UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

SISTEMA DE ALERTA DE INDICADORES MEDIOAMBIENTALES EN UNA TERMOELÉCTRICA

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

HERNÁN ALEJANDRO FIERRO SILVA

PROFESOR GUÍA:
SERGIO FABIÁN OCHOA DELORENZI

MIEMBROS DE LA COMISIÓN:
ALEJANDRO HEVIA ANGULO
JAVIER ALEJANDRO BUSTOS JIMÉNEZ

SANTIAGO DE CHILE
2013

Resumen

La normativa ambiental vigente en Chile exige a las empresas que emiten contaminantes del aire (por ejemplo, SO2 o NOx), mantener sus emisiones bajo niveles definidos. Desde octubre de 2010 opera la *SMA* o *Superintendencia del Medio Ambiente*, encargada de velar por el cumplimiento de dicha normativa y aplicar las sanciones que correspondan de acuerdo a la legislación. Dentro de este contexto, la generadora termoeléctrica E-CL, ubicada en Tocopilla y Mejillones ha solicitado a ASYT, empresa consultora de medioambiente, el desarrollo del software EMIS (Environmental Management Information System), el cual tiene por objetivo ser un apoyo a la gestión del cumplimiento de las exigencias ambientales de E-CL.

El diseño de EMIS contempla la construcción de diversos módulos que permiten: consolidar los datos de emisiones de contaminantes de diversas fuentes, visualizar en tiempo real las emisiones, generar modelos de dispersión y planificar la realización de tareas asignadas por la autoridad, entre otros. El objetivo de este trabajo de título fue analizar, diseñar e implementar el Módulo de Alertas de EMIS, el cual permite a sus usuarios conocer en tiempo real el estado de cumplimiento de la normativa, definir eventos críticos y ser notificados en el caso de que ocurran. El trabajo trajo consigo diversos desafíos como: acoplarse al desarrollo existente de EMIS, definir un diseño software que otorgue un buen nivel de servicio, diseñar interfaces de usuario que logren aceptación por el usuario final y diseñar las alertas que sean útiles para la gestión ambiental de E-CL.

Como primera etapa, se analizó el problema haciendo un levantamiento de la situación actual respecto al estado de avance de EMIS y al alcance de la normativa ambiental vigente. Teniendo esta información se definieron requisitos de aspectos clave del sistema, por ejemplo: cómo se define el estado de cumplimiento, cómo se define la criticidad de eventos, con qué frecuencia se actualiza el estado de cumplimiento, cuándo y a qué usuarios se debe notificar, entre otros.

Luego, se definió una arquitectura de software, modelo de datos e interfaz de usuario que cumplan con los requisitos definidos y le permitan al sistema ser extensible y mantenible. Posteriormente, se implementó la solución teniendo como base *Spring*, *Java EE* y *PostgreSQL*, además de la utilización de bibliotecas externas, cuidadosamente seleccionadas para implementar funcionalidades gráficas y de programación de tareas. El sistema fue evaluado en el aspecto técnico realizando mediciones de performance y corroborando la validez de los cálculos realizados. También fue validado por usuarios finales mediante presentaciones y reuniones que fueron llevadas a cabo en las instalaciones de E-CL.

Finalmente, el sistema fue instalado en los servidores de E-CL como prototipo funcional, lográndose una solución que facilita el cumplimiento con la autoridad y permite enfocarse en los ámbitos importantes de la gestión ambiental y toma de decisiones. Es factible potenciar esta herramienta para ser utilizada en el contexto de otro tipo de normativas, como por ejemplo de aguas, ruido, suelos, flora y fauna, entre otros.

Agradecimientos

A Dios por su ayuda, compañía y dirección a lo largo de mi vida, en particular en mis estudios.

A mi familia por la formación, apoyo incondicional y ánimo que he recibido por parte de ella a lo largo de estos años, que me han permitido poder disfrutar mis estudios.

A Katherine Miranda por su compañía, amor y por haberme sabido soportar, escuchar y aconsejar durante toda la carrera, haciendo los años más llevaderos.

A mis amigos y compañeros de universidad por los buenos momentos vividos, que sin duda se recordarán por muchos años.

Tabla de Contenido

1. Introducción

Dentro de la industria chilena existe una gran variedad de actividades que son afectas a normativa ambiental. Se entiende por normativa ambiental las leyes, decretos o resoluciones que regulan el funcionamiento de alguna actividad industrial, en relación a su impacto en el medio ambiente. La porción de la industria se ve más regulada por normativas corresponde a:

- Mineras
- Pesqueras
- Centrales termoeléctricas
- Plantas faenadoras de ganado
- Criaderos de animales
- Refinerías
- Centrales hidroeléctricas
- Cementeras y hormigoneras

Actualmente, existe legislación medioambiental vigente que exige a las industrias que producen contaminantes atmosféricos, medir y regular sus emisiones para que se mantengan dentro de un rango aceptable. Por otra parte, dentro del proceso de otorgamiento de los permisos necesarios para la construcción y operación de industrias se deben llevar a cabo una serie de estudios que producen como resultado exigencias específicas, llamadas *RCAs* o *Resoluciones de Calificación Ambiental*. Las *RCAs* usualmente contienen exigencias como:

- Labores de construcción, por ejemplo: construcción de rejas, estanques, entre otros.
- Planes de monitoreo, por ejemplo: monitoreo de napas subterráneas
- Regulación de niveles de emisión de contaminantes, por ejemplo: no superar la emisión de 16kg. de CO por hora.

El proceso de fiscalización del cumplimiento de las regulaciones y acuerdos ha sido por mucho tiempo inefectivo. Sin embargo, el 06 de octubre de 2010 comenzó a operar la *Superintendencia de Medio Ambiente (SMA*[1]), cuyos objetivos son promover y verificar el cumplimiento de la normativa ambiental a través de la fiscalización y sanción de los incumplimientos. No obstante sus atribuciones actuales, la SMA podrá hacer uso de ellas sólo una vez que sea conformado el primer Tribunal Ambiental, el cual se encuentra actualmente en proceso de creación. Dada esta situación es interés de muchas empresas estar oportunamente preparados para una posible, pronta y profunda, fiscalización.

1.1. Problema Resuelto

El problema que se abordó en esta memoria fue el diseño e implementación del *Módulo de Alertas* del software *EMIS*. Al ser este módulo parte de un sistema mayor, el trabajo fue llevado a cabo considerando su interacción con otras componentes de *EMIS* y que su interfaz y modo de uso debe ser consistente con los módulos ya implementados.

[1]http://www.sma.gob.cl/

Los usuarios que reciben las alertas son típicamente personas con cargos gerenciales o jefes de plantas en quienes reside la responsabilidad de mantener las emisiones controladas y las evidencias de cumplimiento de compromisos publicadas al día, por lo que el diseño del módulo tuvo que realizarse, tal que la definición de las alertas, el despliegue de las mismas y el tiempo de respuesta permitan a los usuarios realizar su gestión de manera efectiva.

Para las *alertas por sobrepaso de normativa* el diseño e implementación del sistema debió soportar el manejo de múltiples indicadores actualizados con diferentes frecuencias, el cálculo del estado de cumplimiento de los mismos en comparación a la normativa ambiental vigente y el envío de notificaciones a los usuarios correspondientes, en el caso de detectarse niveles peligrosos.

Del mismo modo, en el diseño e implementación de EMIS se consideró el envío de notificaciones únicamente mediante correo electrónico, pero definiendo un diseño que permita incorporar otros mecanismos en el futuro.

1.2. Descripción del Software EMIS

ASYT[2] es una empresa consultora de medioambiente constituida por un equipo de trabajo multidisciplinario, compuesto por ingenieros en: minas, alimentos, medioambiente, industriales y en computación. La empresa ofrece diversos tipos de servicios, tales como: estudios de impacto ambiental, auditorías de cumplimiento de normas ambientales y cuantificación de emisiones e implementación de redes de monitoreo, entre otros.

Recientemente la empresa ha organizado su área de desarrollo de software y decidido crear un software que permita a las empresas de la industria chilena mejorar su gestión ambiental, utilizando soluciones tecnológicas. El software es llamado *EMIS: Environmental Management Information System*, el cual tiene como mercado objetivo empresas que deben manejar un gran número de exigencias ambientales y que poseen un gran número de fuentes de emisión y monitoreo de contaminantes.

EMIS se encuentra actualmente en etapa de diseño y desarrollo, teniendo como objetivo la implementación de los siguientes módulos:

- **El Módulo de Datos** permite consolidar y homogenizar el acceso a datos que provienen de diversas fuentes, ubicaciones y sistemas.
- **El Módulo SMA** permite gestionar las *Resoluciones de Calificación Ambiental (RCAs)*[3] emitidas por las autoridades, dando la posibilidad de planificar mediante actividades concretas cómo será el cumplimiento de cada exigencia asociada a cada RCA.
- **El Módulo Plan de Monitoreo** apoya el cumplimiento de tareas específicas que han sido asignadas por las autoridades. Se maneja en el sistema como cartas Gantt anuales que poseen actividades repetibles a lo largo de los meses.
- **El Módulo Localización** permite ver geolocalizadamente y en tiempo real los datos que están ingresando a EMIS mediante el Módulo de Datos.

[2] http://www.asyt.cl
[3] http://www.kdm.cl/index.php/marcolegal/32-marcolegal/29-queesunaresoluciondecalificacionambiental.html

- **El Módulo Mapas** permite generar modelos de dispersión de contaminantes, utilizando los datos consolidados que llegan mediante el Módulo de Datos.
- **Módulo Dashboard** permite al usuario disponer de un panel de control personal, donde puede visualizar permanentemente los indicadores que aplican a su gestión.
- **El Módulo Alertas** permite al usuario conocer el estado de cada indicador respecto a la normativa vigente, teniendo notificaciones oportunas en caso de detectarse niveles críticos.

1.3. Cliente Asociado a esta Memoria

Dentro del contexto de la legislación ambiental vigente, la central termoeléctrica *E-CL*[4] perteneciente al grupo *International Power GDF Suez*, ubicada en Tocopilla y Mejillones ha solicitado a *ASYT* la implementación de *EMIS*, con el fin de mejorar y modernizar su gestión medioambiental.

E-CL posee dos plantas generadoras a carbón, dos a gas y una a diesel. Producto de su operación E-CL emite contaminantes a la atmósfera como: NOx, SO2, CO y realiza descargas de fluidos al mar. Es por esto que la empresa realiza mediciones en sus chimeneas, en diversos puntos en el mar y en diversas ubicaciones de las ciudades aledañas.

En total E-CL posee alrededor de 150 estaciones de monitoreo de contaminantes, que están midiendo alrededor de 400 indicadores distintos, muchos de ellos asociados a normativas. En cuanto a RCAs E-CL actualmente posee 15 RCAs con, en promedio, 50 exigencias cada una. Muchas de estas exigencias derivan en planes específicos de monitoreo.

1.4. Justificación del Trabajo

El desarrollo del sistema de alertas trajo consigo diversos desafíos de ingeniería que fueron resueltos en el marco de esta memoria. Éstos se enumeran a continuación.

