de riesgos que pueden afectar el avance del proyecto.
3. Análisis Cualitativo: se evalúa la probabilidad de los riesgos utilizando una matriz de probabilidad e impacto del riesgo
4. Análisis Cuantitativo: se evalúa el nivel de riesgo utilizando la matriz de riesgo con los valores obtenidos del análisis cuantitativo del riesgo.
5. Plan de mitigación de riesgo: define una respuesta al riesgo según el resultado del añiláis cuantitativo.
6. Seguimiento y control de riesgo: lleva control de la mitigación de los riesgos y la aplicación de las medidas correctivas de estos. Además, evalúa nuevos riesgos que puedan aparecer durante el avance del proyecto.

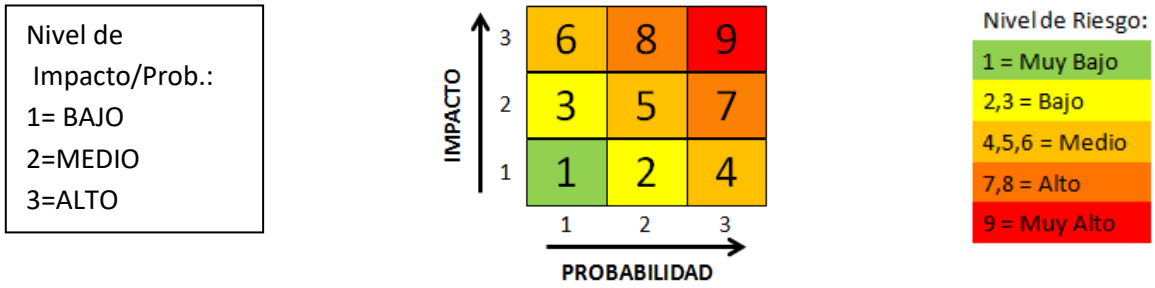


Figura 12: Tablas de riesgo

3.8.1 Identificación de riesgo técnico

Analisis de Riesgo Tecnico						
N°	**Riesgo**	**Posible resultado**	**Plan de mitigacion**	**Plan de contingencia**	**Evidencia**	**Estado**
1	Desconocer lenguave Javascript	Restraso en la puesta en marcha del proyecto	Realizar cursos/investigacion y elaborar prototipo	Contratar a instructor especialista para clases privadas / contratar desarrollador externo	Informe de curso realizado y prototipo funcional	Se realizaron cursos online y entrenamientos presenciales de la tecnologia por parte de la universidad
2	Desconocer lenguaje HTML	Restraso en la puesta en marcha del proyecto	Realizar cursos/investigacion y elaborar prototipo	Contratar a instructor especialista para clases privadas / contratar desarrollador externo	Informe de curso realizado y prototipo funcional	Se realizaron cursos online y entrenamientos presenciales de la tecnologia por parte de la universidad
3	Desconocer el framework Angular.JS	Restraso en la puesta en marcha del proyecto	Realizar cursos/investigacion y elaborar prototipo	Contratar a instructor especialista para clases privadas / contratar desarrollador externo	Informe de curso realizado y prototipo funcional	Se realizaron cursos online y entrenamientos presenciales de la tecnologia por parte de la universidad
4	Desconocer consumir servicios con Angular.js	Restraso en la puesta en marcha del proyecto	Realizar cursos/investigacion y elaborar prototipo	Contratar a instructor especialista para clases privadas / contratar desarrollador externo	Informe de curso realizado y prototipo funcional	Se realizaron cursos online y entrenamientos presenciales de la tecnologia por parte de la universidad
5	Desconocer construccion de webservices SOAP con AXIS	Retraso en las pruebas de conexión para los subsitemas del proyecto	Realizar cursos/investigacion y elaborar prototipo	Contratar a instructor especialista para clases privadas / contratar desarrollador externo	Informe de curso realizado y prototipo funcional	Se realizaron cursos online y entrenamientos presenciales de la tecnologia por parte de la universidad
6	Implementacion y construccion de modelo de datos y motor de base de datos	Retraso en las pruebas de conexión para los subsitemas del proyecto	Realizar cursos/investigacion y elaborar prototipo	Contratar a instructor especialista para clases privadas / contratar desarrollador externo	prototipo funcional	Se realizaron cursos online y entrenamientos presenciales de la tecnologia por parte de la universidad
7	Desconocimiento de lenguaje JAVA e implementacion para webservices	Retraso en las pruebas de conexión para los subsitemas del proyecto	Realizar cursos/investigacion y elaborar prototipo	Contratar a instructor especialista para clases privadas / contratar desarrollador externo	prototipo funcional	Se realizaron cursos online y entrenamientos presenciales de la tecnologia por parte de la universidad

Tabla 6: Riesgo Técnico

CAPITULO 4: Resultados y Discusión

4.1 Arquitectura, Análisis y Diseño

4.1.1 Patrón arquitectónico

La aplicación web que se desarrollará se construirá según el modelo de 3 capas (presentación – negocio- datos) que se ajusta al patrón arquitectónico MVC y en capas