1. **Carga del Sistema.** Con el fin de asegurar de compatibilizar las capacidades de la infraestructura tecnológica de E-CL con la implementación de la solución, fue necesario analizar y predecir cuánta será la carga de la máquina donde correrá el sistema y en base a eso tomar decisiones en cuanto a su arquitectura de software.
2. **Usabilidad del Sistema.** Con el fin de que el sistema sea usable y logre una buena aceptación por parte del usuario, fue necesario diseñar cuidadosamente y luego validar las interfaces de usuario.
3. **Nivel de Control del Usuario.** Se debió diseñar qué libertades y qué opciones tiene el usuario dentro del sistema. Esto tal que las tareas de mantención y configuración por el usuario sean posibles de realizar, tengan sentido de acuerdo a sus necesidades y no provoquen un comportamiento inesperado del sistema.
4. **Acoplamiento con la Solución Completa.** El sistema de alertas es sólo una parte de una solución mayor, por lo que este sistema debió acoplarse con otros módulos construidos por otros desarrolladores. Esto implicó analizar la arquitectura existente para determinar

[4]http://www.e-cl.cl

qué servicios fueron necesarios, cómo y cuáles serían los módulos con los que se interactuaría, etc.

5. **Niveles de Servicio.** Dado que se espera que el módulo de alertas sea un apoyo a la gestión de eventos críticos de E-CL, el software debe garantizar la generación y entrega de alertas en un tiempo pre-establecido. En otras palabras, se tuvo que garantizar ciertos niveles de servicio asociados a la confiabilidad y performance del sistema.
6. **Robustez y Confiabilidad.** Al ser las alertas sobre temas medioambientales algo crítico en la gestión de la termoeléctrica, fue necesario asegurar que no existirán falsos positivos, notificaciones duplicadas, alertas que no se levantaron oportunamente o que simplemente no se levantaron. En este sentido se espera que el producto sea lo suficientemente robusto, como para entregar la confiabilidad que los usuarios de este software requieren.

1.5. Objetivos de la Memoria

El objetivo general de este trabajo fue diseñar e implementar un sistema de software para administrar alertas de cumplimiento de normativa respecto a indicadores ambientales medidos por E-CL. Este sistema de alertas debe lograr procesar la totalidad de los datos de mediciones de indicadores afectos a normativa de contaminantes atmosféricos en tiempo real y notificar al usuario oportuna y eficientemente cada evento significativo. Dicha solución debe ser suficientemente confiable y robusta como para que los usuarios finales lleguen no sólo a utilizarla, sino a percibirla como parte fundamental de su labor de gestión.

Los objetivos específicos definidos para alcanzar el objetivo general fueron los siguientes:

- Construir un subsistema que permita decidir cuándo una alerta debe ser notificada a los usuarios correspondientes.
- Diseñar e implementar un sistema de administración de alertas, es decir, crear, editar, eliminar y configurar alertas.
- Diseñar e implementar un sistema de despliegue de alertas, que soporte diversos dispositivos y servicios.

2. Análisis del Problema

El problema fue analizado en primer lugar, identificando los requisitos de usuario, luego haciendo una recopilación de información (del cliente y de la normativa actual) para finalmente, definir los requisitos de software que responden a las necesidades del cliente.

2.1. Requisitos de Usuario

Los requisitos de usuario identificados fueron especificados tomando como referencia el formato recomendado por el Estándar de Ingeniería de Software de la Agencia Espacial Europea (ESA) [1], cuyo template incluye los siguientes atributos de un requisito: identificador, nombre, necesidad, fuente, prioridad y estabilidad.

A continuación se listan los principales requisitos que se definieron. Para ver el detalle, consultar *Anexo A. Requisitos de* Usuario.

- ***Alertas para Módulo de Datos*.** Corresponde a la capacidad del módulo para lanzar alertas relacionadas a parámetros medidos en el módulo de Datos.
- ***Alertas de Acuerdo a Normativas*.** Las alertas que despliegue el sistema deben ser referidas a la normativa vigente.
- **Histórico de Alertas.** El sistema debe registrar las alertas que han sido levantadas para ser consultadas en el futuro.
- **Criticidad de Alertas.** El sistema debe discriminar entre al menos tres niveles de alertas: leves, medianas y graves.
- ***Notificación por correo electrónico*.** Corresponde a la opción que tiene el usuario administrador de definir por cada alerta si será desplegada por correo electrónico o no.
- ***Notificación en interfaz web*.** Cada alerta debe poder ser desplegada en la interfaz web del sistema una vez que el usuario correspondiente inicia sesión.
- ***Notificaciones a usuarios*.** El usuario administrador podrá seleccionar uno o más usuarios a los que se les notificará una vez levantada la alerta.
- ***Tiempo envío notificaciones*.** El sistema deberá hacer envío de las notificaciones como máximo un minuto después de levantada la alerta.

2.2. Levantamiento de la Situación Actual

Para realizar el levantamiento de la situación actual, en primer lugar, se recopiló la normativa que debe considerarse y cómo ésta aplica a E-CL. Luego se analizó y describió el estado de implementación del software.

2.2.1. Recopilación de Normativa Vigente y su Aplicabilidad a E-CL

Con el fin de tener una estimación preliminar de la complejidad y volumen de datos a procesar por el módulo de alertas, se realizó en conjunto con José Gonzales, experto en redes de monitoreo, una recopilación y análisis de la normativa vigente y su aplicabilidad a la realidad de la operación de las centrales de E-CL.

En primer lugar, se obtuvieron los documentos que definen la normativa vigente (usualmente en *Decretos Supremos)* seleccionando los que aplican a contaminantes emitidos por E-CL, luego éstos fueron analizados y categorizados según:

- **Nombre.** Hace referencia al contaminante y tipo de norma. Para el caso de centrales de generación también se distingue entre tipo de combustible utilizado: líquido (diesel), sólido (carbón, petcoke) o Gas.
- **Clasificación.** Puede referirse a la *Calidad de Aire* o *Emisiones*. *Calidad de Aire* se refiere a mediciones de contaminantes que son hechas en los pueblos o ciudades cercanas a las centrales con el fin de cuantificar el impacto que causa la presencia de E-CL. *Emisiones* corresponde a las mediciones que son hechas directamente en las fuentes de emisión (i.e. chimeneas).
- **Período.** Corresponde al horizonte temporal que abarca la normativa.
- **Agregación.** Espacio de tiempo en que los datos deben ser agrupados.
- **Estadística**. Procesamiento a realizar con la Agregación y Período indicado.
- **Límite.** Valor máximo permitido al aplicar la Estadística en el Período indicado.

A modo de ejemplo, el *Decreto Supremo Nº115/02 del Ministerio Secretaria General de la Presidencia* establece que:

> *"Se considerará sobrepasada la norma primaria de **calidad de aire** para **monóxido de carbono** como concentración de **1 hora**, cuando el **promedio aritmético de tres años sucesivos**, del **percentil 99 de los máximos diarios** de concentración de 1 hora registrados durante un año calendario, fuere mayor o igual a **30000 µg/m3N**."*

Al categorizarlo según se mencionó anteriormente, se obtiene:

- **Nombre.** CO - Horario.
- **Clasificación.** Calidad del Aire.
- **Período.** 3 Años.
- **Agregación.** 1 Hora.
- **Estadística**. Percentil 99 de los máximos diarios.
- **Límite.** 30000 µg/m3N

El resultado obtenido luego de revisar toda la normativa vigente para los contaminantes que E-CL emite es el siguiente:

Tabla 1 - Recopilación de normativa.

Nombre	Clasificación	Periodo	Agregación	Estadística	Límite
MP10 Diaria	Calidad Aire	1 Año	1 Día	Percentil 98 de los valores diarios	150 ug/m3N

O3 Horaria	Calidad Aire	3 Años	1 Hora	Percentil 99 de los máximos diarios	62 ug/m3N
SO2 Diaria	Calidad Aire	3 Años	1 Día	Promedio aritmético de los valores diarios	80 ug/m3N
SO2 Anual	Calidad Aire	3 Años	1 Día	Percentil 99 de los valores diarios	250 ug/m3N
NO2 Anual	Calidad Aire	3 Años	1 Día	Promedio aritmético de los valores diarios	100 ug/m3N
NO2 Horaria	Calidad Aire	3 Años	1 Hora	Percentil 99 de los máximos diarios	400 ug/m3N
CO Horaria	Calidad Aire	3 Años	1 Hora	Percentil 99 de los máximos diarios	3000 ug/m3N
CO 8Horaria	Calidad Aire	1 Año	8 Horas	Percentil 99 de los máximos diarios	1000 ug/m3N
MP2.5 Diaria	Calidad Aire	1 Año	1 Día	Percentil 98 de los valores diarios	50 ug/m3N
MP2.5 Anual	Calidad Aire	3 Años	1 Año	Promedio de los valores de cada año	20 ug/m3N
MP Combust Sólido	Emisiones	1 Hora	1 Hora	Promedio Horario	50 mg/m3N
MP Combust Líquido	Emisiones	1 Hora	1 Hora	Promedio Horario	30 mg/m3N
SO2 Combust Sólido	Emisiones	1 Hora	1 Hora	Promedio Horario	400 mg/m3N
SO2 Combust Líquido	Emisiones	1 Hora	1 Hora	Promedio Horario	30 mg/m3N
NOx Combust Sólido	Emisiones	1 Hora	1 Hora	Promedio Horario	500 mg/m3N
NOx Combust Líquido	Emisiones	1 Hora	1 Hora	Promedio Horario	200 mg/m3N
NOx Combust Gas	Emisiones	1 Hora	1 Hora	Promedio Horario	50 mg/m3N

Además, se cuantificaron los puntos de medición que posee E-CL de cada contaminante y clasificación, obteniendo:

Tabla 2 - Resumen de estaciones por contaminante y clasificación.

Contaminante	Clasificación	Estaciones en que se mide el contaminante
SO2	Calidad Aire	SSITE, JJLAT, FERR, ECENTRO, EMEJ
CO	Calidad Aire	SSITE, JJLAT, FERR, ECENTRO, ESUR
O3	Calidad Aire	SSITE, JJLAT, FERR, ECENTRO, ESUR
NO2	Calidad Aire	SSITE, JJLAT, FERR, ECENTRO, ESUR
MP10	Calidad Aire	SSITE, JJLAT, FERR, ECENTRO, ESUR
MP25	Calidad Aire	SSITE, JJLAT, FERR, ECENTRO, ESUR
NOx	Emisiones	CH1213, CH1415, CTM12, CTA, CTH, CH10, CH11, CTM3

MP	Emisiones	CH1213, CH1415, CTM12, CTA, CTH, CH10, CH11,
SO2	Emisiones	CH1213, CH1415, CTM12, CTA, CTH, CH10, CH11,

De este análisis se pudo concluir que el diseño del módulo de alertas para E-CL debe permitir soportar cálculos con horizontes temporales que van desde 1 hora hasta 3 años, con agregaciones desde 1 hora hasta 1 año. Debe manejar 16 normativas diferentes, para 6 tipos de contaminantes que son medidos en 14 estaciones de monitoreo diferentes. Además, se pudo apreciar que las funciones de cálculo que se utilizan para definir las normativas son reducidas. Sólo se utilizan promedios aritméticos, máximos diarios y percentiles.

2.2.2. Estado Actual de Implementación del Software y su Arquitectura

A continuación, se describe el estado de avance de los módulos del sistema al momento de comenzar el trabajo de memoria y las tecnologías que fueron utilizadas para su desarrollo (Ilustración 1). Para más detalle referirse al *Anexo D. Tecnologías Utilizadas en* EMIS. En cuanto a software el avance encontrado fue el siguiente:

- Desarrollo de la aplicación realizado utilizando Java 6 y Spring Framework 3.1.
- Una base de datos *PostgreSQL* configurada para ser accedida mediante el uso de *Hibernate y Java Peristence Unit.*
- Un servicio de recolección de datos encargado de obtener las mediciones periódicamente desde un *WebService SOAP* servido por *E-CL*. El servicio es programado mediante el uso del gestionador de tareas *Quartz*.
- Los módulos de Datos, SMA, Plan de Monitoreo y Localización, implementados y en etapa de testing.
- Una interfaz web implementada para los módulos antes mencionados, utilizando *Java Server Faces* 2 para la generación de páginas HTML, *HighCharts* para el despliegue de gráficos de datos y *Google Maps API V3* para el despliegue georeferenciado.

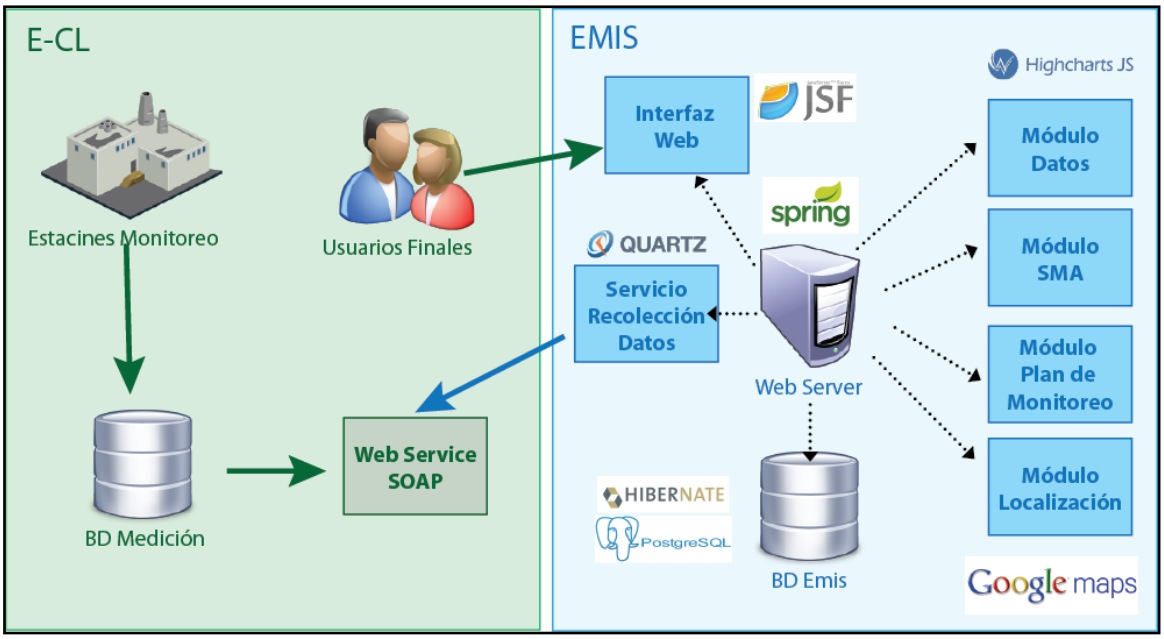

Ilustración 1. Arquitectura al momento de comenzar el trabajo de memoria.

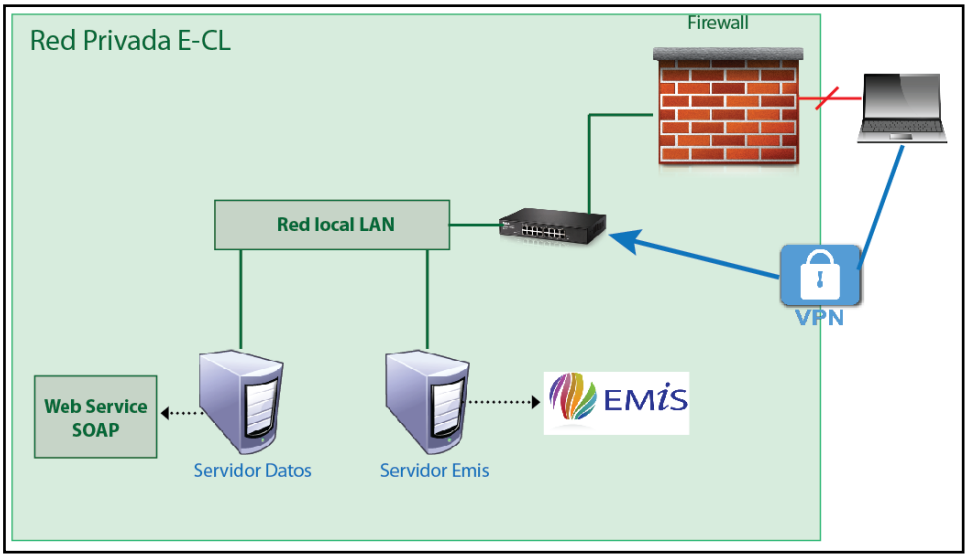

Ilustración 2. Arquitectura Física

En cuanto a arquitectura física se identificó (ver Ilustración 2. Arquitectura Física):

- Una máquina instalada en la red privada de E-CL, que aloja tanto el servidor de aplicaciones *Java Tomcat*, como el motor de base de datos *PostgreSQL*.
- Otra máquina, también dentro de la red privada de E-CL, que contiene el *WebService* que EMIS consume.
- La red privada de E-CL (y por lo tanto EMIS) no es accesible desde el exterior directamente. Esto es posible únicamente mediante una VPN.

2.3. Definición de Requisitos de Software

Luego de tener los requisitos por parte del usuario y una estimación de la cantidad de normativa a considerar, se definieron los requisitos de software asociados al sistema, los cuales pueden ser vistos en detalle en el Anexo B. Requisitos de Software. A continuación se describen las principales decisiones que fueron tomadas asociadas al diseño de los requisitos.

2.3.1. Cálculos Configurados por Usuario v/s Cálculos Preconfigurados

Inicialmente se consideró la idea de que el usuario pudiese determinar manualmente y para cada indicador, cómo se realizaría el cálculo de cada normativa. El sistema debería ofrecerle al usuario un abanico de opciones (agregación, funciones estadísticas, etc.) que le permitiese definir reglas de cálculo que se ajusten a la normativa actual. Sin embargo, luego del trabajo de recopilación de normativa vigente, se pudo apreciar que esta alternativa es contraproducente por razones que son detalladas a continuación.

En primer lugar, se requiere un conocimiento altamente experto por parte del cliente tanto en el aspecto medioambiental (conocer la normativa, identificar cuál es aplicable a cuáles estaciones de E-CL, diferenciarlas según clasificación, combustible, etc.), como en el aspecto matemático (analizar cada normativa y definir cómo debe ser llevado a cabo el cálculo para ella).

En segundo lugar, en relación al punto anterior, tanto en la creación como en la mantención de los cálculos, existe una posibilidad de que lo definido no se ajuste a lo que la normativa establece, o bien, que no se haya aplicado a los indicadores que corresponden. Esto podría provocar falsos positivos y una pérdida de credibilidad del sistema de alertas.

Finalmente, considerando que no se espera que la normativa vigente sufra modificaciones en el corto o mediano plazo, se decidió que las reglas de cálculo para cada indicador vendrían precargadas en el sistema, lo que le simplifica la tarea al usuario y se logra un mayor grado de credibilidad de las alertas.

2.3.2. Configuración de Criticidad

Dado que el sistema debe soportar alertas de tres niveles (leves, medianas y graves) fue necesario definir un mecanismo que permita diferenciar la criticidad entre estos niveles. Se definió la "carga" de un par (indicador, normativa), como el porcentaje del cálculo de la normativa para el indicador, respecto al máximo establecido. Así, por ejemplo, la carga para el SO2 en la estación FERR, respecto de la normativa anual de calidad de aire es de un 70%. Una carga mayor al 100% indica que se ha sobrepasado la normativa.

La diferenciación entre alertas se definió mediante tres valores que representan porcentajes, siendo cada uno de ellos el nivel en que comienza cada nivel de criticidad. Se consideró que por defecto vendría cargado en el sistema para todos los pares (indicador, normativa) la configuración: 70% para Leve, 80% para Mediano, 90% para Grave.

Luego de realizar una reunión con el cliente, se definió que la configuración de niveles de criticidad por defecto podría ser sobrescrita por un usuario administrador de EMIS, esto debido a que para E-CL algunos indicadores requieren más atención que otros.

2.3.3. Reglas de Notificación

Un aspecto importante a analizar fue el comportamiento del sistema de notificación al usuario. Se definió que los usuarios administradores de EMIS pueden definir qué usuarios serán notificados al ocurrir alertas Leves, Medianas o Graves y en los siguientes niveles:

- **Nivel Global**. Considera todas las alertas que se han levantado en el sistema. Por ejemplo, enviar a usuario Patricio Lillo todas las alertas Graves que ocurran.
- **Nivel Estación.** Considera las alertas que han sido levantas para cualquiera de los indicadores que están siendo medidos en una estación de monitoreo específica. Por ejemplo, enviar a usuario Patricio Lillo las alertas Medianas que ocurran en la estación SSITE.
- **Nivel Normativa.** Considera las alertas que se han levantado respecto a una normativa específica, sin importar su ubicación. Por ejemplo, enviar a usuario Patricio Lillo las alertas Leves que tengan relación con la normativa de SO2 anual de calidad del aire.

2.3.4. Frecuencia de Cálculo y Duplicidad de Alertas

De la recopilación y análisis de la normativa se concluyó que la frecuencia con la que se deben detectar alertas en el sistema depende de cada normativa y corresponde a la agregación de la misma (ver 2.2.1). Así, por ejemplo para el indicador SO2 que se está midiendo en la estación SSITE, para la normativa SO2 Diaria de Calidad del Aire, la frecuencia de cálculo será 1 Hora.

Se definió también, que el sistema no debe enviar alertas que se consideren duplicadas, de acuerdo a los casos que se detallan a continuación. Asumiendo que se ha levantado una alerta para un par (indicador, normativa), se considera una alerta duplicada si:

- El servicio de recolección de datos ha sufrido un retraso y no hay nuevos datos para realizar el cálculo asociado una vez que (según la frecuencia de cálculo) corresponde actualizar el estado del indicador.
- Corresponde nuevamente actualizar el estado del indicador y se detecta una alerta del mismo nivel que la anterior. Así, por ejemplo, si la alerta inicial era del tipo Leve y se detecta nuevamente una Leve, ésta no es informada, pues se asume que corresponde al mismo evento que el anterior.