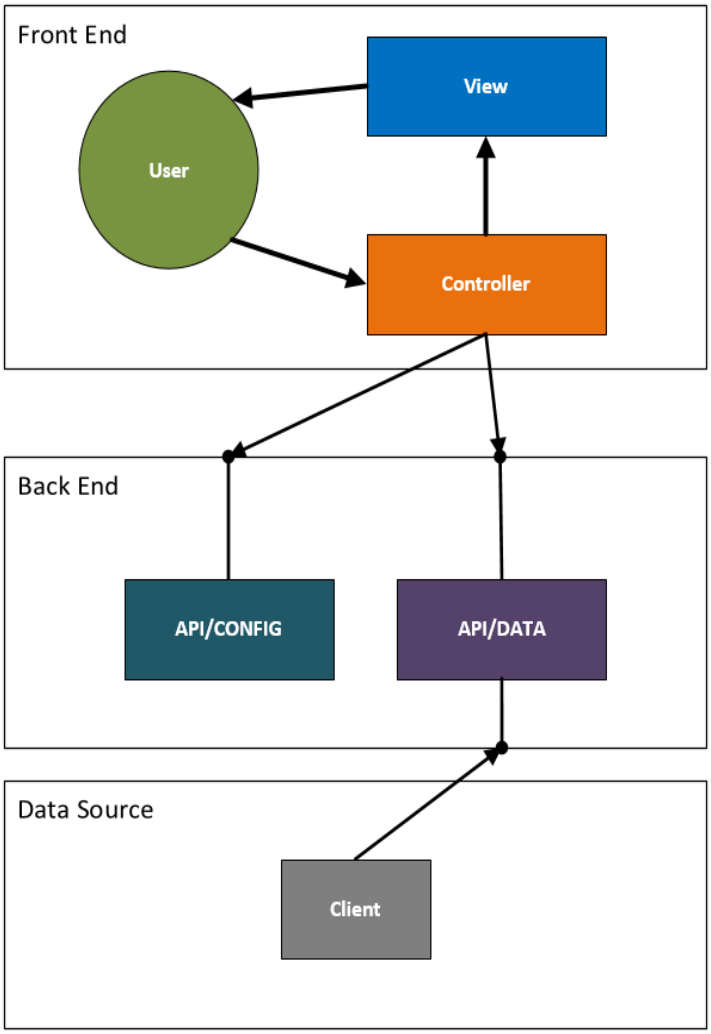


Figura 13: Diagrama de arquitectura de alto nivel

- **Contenedor Front End:** En este contenedor están presentes los componentes que se ejecutan por el lado del cliente, las capas presentes en este contendor son:
    - **Controller:** conceptualmente esta capa se encarga de contener la lógica de negocio para la aplicación, para este proyecto la capa de “Controller” contendrá los métodos necesarios para consumir servicios desde la capa de “Back End” y guardarlos para ser utilizados por la capa View para la representación de los datos en la aplicación. Las interacciones con las demás capas están relacionadas con las solicitudes iniciadas por el usuario, una vez que la capa “Controller” reciba solicitudes de parte del usuario, se consumen los servicios presentes en la capa “API” que devolverá los datos que serán presentados a la capa “View”.

    - **View:** Esta capa es la que recibe los datos y métodos desde la capa “Controller” de manera que sea posible representar estos datos de una forma que sea entendible para el usuario, por ejemplo, gráficos, tablas estadísticas o representaciones en mapas. Aunque esta capa forma parte del componente visible de la aplicación, su interacción con las demás capas inicia en la capa “Controller” ya que ella se encarga de solicitar y entregar los datos a la capa “View” y esta los muestra al usuario final.

- **Contenedor Back End:** Este contenedor posee el/los componentes que se ejecutan por el lado del servidor, la capa presente en este contenedor es:

    - **API:** Esta capa posee los componentes que entregan servicios web que permiten mostrar e ingresar datos a una base de datos definida previamente. Toda la lógica de negocio necesaria para la muestra de datos se implementará en esta capa ya que la principal idea es que los

componentes presentes en el contenedor de “Front End” no procesen datos ya que solo deberían mostrarlos.

- **Data Source:** Este contenedor representa la fuente de datos para la aplicación, si bien en la entrega final del producto no se considerara como componente a desarrollar es parte de la arquitectura ya que este se encargará de consumir los servicios que permitan ingresar datos.

    - **Client:** capa de cliente de web services, la aplicación de cliente será externa al producto final ya que puede representar una estación de monitoreo que ingresará los datos utilizando la capa “API”. En este punto del proyecto esta capa representa una estación de monitoreo emulada que está haciendo uso de los servicios web para insertar datos.

4.1.2 Diagrama de despliegue

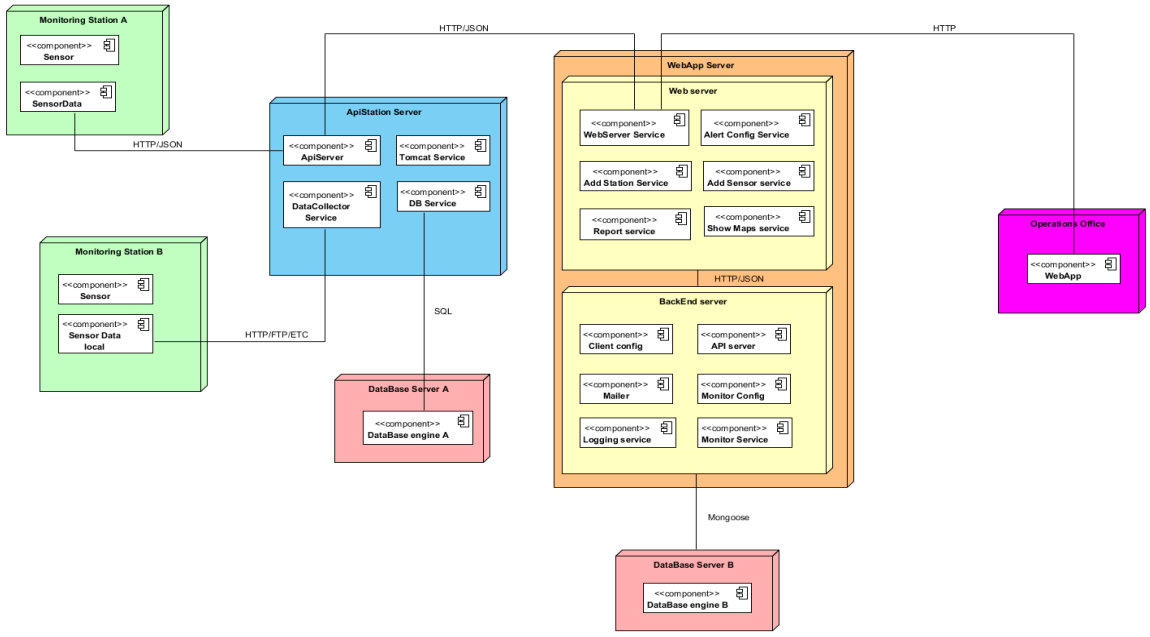


Figura 14: Diagrama de componentes desplegado

En la figura 4.8 se muestran los componentes que forman parte de la arquitectura del proyecto.

Para Este Sprint se desarrollaron componentes para el Api (ApiStation) de emuladores y también en la aplicación Web (WebApp Server), para las historias de usuario planificadas para esta iteración.

A continuación, se dará explicación a los nodos y componentes presentes en este diagrama:

- **ApiStation Server:** Este nodo es el servidor de aplicaciones implementado con Tomcat 8.0 para la ejecución del api de estaciones de monitoreo, los componentes presentes en este nodo que actualmente se encuentran implementados son:

- **ApiService:** contiene los métodos expuestos como servicios REST tanto como para obtener datos e insertarlos en la base de datos.
- **DB Service:** contiene los métodos para la conexión a la base de datos implementada.

- **WebApp Server:** El nodo de web server contiene los componentes que consumen servicios web desde el api de sensores y muestran estos datos al usuario final, los componentes que actualmente están implementados son:
  - **Web server service:** servidor web para la aplicación web implementado en Apache web server 2.4, este es capaz de procesar las solicitudes desde un cliente web y además las transacciones entre la aplicación web y el api de estaciones.
  - **Show maps service:** permite la conexión a un api de mapas. Para esta aplicación se implementa el api de Google maps.
  - **Chart Service:** implementa la representación de los datos como graficos en diferentes vistas.
  - **Save to PDF:** implementa la posibilidad de guardar reportes como PDF.
  - **Report Service:** permite la vista de reportes de manera de imitar los reportes obtenidos como parte de la documentación de este proyecto.