2.3.5. Normativas con Periodos Calendarios

De acuerdo a lo recogido en el análisis de la normativa, se pudo apreciar que las normativas que corresponden a la clasificación *Calidad del Aire*, están definidas utilizando períodos en años calendarios, es decir en años contando desde el 1ero de enero al 31 de diciembre. Esto plantea una limitante al momento de definir alertas oportunas, que sean relevantes y que permitan tomar acciones de manera preventiva. El problema radica en que si se realizan los cálculos de carga de cada indicador utilizando periodos calendarios, los resultados obtenidos se vuelven irrelevantes al principio del año, y por el contrario, irremediables al final del año.

Para ejemplificar, tomemos la normativa de *MP10 Diaria de Calidad del Aire*, que establece que el percentil 98 de los promedios diarios de un año calendario no puede superar los 150 ug/m3N. Ahora supongamos que es 1ero de febrero y se detecta un nivel de carga peligroso, digamos 90%.

La alerta en este caso es oportuna puesto que es posible remediar la situación tal que no se caiga en un incumplimiento de la normativa puesto que le restan al rededor 330 días al año, y es por esta misma razón que la alerta es irrelevante puesto que no indica una verdadera situación crítica y la cantidad de datos utilizados no es representativa. Por otra parte, una alerta en el mes de diciembre es relevante, puesto que realmente indica una situación crítica, pero eventualmente no oportuna, puesto que podría no ser factible evitar un incumplimiento. Se desea que las alertas levantadas sean en cuanto a oportunas y relevantes, lo más homogéneas posible, sin que las cualidades anteriormente señaladas dependan del día del año.

Luego de una reunión realizada con el cliente, se resolvió utilizar años móviles para los cálculos que se efectúen, es decir, que cada vez que se realice un cálculo de normativa para algún indicador, se utilizará un periodo móvil que termina en el último valor medido del indicador. Esto cambia la semántica que una alerta tenía inicialmente y le quita precisión, sin embargo, a juicio del cliente, esto le permite tener alertas más útiles en su gestión diaria, puesto que cumplir siempre con la normativa respecto a períodos móviles implica cumplirlo para períodos calendarios (notar que los períodos calendarios son un caso particular de los móviles).

3. Diseño del Módulo de Alertas

En las siguientes secciones se presenta el diseño del sistema indicando su arquitectura física, de software, modelo de datos, interfaz del usuario y aspectos de implementación.

3.1. Arquitectura Física

La arquitectura física del módulo de alertas no difiere de la analizada al momento de hacer el levantamiento inicial (ver 2.2.2), puesto que en las etapas de diseño, implementación y testing no se requirieron equipos o capacidades extra a las existentes.

3.2. Arquitectura de Software

En esta sección se describe el diseño de la arquitectura del módulo desarrollado, presentando su diseño en una vista de capas, explicando en detalle los componentes diseñados e indicando cómo estos interactúan desde un punto de vista temporal.

3.2.1. Vista de Capas

El sistema fue diseñado pensando en una estructura de capas, pudiendo cada capa interactuar únicamente con las que la preceden. El detalle de cada componente se puede encontrar en la sección siguiente. A continuación se detalla cada capa (Ilustración 3):

- **Datos.** Esta capa es la encargada de contener las entidades, mapeándolas a tablas en la base de datos. Se encarga de proveer acceso a las entidades y generación de consultas a las capas superiores.
- **Servicios.** Aquí se aloja la lógica de la solución. Esta capa está encargada de realizar todos los cálculos de normativa, programar su frecuencia, levantar las alertas y notificar a los usuarios correspondientes. Esta capa también se encarga de la lógica de las configuraciones que el usuario debe hacer (niveles de criticidad y notificaciones).
- **Presentación.** Esta capa está encargada del despliegue y la interacción con el usuario, a través de los *Beans*, páginas *JSF* y gráficos.

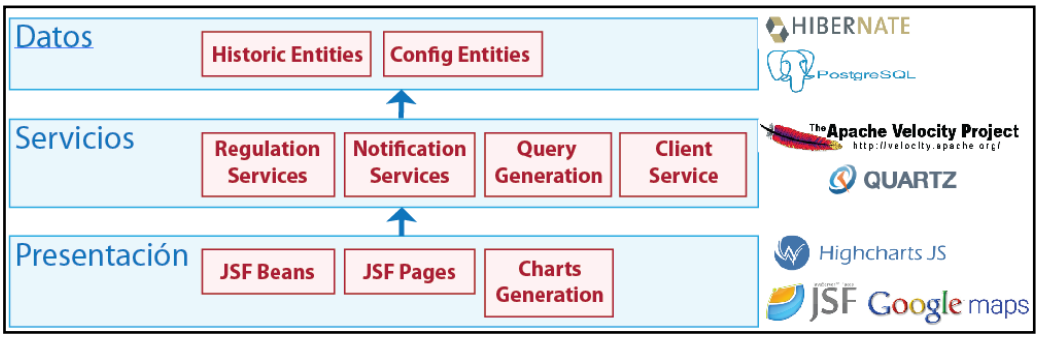

Ilustración 3. Vista de capas

3.2.2. Componentes

En esta sección se describe en detalle los componentes y subcomponentes de software de cada capa de la arquitectura, indicando su responsabilidad e interacción con otros componentes. En la

Ilustración 4 se puede apreciar un diagrama que resume lo antes mencionado. Tanto las capas, como los componentes de las mismas, fueron diseñados para tener las responsabilidades lo más desacopladas posible, fomentando una buena mantenibilidad del sistema y reusabilidad de los componentes.

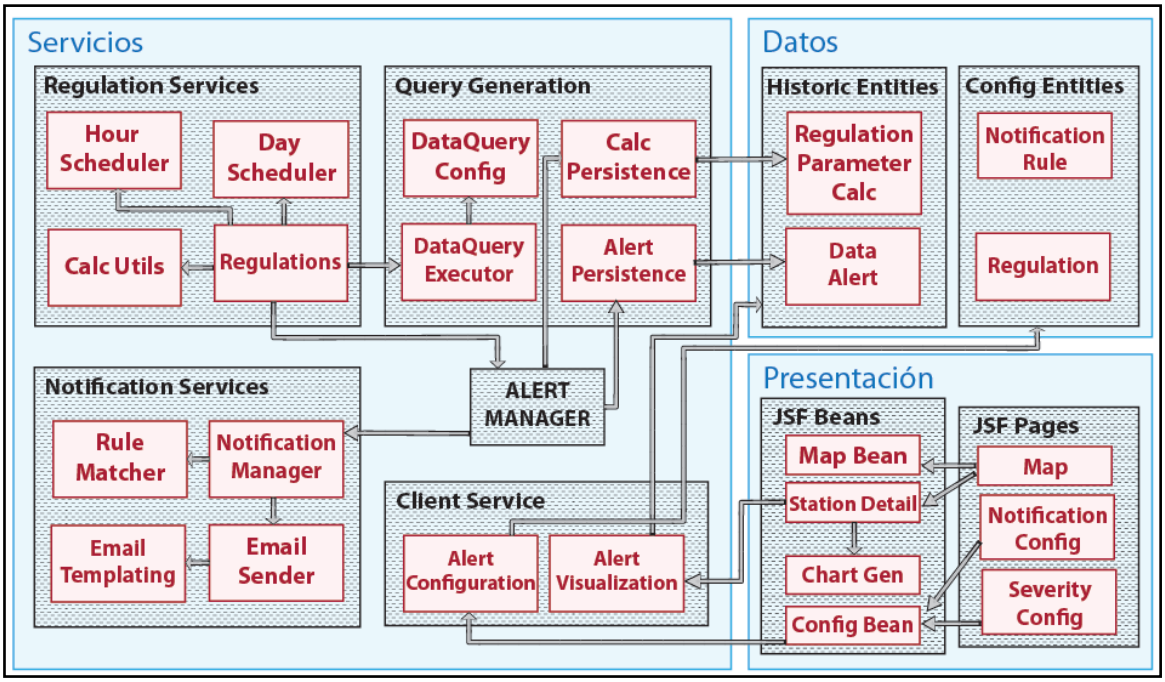

Ilustración 4. Detalle de componentes por capa y su interacción

I - Capa de Datos

Los componentes de esta capa contienen la representación de las entidades que son persistidas en la base de datos (ver 3.3 Modelo de Datos), proveyendo de operaciones CRUD, además de herramientas para la ejecución de consultas avanzadas.

- **Historic Entities.** Contiene a las entidades *RegulationParameterCalc* y *DataAlert*. Estas entidades son las que representan eventos de cálculo de carga de un indicador y alertas levantadas, respectivamente.
- **Configuration Entities.** Contiene a las entidades *Regulation* y *NotificationRule*, que representan las normativas vigentes y las reglas de notificación que el usuario ha definido.

II - Capa de Servicios

- **Regulation Services.** Responsable de la lógica de cálculo de la carga de los indicadores respecto a su normativa aplicable y de la realización de la misma periódicamente. Este componente debe hacer uso de *Query Generation*, para obtener los datos de mediciones que alimentan el cálculo, y también *Alert Manager*, para la persistencia del cálculo y levantamiento de las alertas, en el caso que corresponda. En detalle:
    - **Hour Scheduler y Day Scheduler.** Encargados de la lógica de agendar el cálculo de normativa horaria y diaria, respectivamente.

- **Calc Utils.** Encargado de proveer a *Regulations* de las funciones matemáticas y otras utilidades necesarias para realizar los cálculos. Por ejemplo, cálculo de percentiles o filtrar máximos diarios.
- **Regulations.** Encargado de contener la lógica de cálculo para cada normativa. Agenda el cálculo de cada normativa utilizando *Hour Scheduler* y *Day Scheduler*. Cada vez que se debe realizar un cálculo, utiliza *DataQuery Config* y *DataQuery Executor* para obtener los datos de mediciones necesarios. *Alert Manager* es invocado para la persistencia y levantamiento de alertas.

- **Query Generation.** Este componente se encarga de la generación de consultas a las entidades para obtener datos de mediciones y persistir históricos de cálculos y alertas.
  - **DataQuery Config.** Se encarga de representar una configuración específica de una consulta de mediciones. Dentro de las configuraciones posibles están: indicador a consultar, rango de fechas, función de agregación. Esta componente es utilizada por *DataQuery Executor* para la generación de las consultas.
  - **DataQuery Executor.** Se encarga de ejecutar una configuración de *DataQuery Config*, transformando la configuración en una consulta ejecutable para la entidad *Measurements*, que almacena el histórico de mediciones. Esta componente es utilizada por *Regulations* para obtener las mediciones necesarias para sus cálculos.
  - **Calc Persistence.** Esta componente persiste los cálculos de carga de los indicadores, utilizando las operaciones CRUD de *RegulationParameterCalc*. Se encarga de validar cada cálculo que se desea persistir, eliminando cualquier tipo de duplicado Esta componente es invocada por *Alert Manager*.
  - **Alert Persistence.** Decide cuándo una alerta detectada está duplicada o no y la persiste utilizando las operaciones CRUD de *DataAlert*. Esta componente es utilizada por *Alert Manager* para determinar cuándo enviar notificaciones a usuarios.