- **BackEnd Server:** El nodo de backend server contiene los componentes de la API/CONFIG de la arquitectura antes mencionada, esta encargada de dar servicios de almacenamiento y ejecución de las tareas rutinarias del sistema, como ejecutar el monitoreo del estado de las estaciones y ejecutar los clientes de prueba para el ingreso de datos. Este nodo contiene los siguientes componentes:
  - **Client config:** guarda la configuración para los clientes emulados del sistema, donde se puede modificar la frecuencia de envío de datos y los rangos de datos.

- **Monitor config:** Guarda la configuración de los monitores para la aplicación, donde se modifican los criterios para detectar si existen fallas en los sensores. Por ejemplo, que un sensor no esté enviando datos o que los datos estén fuera de un rango establecido.
- **Logging service:** crea logs según los eventos que ocurren tanto por el lado del cliente como por el monitor, se acoge al estándar de syslog definido por el IEEE RFC 5424, para los niveles de severidad de los mensajes.
- **Mailer:** contiene las librerías para la conexión a un servidor de correo electrónico para el envío de alarmas.
- **Monitor service:** ejecuta las rutinas de monitoreo para las estaciones, donde se revisa si las estaciones cumplen con ciertos criterios como por ejemplo que los datos estén en ciertos rangos de datos o que los sensores estén enviando datos.

- **Monitoring Station A:** Este nodo representa un emulador de estación de monitoreo que actualmente está ingresando datos utilizando servicios web del api. Este nodo se ejecuta utilizando la aplicación de atomización de tareas Crontab presente en la distribución de Linux Centos 5.7. Los componentes presentes en este nodo son:
  - **Sensor:** Emula los datos que se obtienen de los sensores presentes en una estación de monitoreo, estos pueden ser sensores pueden ser termómetros, caudalometros o sensores de humedad.
  - **SensorData:** Componente que genera los datos y consume el servicio del api para insertar datos.

4.1.3 Diagrama de despliegue – Implementación servidor de prueba

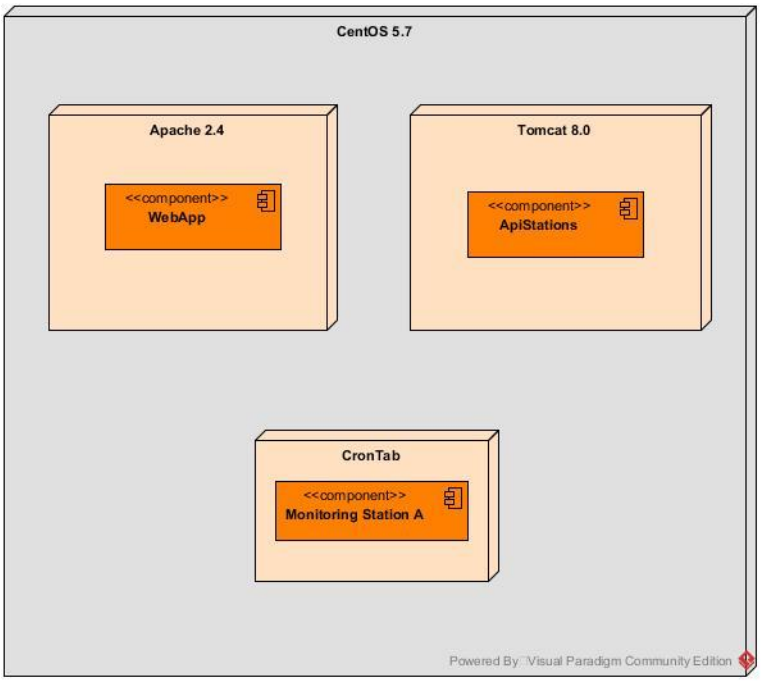


Figura 15: Diagrama de despliegue – implementación en servidor de pruebas

Este diagrama representa la implementación actual de los componentes mencionados anteriormente. Ya que el proyecto aún se encuentra en fase de prototipo la implementación de los diferentes componentes se realizan en un solo servidor, aunque idealmente se deben implementar en servidores separados para realizar pruebas reales de la eficiencia de las comunicaciones entre estos.

4.1.4 Modelo relacional para base de datos

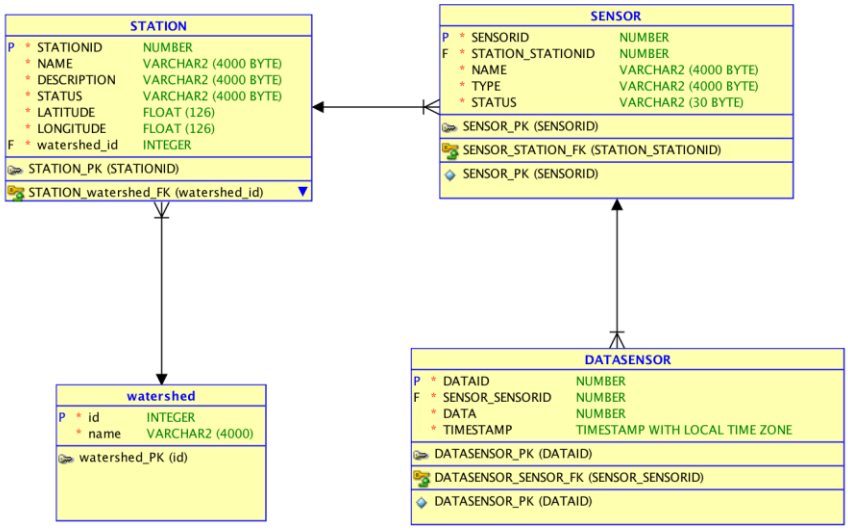


Figura 16: Modelo relacional base de datos

Esta figura representa el modelo relacional para la base de datos implementada en el nodo mencionado anteriormente. El motor de base de datos utilizado para esta parte del proyecto es Oracle 11g XE.