- **Alert Manager.** Es invocado por *Regulations* una vez que se ha realizado algún cálculo. *Alert Manager* se encarga de detectar duplicados, detectar alertas y persistirlas en el caso que se deba, para esto hace uso de *Calc Persistence* y *Alert Persistence*. En el caso que se levante una alerta, invoca a *Notification Manager* para el envío de notificaciones a los usuarios que corresponda.

- **Notification Services.** Esta componente se encarga de aplicar las reglas de notificación cuando se levanta una alerta y de enviar las notificaciones que correspondan.
  - **Notification Manager.** Se encarga de la lógica de envío de notificaciones. Esta componente maneja las posibles métodos de notificación (actualmente sólo correo electrónico), utiliza *Rule Matcher* para buscar los usuarios a notificar, y envía las notificaciones haciendo uso de *Email Sender*. *Notification Manager* es usado por *Alert Manager* cada vez que se decida que una alerta debe ser notificada.
  - **Rule Matcher.** Se encarga de tomar una alerta levantada y encontrar las reglas notificación que apliquen. Para esto hace uso de *Notification Rule* de la Capa de *Datos*. Esta componente es usada por *Notification Manager* una vez que se ha determinado que una alerta debe ser notific*ada*.

- **Email Templating.** Este componente es responsable de tomar una alerta y generar texto HTML con la información de notificación para que *Email Sender* pueda difundirlo.
- **Email Sender.** Se encarga de la lógica de envío de correos electrónicos, los que son enviados de manera asíncrona, de modo que no interfieran con el proceso de cálculo y levantamiento de alertas. Hace uso de *Email Templating* para la generación del contenido de los correos.

- **Client Services.** Es el componente intermediario entre las entidades de la Capa de *Datos* y la capa de *Presentación* para la visualización y configuración de las alertas.
  - **Alert Configuration.** Contiene la lógica de validación y persistencia de tanto las configuraciones de niveles de criticidad, como de reglas de notificación. Es usado por *Config Bean* de la Capa de *Presentación.*
  - **Alert Visualization.** Contiene la lógica de visualización de alertas en la Capa de Presentación, realizando consultas de alertas levantadas y cálculos históricos a *Historic Entities.* Esta componente es utilizada por *Station Detail*.

III - Capa de Presentación

Los componentes de esta capa se encargan de la interacción con el usuario, para la configuración y visualización de las alertas. Fueron diseñados para ser utilizados en el framework *Java Server Faces,* que es lo que *EMIS* utiliza para la generación de la interfaz de usuario en *HTML*.

- **JSF Beans.** Contiene la lógica de presentación, siendo el nexo entre la generación de páginas HTML y la lógica de *Client Services.*
  - **Map Bean.** Encargado de proveer de los filtros por localización, central, estación; filtros por criticidad, clasificación de normativa. Para esto se debe hacer uso de *Alert Visualization,* además de utilizar otras componentes de *EMIS* para obtener la información de estaciones, centrales, clasificación y localización.
  - **Station Detail.** Encargado de proveer de la información necesaria para visualizar el detalle de una estación. Esta componente está en cargada de consultar a *Alert Visualization* por el estado de carga de cada indicador y de generar los indicadores angulares utilizando *Chart Generation.*
  - **Chart Generation.** Esta componente se encarga de generar los indicadores angulares a utilizar para mostrar la carga de un indicador respecto a una normativa.
  - **Config Bean.** Provee de los datos y ejecuta la lógica de validación y persistencia de *Alert Config* para las vistas de configuración de niveles de criticidad y notificaciones.

- **JSF Pages.** Aquí residen los templates JSF con los que se genera las páginas HTML que serán mostradas al usuario.
  - **Map.** Se encarga del despliegue de la interfaz de visualización de estaciones utilizando mapas e indicadores angulares.

- **Notification Config y Severity Config.** Se encargan del despliegue de las interfaces de configuración de niveles de criticidad y configuración de notificaciones, respectivamente (ver 3.4.3 Configuración de Criticidad y 3.4.4 Configuración de Notificaciones).

3.2.3. Diagrama de Secuencia

Desde un punto de vista temporal, el cálculo de una normativa comienza cuando a *Hour Scheduler* o *Day Scheduler* les corresponde invocar a sus normativas asociadas, en la Ilustración 5 se puede apreciar la secuencia de interacción de las componentes para una normativa cualquiera.

Day ó *Hour Scheduler* inicia el proceso de cálculo de *Regulations*. Luego *Regulations* consulta a la entidad *Parameters* los indicadores que están asociados a la normativa en cuestión. Luego, por cada indicador se procede a realizar los cálculos. Se configura la consulta y se ejecuta con *DataQueryExecutor* quien a su vez consulta la entidad *Measurements*. Una vez que se tienen los valores desde la base de datos se realizan los cálculos finales con *CalcUtils*.

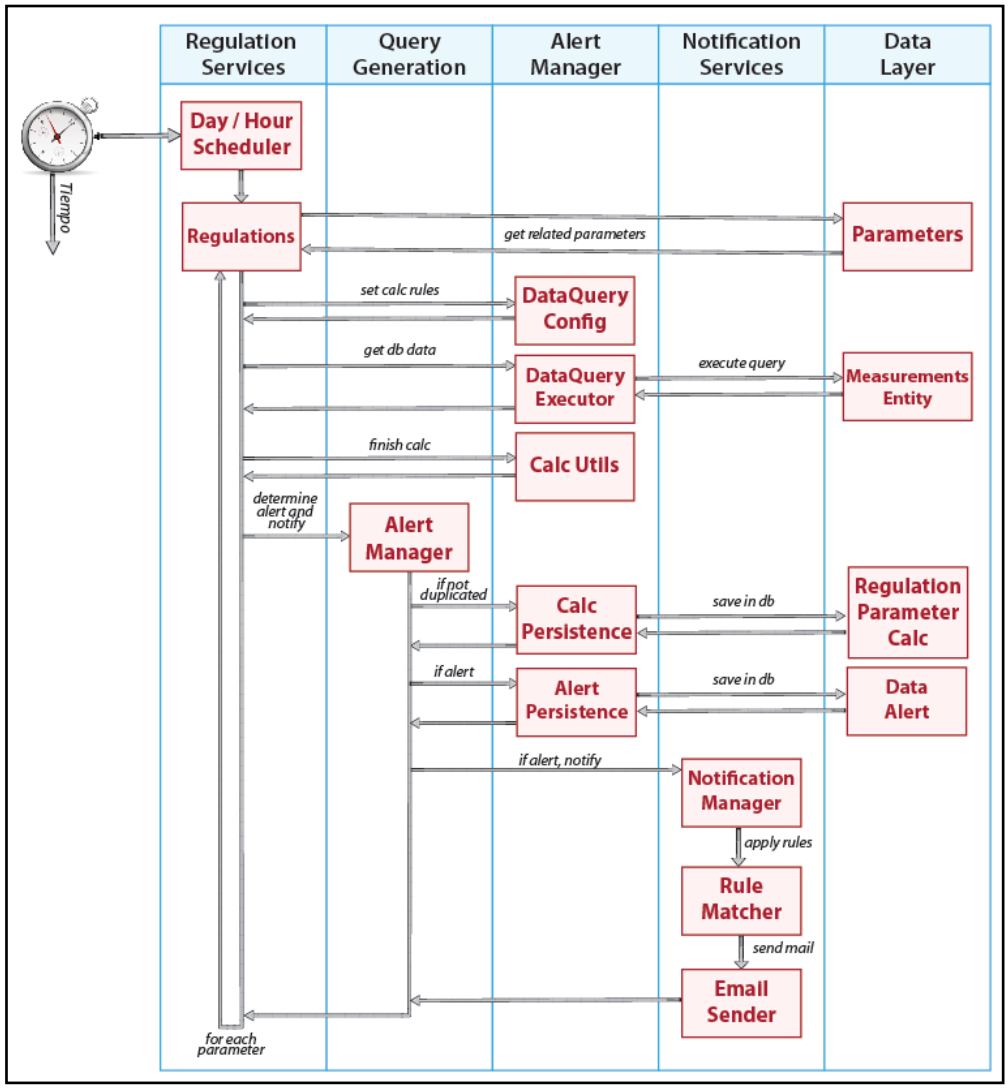

Ilustración 5 Vista temporal de interacción de componentes

Ya obtenido el nivel de carga del indicador respecto a la normativa, *AlertManager* se encarga de decidir si existen cálculos o alertas duplicadas y de persistirlas, invocando a *CalcPersistence y Alert Persistence.* En el caso que se detecte alguna nueva alerta, se invoca a *NotificationManager*, el cual utiliza a *RuleMatcher* para obtener los usuarios a notificar y luego invoca a *EmailSender* para el envío de notificaciones por correo electrónico.

3.3. Modelo de Datos

El modelo de datos diseñado para soportar al módulo se puede apreciar en la Ilustración 6. A la derecha están las entidades ya existentes en EMIS, en tanto a la izquierda se puede apreciar las entidades diseñadas para el módulo de alertas. A continuación, se describen cada una de las entidades modeladas, indicando el significado de sus campos. Por el lado de los módulos de EMIS ya implementados:

- **User.** Representa a los usuarios del sistema. Estos usuarios son los que podrán ser notificados cuando se levante una alerta, a la dirección de correo que indique el campo *email*.
- **Station.** Representa las estaciones de monitoreo. El campo *name* indica el nombre que identifica a la estación, por ejemplo "CH10". Central hace referencia a la central a la que pertenece la estación, por ejemplo la "Central Térmica Mejillones". *Latitude* y *longitud*e indican las coordenadas geográficas en donde se sitúa la estación.
- **Parameter.** Representa los indicadores. Posee un nombre que lo describe, por ejemplo "MP10 en JJLatorre". *Station* corresponde a la estación a la que pertenece el indicador (en este caso JJLatorre). Frecuency corresponde a la frecuencia de actualización del parámetro en minutos, que es leído por el servicio de recolección de datos para programar las consultas al WebService. *Unit* corresponde a la unidad en la que se mide el indicador, por ejemplo, "ppm" o *Partículas por Millón.* Finalmente, *code* indica un código que lo identifica al consultar al WebService, por ejemplo "MP10@JJLAT".
- **Measurement.** Representa cada medición hecha por cada indicador. *Parameter* es el indicador de la medición. *DateTime* corresponde a la fecha y hora de la medición. En tanto que *value*, corresponde al valor registrado.