4.1.5 Diagrama de secuencia de alto nivel

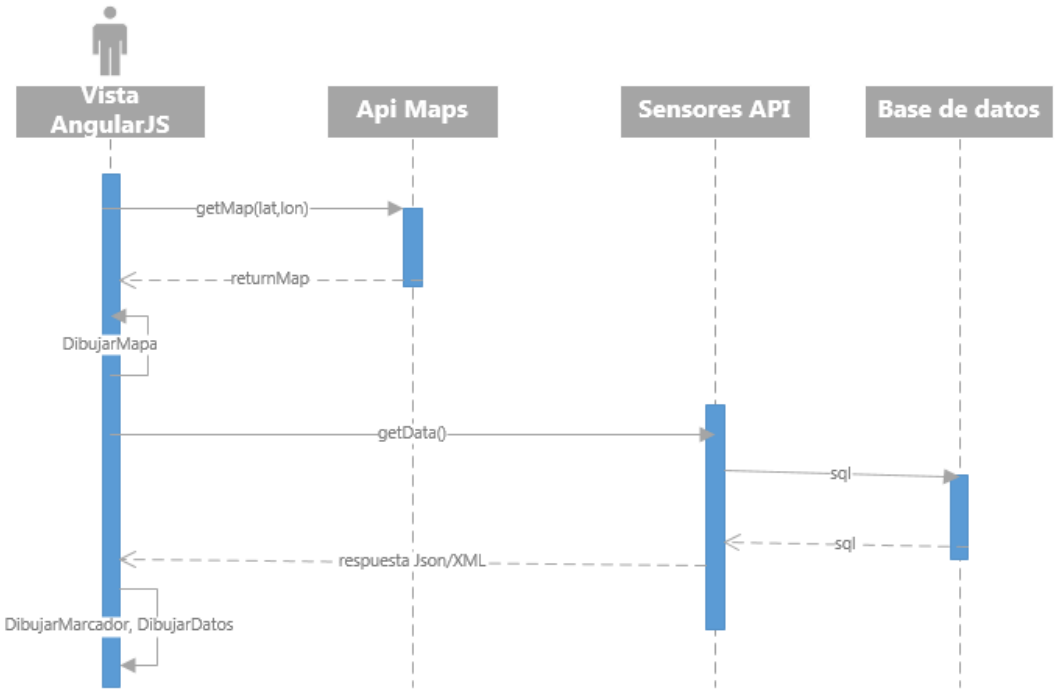


Figura 17: Diagrama de secuencia

4.1 Diseño detallado

4.1.1 Diseño detallado aplicación web

A continuación, se presenta el diseño detallado para la aplicación web por el lado del cliente, dada la arquitectura mencionada anteriormente en este diagrama solo se incluyen los componentes que se ejecutan en el cliente.

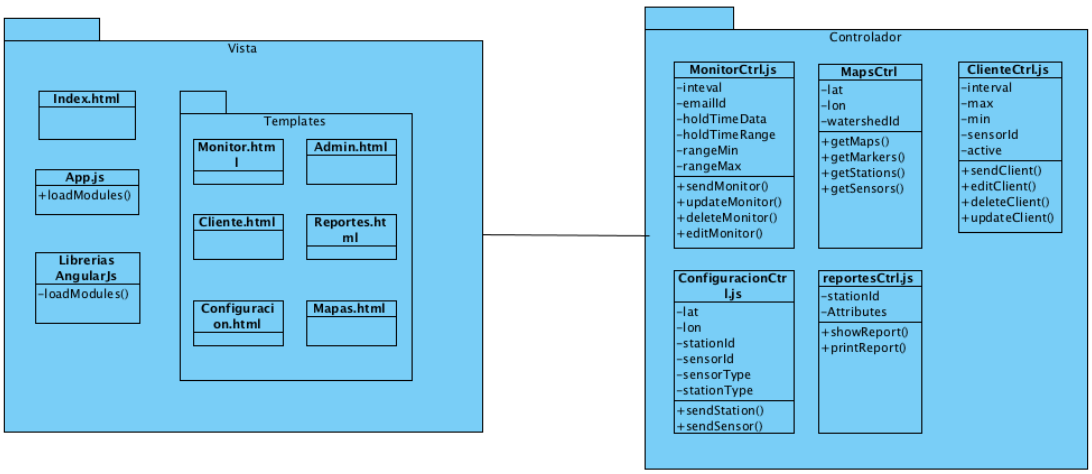


Figura 18: Diseño detallado aplicación web

Dado que la aplicación web no se construye con el paradigma de desarrollo orientado a objetos, no se menciona la interacciones entre los componentes que estan presentes en el diseño. Los dos nodos descritos en el diagrama representan los componentes ejecutados en la capa de front-end de la aplicación, es decir, los que se ejecutan por el lado del cliente. A continuación, se describen las funciones de cada nodo.

Nodo Vista: contiene los elementos visuales de la aplicación. Debido al lenguaje de programación y framework utilizados, los componentes se cargan como plantillas dentro de una vista principal, en este caso index.html. Todos los demás componentes se cargan sobre la plantilla principal. En este nodo solo se considera como componente de lógica el elemento app.js y las librerías de angular relacionada, este componente es el encargado de cargar todos los módulos externos y los controladores de la aplicación de manera que puedan ser utilizados en la aplicación.

Nodo controlador: contiene la lógica que permite el intercambio de datos con las api de configuración y datos. Cabe destacar que cada plantilla tiene un controlador separado por lo tanto no hay intercambio de mensajes entre los diferentes controladores, cada uno trabaja de forma separada al resto.

4.1.2 Diseño detallado API/Data

Para el diseño detallado de la api de datos, primero se especifica un diagrama de paquetes que describe la función de alto nivel de para la aplicación, de esta manera se puede comprender de mejor manera el funcionamiento de las clases que contiene cada uno de los paquetes que se describirán a continuación.

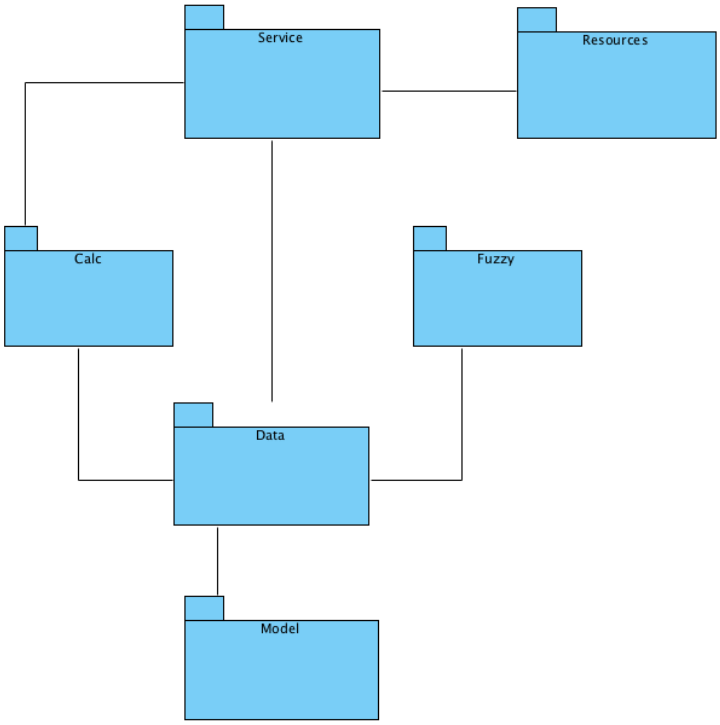


Figura 19: Diagrama de paquetes API/DATA

La idea principal de esta aplicación es representar las tablas de la base de datos y exponerlas como servicio web RESTful, para esto se dividieron las clases en paquetes de java que representan su función.

Model: cada clase de este paquete representa una consulta de la base de datos, o bien un conjunto de datos que se utilizaran en la aplicación web. Estas clases solo contienen los atributos que se mostrara posteriormente como mensaje JSON, la decisión de construir las clases de esta forma está relacionada con el framework para servicios web Jesey que devuelve instancias de objetos como formato para el mensaje JSON.