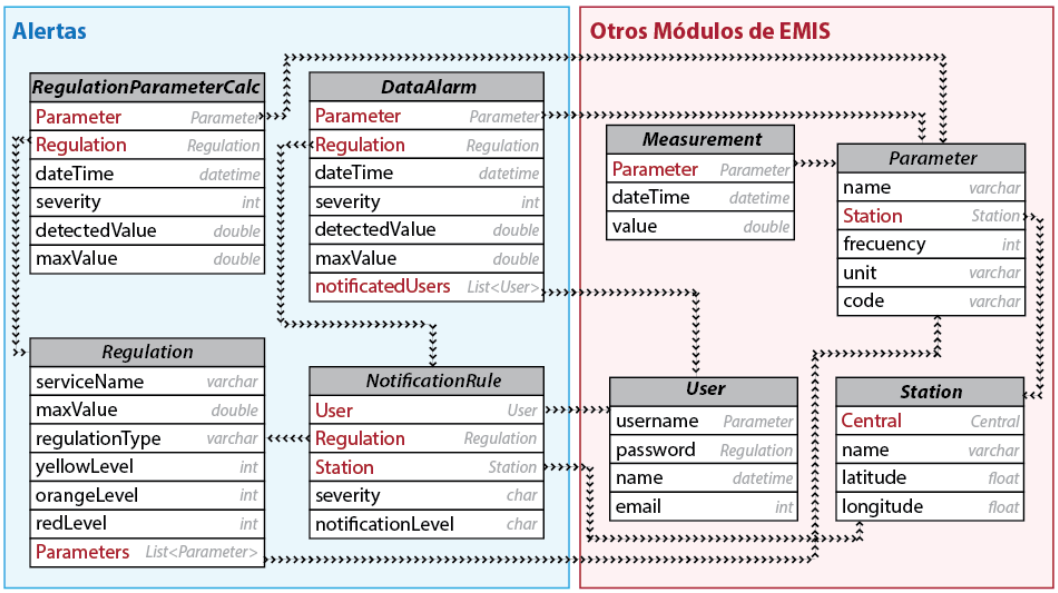

Ilustración 6. Modelo de Datos

Por el lado del módulo de Alertas:

- **Regulation.** Representa las normativas. A continuación se detalla cada campo:
  - **serviceName.** Corresponde al nombre del servicio de *Spring* al que está asociado esta normativa, por ejemplo "SO2Annual" (ver 3.5.1 Cálculo Normativas y Mapeo a BD).
  - **maxValue.** Máximo permitido por la normativa.
  - **regulationType.** Clasificación de la normativa, por ejemplo "Emisiones - Horaria".
  - **yellowLevel, orangeLevel y redLevel.** Corresponden a las configuraciones de criticidad para cada nivel, Leve, Mediano y Grave, respectivamente.
  - **parameters.** Corresponde a los indicadores que están afectos a la normativa.
- **RegulationParameterCalc.** Esta entidad representa al histórico de cada cálculo que se ha realizado, por cada indicador y normativa.
  - **Parameter.** Corresponde al indicador.
  - **Regulation.** Normativa asociada.
  - **dateTime.** Fecha y hora en que fue realizado el cálculo.
  - **dataDateTime.** Fecha y hora que corresponde a la fecha del último valor medido de Parameter.
  - **severity.** Criticidad de la medición. Un valor nulo indica que no es crítico, en tanto que los valores 0, 1 y 2 corresponden a los niveles *Leve*, *Mediano* y *Grave*, respectivamente.
- **NotificationRule.** Representa las reglas de notificaciones a usuarios, a continuación se explica cada campo:

- **User.** Usuario a notificar.
- **notificationLevel.** Corresponde al nivel de la regla de notificacion. Los valores 'g', 'r' y 's', que corresponden a los niveles *Global*, *Normativa* y *Estación*.
- **Regulation.** Corresponde a la normativa asociada a la configuración. Para los niveles Estación y Global este valor es nulo.
- **Station.** Estación asociada a la regla. Para los niveles Global y Normativa este valor es nulo.
- **severity.** Nivel de criticidad en el cual se desea notificar al usuario.

- **DataAlarm.** Esta entidad representa al histórico de alertas que han sido levantadas. Los campos son los mismos que *RegulationParameterCalc*, solo que se agrega la lista de usuarios que fueron notificados. A diferencia de *RegulationParameterCalc*, acá solo son registrados las alertas que se levantan, eliminando cualquier duplicidad.

3.4. Diseño de la Interfaz de Usuario

Para la interfaz de usuario se diseñaron cuatro pantallas principales, que son detalladas en esta sección. La primera, es una vista satelital de las estaciones de monitoreo que muestra el estado de cada una de ellas. La segunda, corresponde a la visualización del histórico de alertas ocurridas en una estación. La tercera, corresponde a la interfaz de configuración de criticidad de cada normativa. La cuarta y última, corresponde a la interfaz de configuración de notificaciones.

Las interfaces finales e implementadas pueden ser consultadas en el apartado 5.1 Instalación del Módulo de Alertas. Por otra parte las que no fueron implementadas son propuestas como trabajo futuro en el capítulo 6.

3.4.1. Visualización de Estaciones

El fin de esta vista es que el usuario acceda fácilmente al estado y detalle de cada estación de monitoreo, para lo cual se diseñó una visualización satelital de las estaciones, como puede ser visto en la Ilustración 8. Se utilizó un código de colores (ver Ilustración 7) para identificar el estado de cada estación. El estado "Sin Alertas Configuradas" es utilizado para estaciones en las que no hay normativa vigente aplicable. El estado "Sin Alertas" aplica para estaciones que poseen normativa aplicable pero que todos sus indicadores están fuera de los tres rangos críticos (Leve, Media, Grave). Para estaciones con alertas, el estado de la estación corresponde a la criticidad más alta de las alertas detectadas.

Ilustración 7. Código de colores del estado de una estación

Al ingresar a esta vista, el usuario apreciará todas las estaciones de monitoreo, sin embargo, mediante los filtros que posee en el panel lateral izquierdo, podrá modificar su vista, las opciones son las que se listan a continuación:

- **Ir a estación**. Permite buscar una estación según su localidad (por ejemplo, Tocopilla) y central (por ejemplo, Central Térmica Hornitos) y ubicarla en el mapa, mostrando su detalle.
- **Filtro por clasificación (o Variable)**. Permite mostrar u ocultar las estaciones de Calidad de Aire o de Emisiones.
- **Filtro por estado de gravedad**. Permite mostrar/ocultar estaciones según su estado.

Al hacer click en una estación se despliega un detalle que muestra la carga de cada indicador medido en la estación respecto a su normativa aplicable, en forma de medidor angular. Como se puede apreciar en la Ilustración 9, el medidor angular tiene como máximo el máximo de la normativa a la que hace referencia. Los niveles de criticidad están marcados mediante bandas de color amarillo, naranjo y roja. Dentro del círculo del medidor se muestra además, la unidad en que se está midiendo el indicador, el valor numérico calculado y el contaminante. Al costado derecho del indicador se puede apreciar la carga (en porcentaje) del indicador, el valor numérico calculado (con su respectiva unidad), el rango de fechas que se utilizó para los cálculos, la normativa asociada y el máximo permitido. Las letras de la carga del indicador y de su valor numérico pueden ser de color verde, amarillo, naranjo o rojo, dependiendo del nivel en que se encuentre.

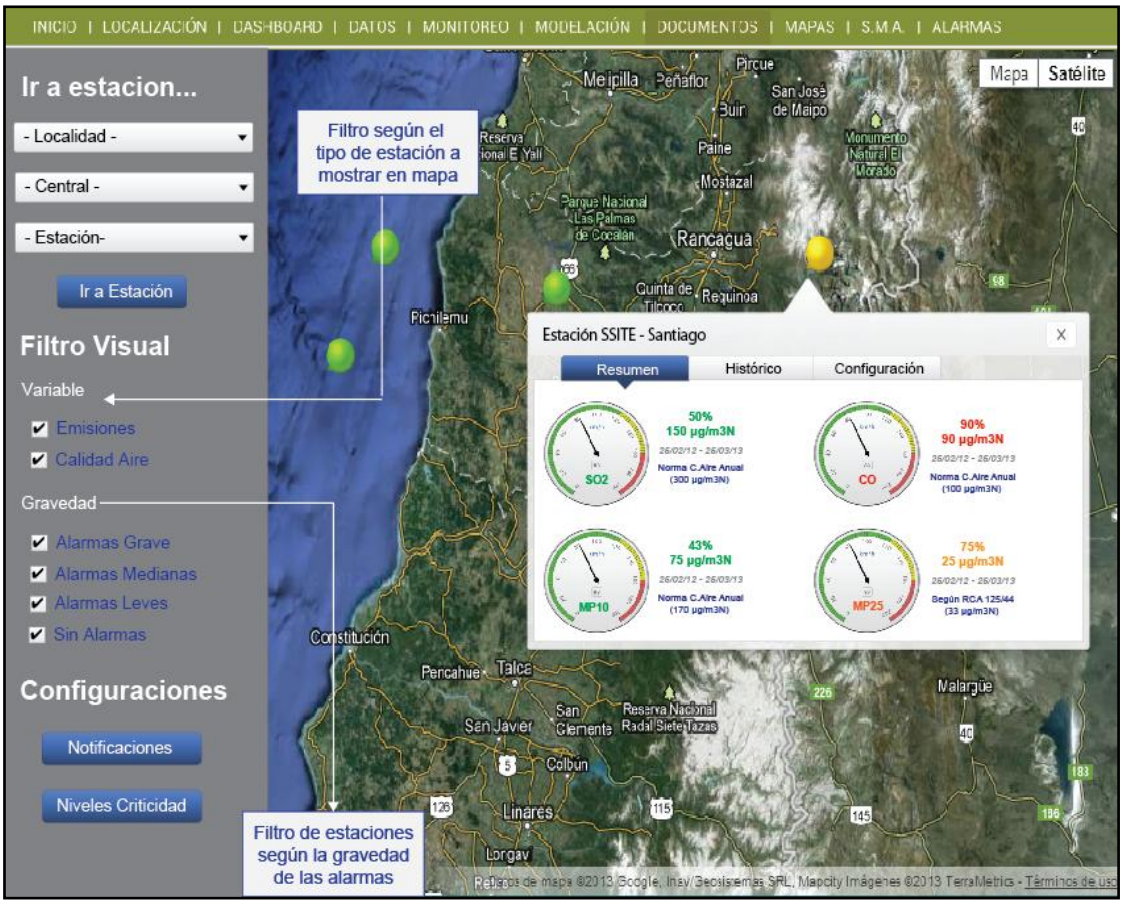

Ilustración 8. Vista geolocalizada de las estaciones de monitoreo con sus respectivas mediciones