Data: contiene las clases que representan la conexión a la base de datos y las diferentes consultas realizadas a esta. Cada clase es una consulta diferente sobre los

componentes utilizados para la aplicación web, las clases de este paquete retornan instancias del paquete model.

Calc: contiene las clases y métodos que se utilizan para el cálculo de algunas operaciones utilizadas por otros métodos, como por ejemplo el cálculo de percentiles utilizado para el cálculo de drifts provenientes de los datos de los sensores.

Fuzzy: contiene las clases que representan los métodos utilizados para el cálculo de drifts que ayuda a determinar si existe un evento donde ocurra un cambio abrupto en los datos que se reciben desde un sensor en una estación de monitoreo. Estos métodos pueden determinar si existe evidencia de que este evento es real o bien solo es una falla o ruido en la medición.

Service: las clases de este paquete funcionan como intermediario entre la lógica antes mencionada y el recurso que se expone como servicio web. En estas clases se instancian los objetos que representan los datos o cálculos consultados.

Resource: clases que se presentan como recurso del servicio web, en estas clases se representan, por ejemplo, la url del recurso solicitados con sus parámetros. Los métodos que se ejecutan aquí instancian las clases del paquete service.

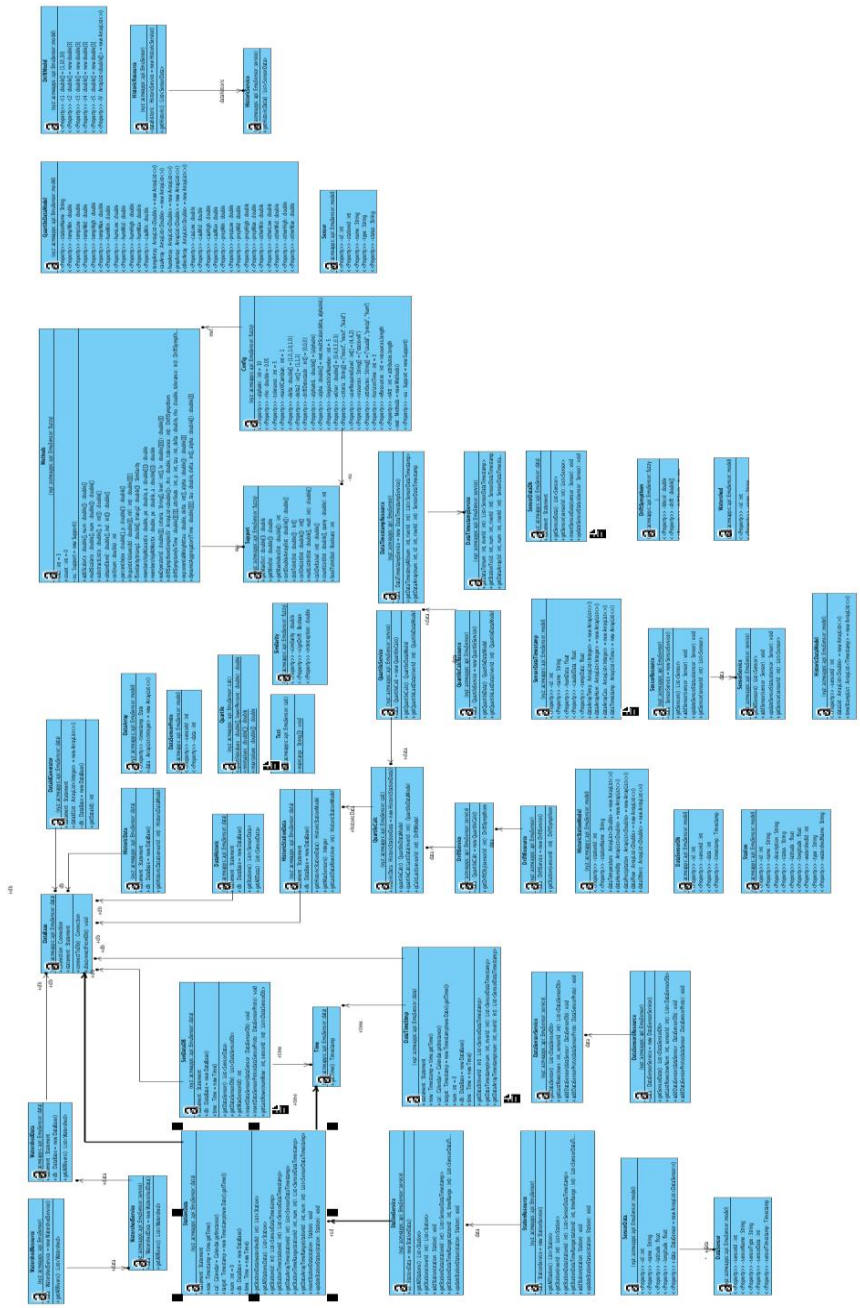


Figura 20: diagrama de clases API/DATA

4.1.3 Diseño detallado API/CONFIG

Este componente es resultado de la necesidad de crear una herramienta que ejecute tareas rutinarias de forma constante y que puedan ser configuradas fácilmente desde la aplicación web. Esta aplicación está construida en Javascript utilizando NodeJS y express como framework para la utilización de este lenguaje de programación por el lado del servidor.