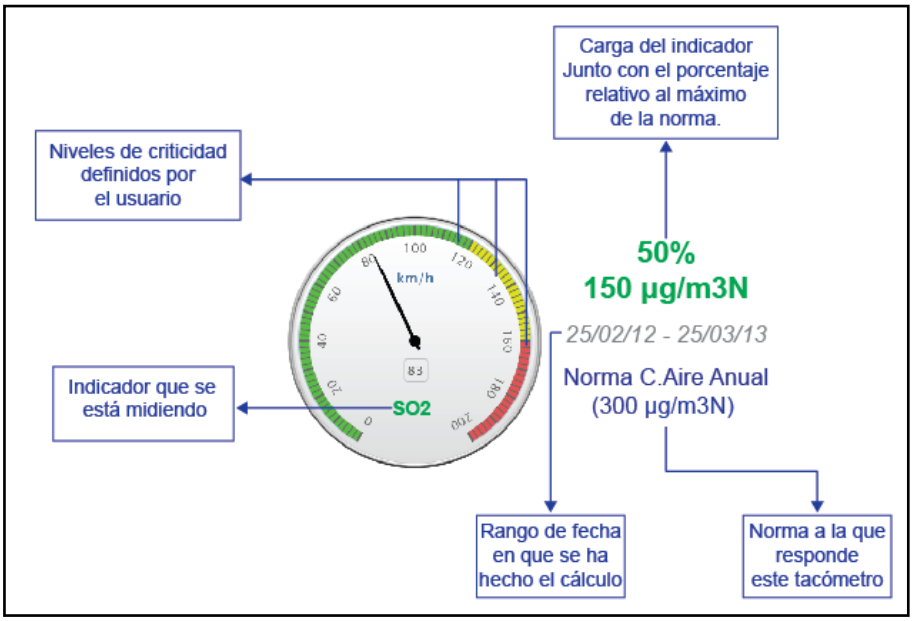

Ilustración 9. Diseño del tacómetro para un indicador y normativa

3.4.2. Histórico de Alertas

Esta interfaz tiene como propósito brindar al usuario la trazabilidad de las alertas que han ocurrido en el sistema. Se acopla a la interfaz de visualización de estaciones, agregándose como

una pestaña llamada "histórico". Como se puede apreciar en la Ilustración 10, la vista posee una pestaña por cada indicador, al seleccionar uno, se muestra la lista de alertas ocurridas para aquel indicador.

Ilustración 10. Histórico de alertas en una estación

Las alertas son presentadas ordenadas por fecha de ocurrencia indicando (Ilustración 11): su nivel de criticidad, valor detectado, carga detectada del indicador, fecha de ocurrencia, cantidad de usuarios notificados y la normativa a la que corresponde. El color del texto y el ícono asociado dependen de la criticidad de la alerta detectada.

Al hacer click en "detalle" se puede visualizar de manera más detallada la alerta (Ilustración 12), presentándose adicionalmente un gráfico que muestra el comportamiento del indicador antes y después de detectada la alerta. El gráfico se ajusta automáticamente dependiendo del horizonte temporal de la normativa y de la agregación de la misma. Adicionalmente, se presenta la lista de los usuarios que fueron notificados de la ocurrencia de la alerta.

Ilustración 11. Histórico de una alerta

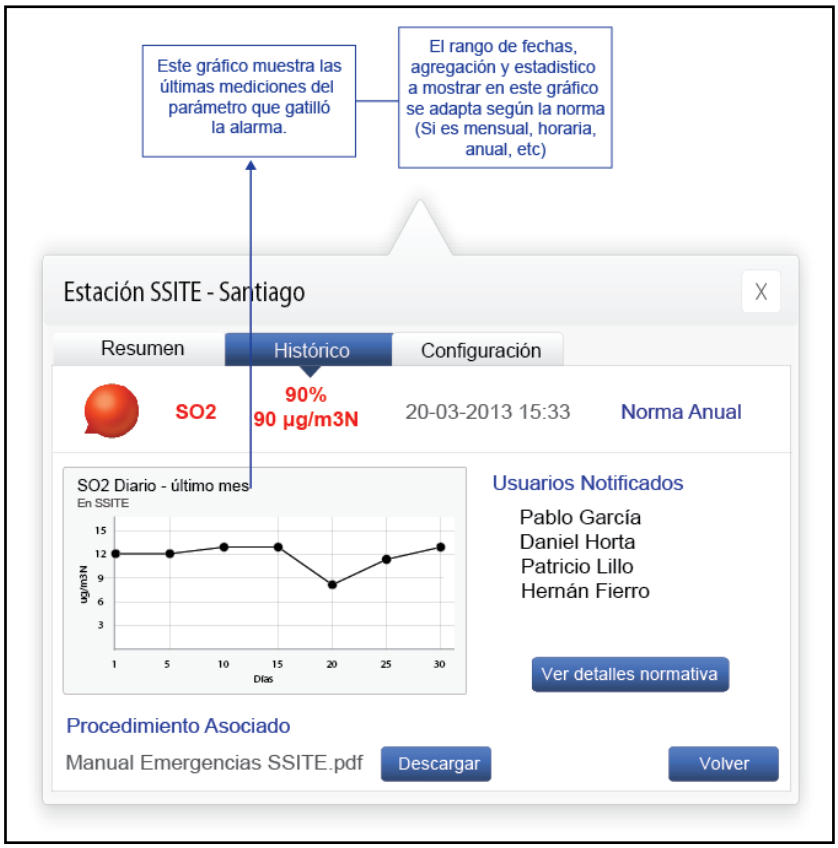

Ilustración 12. Vista detallada de una alerta

3.4.3. Configuración de Criticidad

El fin de esta interfaz es que el usuario pueda determinar por cada normativa, los porcentajes de carga en los que se levantarán las alertas Leves, Medianas y Graves. El usuario selecciona la normativa que desea consultar o modificar e inmediatamente se despliega información referente a la misma: máximo permitido, forma de cálculo, niveles de criticidad actuales (Ilustración 13). Esta interfaz también muestra al usuario las estaciones que poseen indicadores asociados a la normativa seleccionada, indicando su nivel actual de carga.

Ilustración 13 Configuración de criticidad

3.4.4. Configuración de Notificaciones

En estas interfaces el usuario puede configurar a quién y en qué casos se deben enviar notificaciones. Permite al usuario crear reglas de notificación en los niveles Global, Estación y Normativa, definidos en la sección 2.3.3. La configuración para nivel Global puede ser vista en la Ilustración 14, el usuario puede agregar o quitar usuarios del sistema a los que se les enviarán notificaciones globales, según nivel de criticidad.

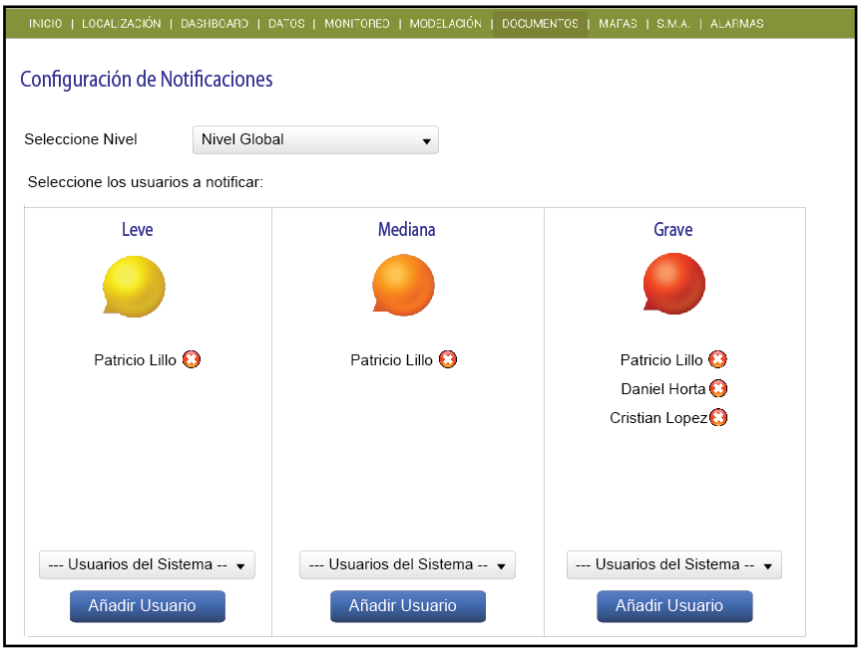

Ilustración 14. Configuración de notificaciones nivel global

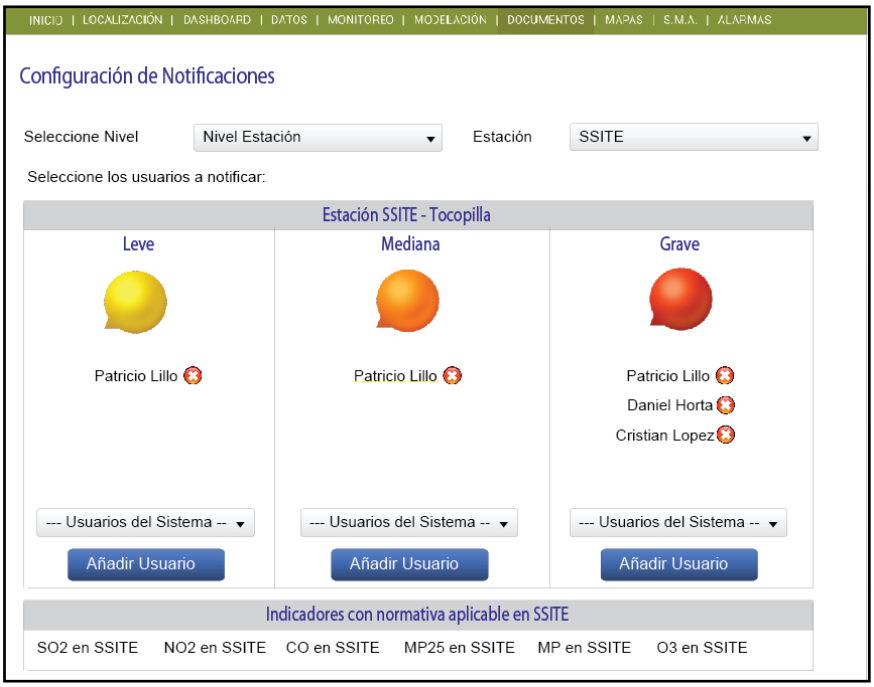

Ilustración 15. Configuración de notificaciones nivel estación

Para el caso de configuración del Nivel Estación (Ilustración 15), el usuario debe seleccionar la estación a configurar y luego seguir la misma lógica del Nivel Global. En la parte inferior se muestra al usuario los indicadores de la estación seleccionada, que estarán afectos a esta configuración.

El caso del Nivel Normativa (Ilustración 16) es análogo al de Nivel Estación. En la parte inferior de la pantalla se muestran los indicadores asociados a la normativa seleccionada.