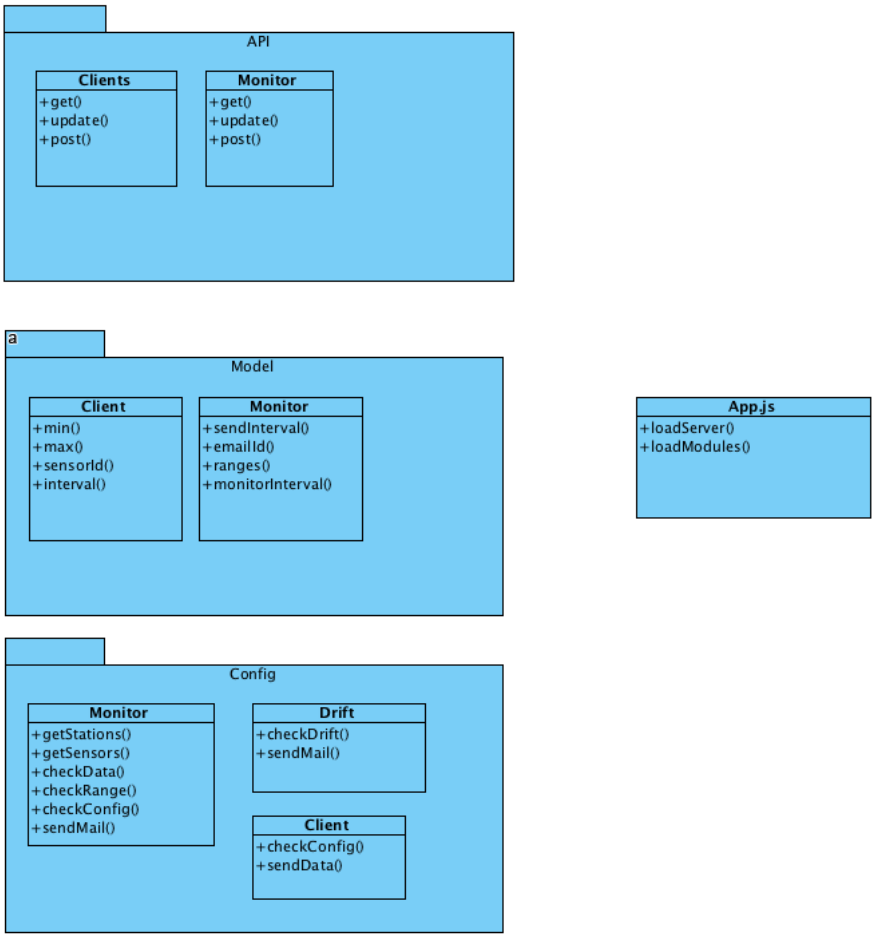


Figura 21: diseño detallado API/CONFIG

El diseño para este componente es representado por un diagrama de paquetes donde cada paquete es una función de la aplicación:

Config: contiene los métodos que se ejecutan de forma rutinaria, las funciones que se ejecutan aquí son:

Monitor: conjunto de funciones que chequean el estado de las estaciones según el criterio configurado de umbral de datos y de tiempo de envio de datos. Además de revisar el estado de los datos en un intervalo de tiempo configurable, chequea que la configuración se mantenga. En caso de que ocurra un cambio en la configuración el monitor se reinicia con la nueva configuración.

Client: funciones que ingresan datos en la API/DATOS de forma constante en un intervalo de tiempo configurado. Al igual que el componente de monitor, también se revisan los cambios de configuración, si se detecta que la configuración cambio el cliente se reinicia con la nueva configuración.

Drift: se revisan que los cambios abruptos en los datos sea un evento real y no ruido o datos erróneos de parte del sensor. Si se detecta un drift se informa por correo electrónico.

4.2 Experimento

El experimento que se realizara corresponde a una de las funcionalidades más interesantes de la plataforma que es la comprobación de eventos en los datos entregados por la fuente, es decir identificar si efectivamente existe un evento donde los datos cambian de forma abrupta o solo es ruido o un dato alto por funcionamiento erróneo temporal de un sensor.

El algoritmo utilizado para la detección de los drifts está basado en el algoritmo de detección de drifts propuesto por el profesor guía. El algoritmo propuesto es el siguiente:

Algorithm 1 Compute if there is enough evidence that a linguistic value has drifted

Require: Δ, *tolerance*, ρ and the temporal sequence of membership functions of LV_p^j: $(\mu_p^j)_\Delta$: $\mu^\tau \succ \mu^{\tau-1} \succ \mu^{\tau-2} \succ ... \succ \mu^{\tau-\Delta+2} \succ \mu^{\tau-\Delta+1}$

Ensure: Return true if there is enough evidence that the linguistic value has drifted through time.

1: **for** $t = \tau$ downto $t = \tau - \Delta$ **do**
2: Compute the similarity between the two temporal fuzzy numbers $sim = Sim(\mu^t, \mu^{t-1})$ calling Algorithm 2 (which uses the proposal for triangular fuzzy numbers proposed in [**LZK09**]).
3: Compute $\varsigma = \begin{cases} 1 & if\ c_{\mu^t} < c_{\mu^{t+1}} \\ 0 & if\ c_{\mu^t} = c_{\mu^{t+1}} \\ -1 & otherwise \end{cases}$
4: Compute the drift's symptom $drift = |\varsigma(1 - sim)|$ using equation (5.5)
5: **if** $drift > \rho$ **then**
6: $drift[t] = \varsigma$;
7: **else**
8: $drift[t] = 0$;
9: **end if**
10: **end for**
11: Compute *counter* in the drifts vector, which is the largest sequence of drift symptoms with same ς (zeros do not interrupt the sequence but they are not counted).
12: **if** *counter* $\geq$ *tolerance* **then**
13: return **true**(there is enough evidence that the linguistic value has drifted).
14: **else**
15: return **false**
16: **end if**

Figura 22: Algoritmo de detección de drifts

Como resumen, el algoritmo mide la diferencia entre los datos lingüísticos de una serie de tiempo, cada serie temporal mide el valor mínimo, medio y máximo para un sensor dado. Luego se comparan los valores utilizando el algoritmo antes mencionado. Si el algoritmo detecta una diferencia entre los datos en el tiempo durante un tiempo de tolerancia marca el evento como drift y por lo tanto se le debe poner atención. Los datos que devuelve el algoritmo son:

Confirmación de drift: si se confirma que el drift supero el tiempo de tolerancia se marca como valor “1”.

Valor de drift: según las mediciones, se devuelve un valor de drift que representa la distancia entre la comparación de los dos valores, este valor tiende a 1 donde 1 significa drift detectado, este valor es usado para determinar si el drift es válido. Ej.: si el conteo de drifts detectados es igual o mayor que la tolerancia, se confirma un drift valido.

Utilizando los datos resultantes del cálculo, tanto de los percentiles como de la ocurrencia de drift, es posible graficar el estado de los sensores comparando los valores de referencia contra los valores de la serie de tiempo como se muestra a continuación.

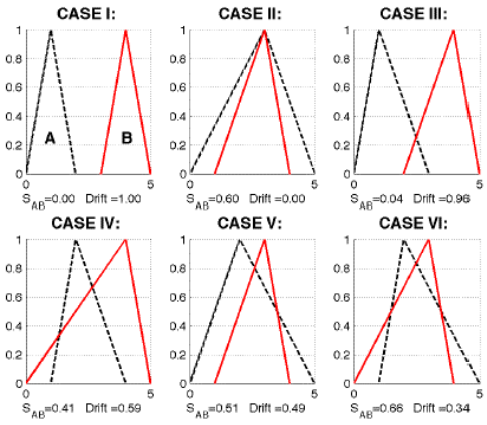


Figura 23: Casos gráficos de drifts

En el grafico se puede observar los diferentes casos que podrían ocurrir cuando se comparan los datos, donde se identifica en el caso I el cambio en la medición de los datos y donde se identifica el drift.

4.2.1 Caso de prueba

Se presenta un caso de prueba con datos ficticios donde se comprobará el funcionamiento del algoritmo y como este está aplicado en la plataforma.

Sea una serie temporal t1,t2,t3,t4 y t5, donde t1 representa el valor de referencia y con una tolerancia de 3, es decir para considerar que el drift es válido se deben obtener tres coincidencias consecutivas.

Se entregan los datos como arreglo ordenado según su valor lingüísticos, mínimo, medio y máximo:

- T1 = [1,7,15]
- T2= [2,10,20]
- T3=[5,6,30]
- T4=[20,30,40]
- T5=[25,35,50]

Calcular un síntoma de drift, para esto se comparan los centroides o valores medios de los datos ingresados.

- T1,T2 = 7<10 → hay síntoma de drift
- T1,T3 = 7>6 → no hay síntoma de drift
- T1,T4 = 7<30 → hay síntoma de drift
- T1,T5 = 7<35 → hay síntoma de drift

Calcular similaridad, para esto se realizan operaciones matemáticas sobre los datos obtenidos según los criterios descritos en los casos del algoritmo. Las operaciones que se realizan sobre los datos son:

Caso I

Si centroide1 = centroide2:

Si (maximo1 – minimo1)<(maximo2 – minimo2) entonces:

Similaridad = (maximo1 – minimo1)/(maximo2 – minimo2)

De otro modo:

Similaridad = (maximo2 – minimo2)/ (maximo1 – minimo1)

Caso II

Si maximo1 <= maximo2:

Similaridad = 0

Caso III

Si (maximo1 <= maximo2 Y minimo1 <= minimo2 y maximo1 > minimo2):

intercepto = 0.5 * (((maximo1 - minimo2) * (maximo1 - minimo2)) / (maximo1 - centroide1 + centroide2 - minimo2));

Similaridad = Intercepto / (0.5 * (maximo1 - minimo1 + maximo2 - minimo2) - Intercepto);

Caso IV

Si maximo1 <= maximo2 Y minimo1 > minimo2:

h1 = (minimo1 - minimo2) / (centroide2 - minimo2 + minimo1 - centroide1);

h2 = (maximo1 - minimo2) / (centroide2 - minimo2 + maximo1 - centroide1);

intercepto = 0.5*(maximo1 - minimo2)*h2 - 0.5*(minimo1-minimo2)*h1;

Similaridad = (intercepto / (0.5 * (maximo2 - minimo2 + maximo1 - minimo1) - intercepto));

Caso V

Si maximo1 > maximo2 && min1 <= minimo2:

h1 = (maximo1 - minimo2) / (maximo1 - centroide1 + centroide2 - minimo2);

h2 = (maximo1 - maximo2) / (maximo1 - centroide1 + centroide2 - maximo2);

intercepto = 0.5*(maximo1-minimo2)*h1 - 0.5*(maximo1-maximo2)*h2;

Similaridad = intercepto / (0.5 * (maximo1 - minimo1 + maximo2 - minimo2) - intercepto);

Caso VI

Si maximo1 > maximo2 Y minimo1 > minimo2:

h1 = (minimo1 - minimo2) / (centroide2 - minimo2 + minimo1 - centroide1);

h2 = (maximo1 - maximo2) / (maximo1 - centroide1 + centroide2 - maximo2);

h3 = (maximo1 - minimo2) / (maximo1 - centroide1 + centroide2 - minimo2);

intercepto = 0.5*(maximo1 - minimo2)*h3 - 0.5*(minimo1-minimo2)*h1 - 0.5*(maximo1-maximo2)*h2;

Similaridad = intercepto / (0.5 * (maximo2 - minimo2 + maximo1 - minimo1) - intercepto);

Se calcula para los datos mencionados anteriormente:

- T1,T2(Caso III):
  - Intercepto: 0,5*(((15-2) * (15-2))/(15-7+10-2)) = 5,281
  - Similaridad: 0.8125/(0,5 * (15 -1 + 20-2) – 0,8125) = 0.49
- T1,T3(Caso VI):
  - H1: (1-5)/(6-5+1-7) = 0,8
  - H2: (15-30)/(15-7+6-30) = 0,93
  - H3: (15-5)/(15-7+6-30) = -0.625
  - Intercepto: 0,5(15-5)*-0.635-0.5(1-5)*0,8*0,5(15-30)*0,93 = 5,445
  - Similaridad: 5,445/(0,5 * (30 -5 + 15-1) – 5.445) = 0,38
- T1,T4(Caso II)
  - Similaridad=0
- T1,T5(Caso II)
  - Similaridad = 0
  -

Luego de tener los datos correspondientes a síntoma de drift, similaridad e intercepto, se procede a determinar si la cantidad de síntomas corresponden a drift verdadero según la tolerancia utilizada y calcular el índice de drift.

Calcular el valor de drift según algoritmo: Drift = 1- similaridad

- Drift(T1,T2): 1 - 0.49 = 0,51
- Drift(T1,T3): 1- 0,38 = 0,62
- Drift(T1,T4): 1 – 0 = 1
- Drift(T1,T5): 1- 0 = 1

Comparar la cantidad de drifts positivos (valor de drift = 1) con la tolerancia.

Para tolerancia de 3, entonces:

Drifts=2<tolerancia=3

Por lo tanto, se concluye que el drift no es válido para el espacio de tiempo medido.

4.2.2 Caso de prueba en el sistema

Dado el ejemplo anterior, se utilizarán los mismos datos para el cálculo del drift. Se omitirá el cálculo específico para cada valor, solo se incluye el valor final que corresponde a que el sistema encontró o no drift valido.

```
double[] t1 = {1, 7, 15};
double[] t2 = {2,10,20};
double[] t3 = {5,6,30};
double[] t4 = {20,30,40};
double[] t5 = {25,35,50};


ArrayList<double[]> lV = new ArrayList<>();
lV.add(t1);
lV.add(t2);
lV.add(t3);
lV.add(t4);
lV.add(t5);

DriftSympthom ds = met.driftSympthomSimple(lV, rho, tolerance);
System.out.println("Drifts encontrados: "+ Arrays.toString(ds.getDrift()));
System.out.println("Drift valido 0: no 1: si"+ ds.getBbool());
```

código de alto nivel para el cálculo de drifts

En la figura anterior se muestra el código de alto nivel en Java para el cálculo de los drifts, se considera de alto nivel ya que no se muestra la lógica en backend para computar los valores según se describió en el caso de prueba, el resultado del código ejecutado es el siguiente.

```
Drifts encontrados: [0.51, 0.62, 1.0, 1.0, 0.0]
Drift valido si: 1, no: 0. resultado: 0.0
```

Figura 24: resultado de ejecución del algoritmo

En la aplicación web se agrega un módulo que puede representar de forma gráfica el cálculo de los drift como se mostró en el caso de prueba.

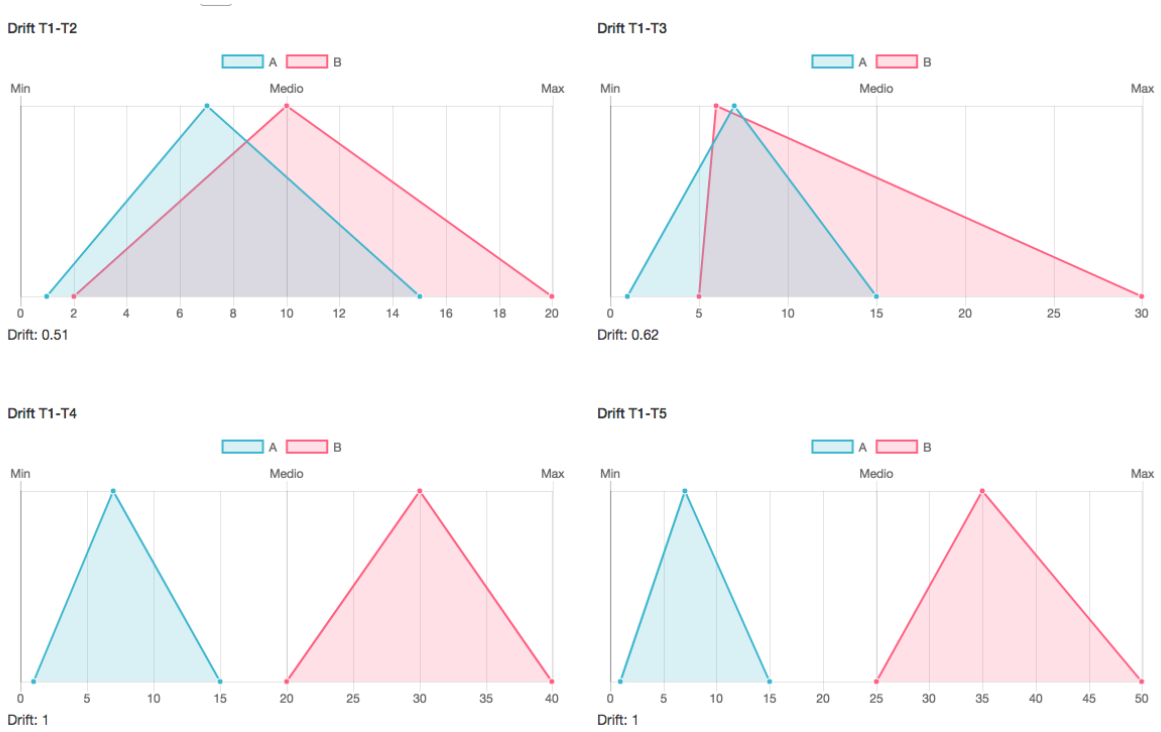


Figura 25: Representación gráfica de la medición de drifts

El resultado de este experimento, demuestra que el algoritmo retornando los valores esperados, esto se puede apreciar en la figura anterior ya que se acerca a la propuesta original.

La aplicación de este tipo de algoritmos en aplicaciones donde se deba recibir datos de forma constante de manera de monitorear el comportamiento un elemento, permite agregar una herramienta más que puede ayudar a la toma de decisiones al reducir la probabilidad de que se consideren datos erróneos como un evento anómalo considerando niveles normales.

4.3 Comparación con trabajos anteriores.

Debido a las características de este proyecto, no se ha podido encontrar un punto de comparación. Si bien existen herramientas de monitoreo, estas no se ajustan a situaciones donde los datos puedan ser insertados de diferentes fuentes ni que el sistema sea capaz de identificar si los cambios en los datos ingresados representan un evento que se debe considerar.

Al finalizar este proyecto, la herramienta aún se considera como prototipo ya que aún se puede refinar y agregar nuevas funcionalidades a la aplicación, esto se detallará en el capítulo 5 de este documento.

CAPITULO 5: Conclusiones

5.1 Conclusión

La construcción de esta plataforma, permite la reutilización del código en particular del componente API ya que este puede ser adaptado según otras necesidades, especialmente en aplicaciones donde se requiera utilizar mapas para generar zonas de calor, georreferencia y cálculo de datos en general utilizando diferentes librerías.

También la construcción de una aplicación tipo Web Mashup permite mostrar datos de diferentes fuentes (APIs) existentes, de manera de hacer uso de esta y centralizar la muestra de datos en una plataforma.

El resultado de la construcción de la aplicación fue que es posible adaptar diferentes tecnologías para lograr la captura de datos, mostrarlos y ofrecer una herramienta más al usuario en la toma de decisiones ante cualquier tipo de evento relacionado con la toma de una métrica especifica. Además, ya finalizado en proyecto fue posible implementar un algoritmo complejo que permite la detección de eventos donde la medición de datos tenga un cambio brusco e identificar si este cambio es un evento real o posible error de la fuente de datos, como una des calibración temporal de un sensor o ruido en el dato.

El hecho de que esta aplicación fuese desarrollada en capas y utilizando diferentes tecnologías permite crear una herramienta adaptable donde se puede modificar a diferentes situaciones como, por ejemplo:

-Hospitales

-Datos de Censo

-Aplicaciones de georreferencia

5.2 Problemas abiertos

Los problemas abiertos que quedaron al final del desarrollo de este proyecto tienen relación con investigación futura y la agregación de funcionalidades a la plataforma.

Como parte de la investigación y desarrollo de algoritmos para la detección temprana de catástrofes, se debe investigar una solución o algoritmo que pueda ser implementado para cumplir con este propósito, es decir, agregar un componente que permita acelerar el proceso de toma de decisiones de los interesados al tener modelos predictivos para el comportamiento de los datos ya que hasta la finalización de este documento solo se alcanzó a implementar un sistema que permitiera detectar los cambios abruptos en la recolección de datos.

Ya que este proyecto aún se encuentra en la etapa de prototipo, también se recomienda buscar soluciones que permitan llevar la aplicación a producción sin mayores complicaciones. Esto debido a que todas las pruebas se realizaron en servidores y máquinas de prueba por lo que también se hace necesario probar la aplicación con grandes volúmenes de datos emulando el escenario real de un servidor en producción.

5.3 Trabajo futuro

Según lo mencionado anteriormente en los problemas abiertos, a continuación, se lista el trabajo futuro para la aplicación.

- Desarrollo e investigación de modelos predictivos en modo de ayuda a la toma de decisiones
- Llevar a producción la aplicación en entorno real y con volúmenes de datos reales.
- Realización de pruebas unitarias para todas las funciones de la aplicación web.

5.4 Referencias

- Romina Torres, Nelly Bencomo, Hernan Astudillo, "Addressing the QoS drift in specification models of self-adaptive service-based systems", , vol. 00, no. , pp. 28-34, 2013, doi:10.1109/RAISE.2013.6615201
- Valdés-Pineda, R., Valdés, J. B., García-Chevesich, P. 2017. Mudflow Modeling in the Copiapó Basin, Chile. Ingeniería del agua, 21(2), 135-152. https://doi.org/10.4995/Ia.2017.7366
- Farhan Shafiq, Kamran Ahsan, “An ICT based Early Warning System for Flood Disasters in Pakistan” Vol.3(9) pp: 108-118, September 2014 ISSN 227-2502