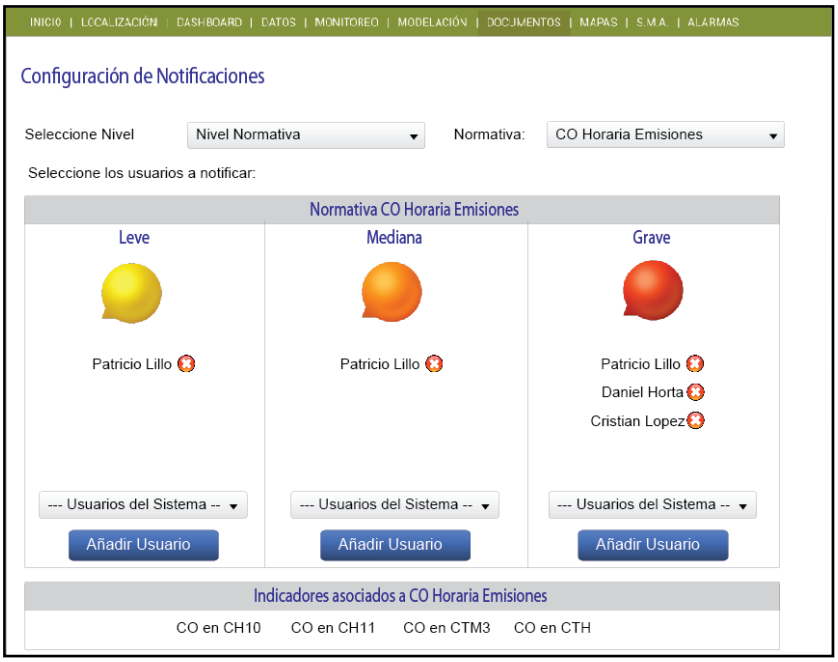

Ilustración 16. Configuración de notificaciones nivel normativa

3.5. Aspectos de Implementación

En esta sección se describe la implementación de ciertos aspectos relevantes del software, indicando detalles técnicos y razones de las decisiones tomadas.

3.5.1. Cálculo Normativas y Mapeo a BD

Como se puede apreciar en el modelo de datos (ver 3.3), la entidad *Regulation* (que representa las normativas) no posee campos que le permitan almacenar las reglas de cálculo. Para esto posee el campo *serviceName*, cuyo significado será explicado en este apartado.

Considerando que la autoridad chilena no emite nueva normativa frecuentemente y que la existente no sufre modificaciones en el corto o mediano plazo, las reglas de cálculo se implementaron totalmente en código. Para esto, se creó la clase abstracta *RegulationService*, cada clase (no abstracta) heredada de ella representa una normativa diferente. Además, *RegulationService* posee la lógica por defecto de llamado a *AlertManager*, obtención de indicadores asociados a la normativa y cálculo de carga del indicador con respecto a la normativa. Cada clase heredada debe implementar los siguientes métodos abstractos:

- **public abstract Double getMaxValue()**. Retorna el máximo permitido de la normativa.
- **public abstract RegulationParameterCalc calculateParameterValue(Parameter parameter).** Toma como argumento un indicador y calcula su valor actual respecto a la normativa. Aquí se realizan las consultas de mediciones a la base de datos y se aplican las reglas de cálculo.

Por otra parte, *RegulationService* implementa los siguientes métodos:

- **public Integer calculateParameterLoad(Parameter parameter).** Se encarga de invocar a *getMaxValue* y *calculateParameterValue* para un parámetro en especifico, retornando el porcentaje de carga del indicador respecto a la normativa.
- **public void doJob().** Se encarga de realizar toda la lógica para la normativa. Primero, consulta los parámetros afectos a la normativa, luego invoca *calculateParemterLoad* para cada uno de ellos, para luego invocar a *AlertManager*, que se encarga de persistir el cálculo y decidir si existen alertas. Este método posteriormente es agendado según la frecuencia que corresponda a la normativa (ver apartado siguiente).

Cada clase heredada lleva la anotación de *Spring "@Service('serviceName')"*, el mapeo de normativa entre base de datos y código es realizado utilizado el nombre del servicio, debe haber un match de *serviceName* entre la entidad en base de datos y el servicio *Spring*. La anotación *@Service* implica que *Spring* instanciará la clase al momento de levantar la aplicación, la instancia posteriormente puede ser buscada y referenciada utilizando su *serviceName*.

3.5.2. Programación de Tareas de Cálculos de Normativa

Para la implementación de la programación de cálculos de normativa tanto diaria, como horaria, se utilizó la librería *Quartz Scheduler*, la cual es utilizada por el *Servicio de Recolección de Datos* para agendar las consultas al WebService proporcionado por *E-CL*.

Luego de revisar la documentación oficial de la integración de *Spring* y *Quartz*, se identificó la anotación Java "*@Scheduled"*, la cual permite de manera simple crear una tarea programada, basta incluir esta anotación en un método de tipo "*void"* e indicar una cantidad de milisegundos. Con esto, cada vez que se levante la aplicación, *Spring* se encargará de invocar a *Quartz* para crear la tarea programada.

Lo anterior fue utilizado para agendar la ejecución del método *doJob()* de *RegulationService* (ver apartado anterior), esto se realizó definiendo la estructura de clases que se describe a continuación (ver Ilustración 17):

- Las clases abstractas *HourlyUpdatedRegulation* y *DailyUpdatedRegulation* que extienden de *RegulationService*.
- Estas clases sobrescriben el método *"void doJob()"* (ver apartado anterior) agregando las anotaciones *"@Scheduled(fixedRate = 3600000)"* y *"@Scheduled(fixedRate = 86400000)*", que corresponden a una hora y un día en milisegundos, respectivamente.
- Cada implementación concreta de *RegulationService* extiende de *HourlyUpdatedRegulation*, o bien, de *DailyUpdatedRegulation*, dependiendo de su frecuencia de actualización.

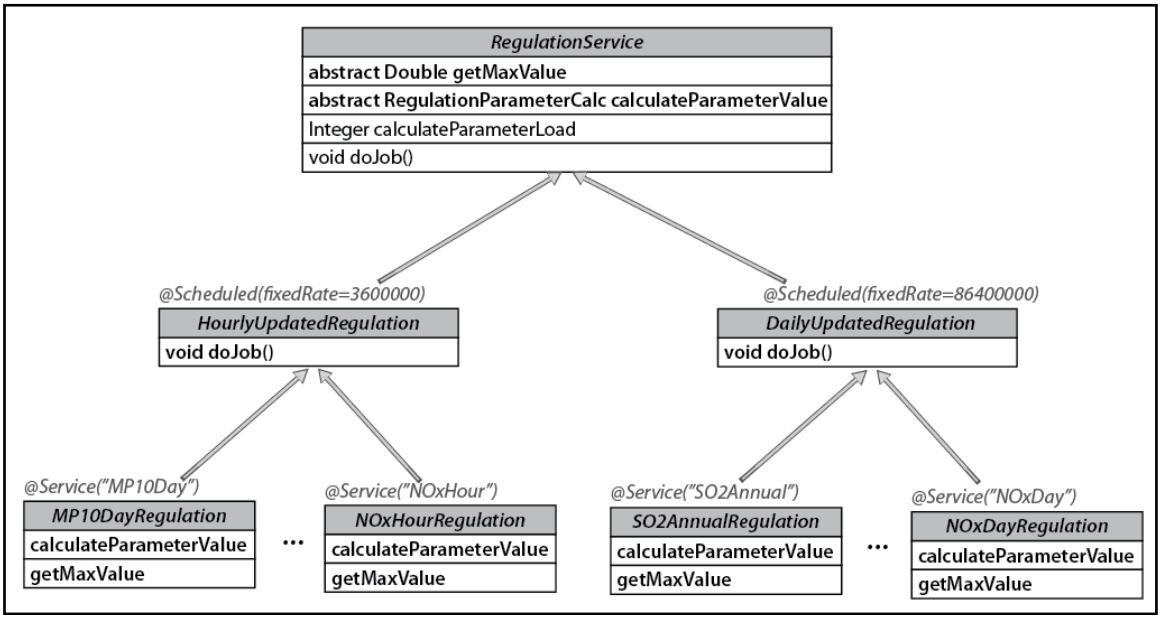

Ilustración 17. Jerarquía clases para *RegulationService*

La decisión de utilizar *Quartz* fue tomada principalmente debido a que se deseó mantener uniformidad en las tecnologías utilizadas y considerando que permite de manera simple cumplir con los requerimientos de la programación de los cálculos.

3.5.3. Generación de Consultas

Dentro de los pasos necesarios para ejecutar las reglas de cálculo definidos por cada normativa, se encuentra el obtener datos de mediciones históricas para los indicadores, utilizando diversos horizontes temporales, funciones de agregación, etc. Hasta antes de iniciar el trabajo de construcción del módulo de alertas, para la generación de consultas, se habían utilizado las características estándares de la *Java Persistence API*. Estas son:

- **JQL.** Lenguaje de consultas especifico del estándar *JPA*, similar a SQL. Las consultas son escritas en un S*tring*.
- **CriteriaBuilder.** Permite la generación de consultas dinámicas, donde cada elemento de la misma está mapeado como un objeto. Tienen la ventaja de que son *type-safe*.

Luego de comparar estas opciones con otras librerías, se optó por el uso del framework *QueryDSL*, el cual permite generar consultas de manera similar a *CriteriaBuilder* pero con ciertas ventajas. Por una parte, la construcción de consultas es más simple y legible. Por otra parte, posee un mecanismo de chequeo de consultas superior a *CriteriaBuilder*, puesto que no solo se asegura que haya un *type-safing* para las entidades y funciones SQL, sino que también chequea que la consulta generada sea sintácticamente correcta.

Un ejemplo de construcción de consultas con *QueryDSL* puede ser apreciado en la Ilustración 18, que corresponde a una consulta que obtiene el promedio diario de mediciones para un indicador dado, en período desde el 06-27-2009 al 06-27-2012.

```
QMeasurement qm = QMeasurement.measurement;
Parameter parameter = getParameter(); //a parameter
Calendar start = Calendar.getInstance();
start.set(2009,06,27);
Calendar end = Calendar.getInstance();
end.set(2012,06,27);
JPAQuery jpq = new JPAQuery(em);
jpq = jpq.from(qm).where(qm.parameter.eq(parameter));
jpq = jpq.where(qm.dateTime.goe(start.getTime()));
jpq = jpq.where(qm.dateTime.lt(end.getTime()));
jpq = jpq.orderBy(qm.dateTime.min().asc());
jpq = jpq.groupBy(qm.dateTime.year());
jpq = jpq.groupBy(qm.dateTime.month());
jpq = jpq.groupBy(qm.dateTime.dayOfMonth());
List<Double> values = jpq.list(qm.value.avg());
```

Ilustración 18. Ejemplo de consulta con QueryDSL

Con el uso de *QueryDSL* se logró un desarrollo más productivo, puesto que se tiene mayor certeza que las consultas generadas sean correctas, en tiempo de compilación.

3.5.4. Generación de Medidores Angulares

El despliegue de gráficos en los otros módulos del sistema fue implementado por el equipo de desarrollo utilizando la librería *JavaScript HighCharts JS, la* cual opera recibiendo las configuraciones y datos en formato *JSON*. Esta librería posee la facultad de generar indicadores angulares, por lo que se decidió mantenerla y utilizarla para el módulo de alertas.

Con el fin de hacer el código más mantenible y preservar una buena separación entre las capas de la arquitectura de la solución, se construyó un wrapper de la librería *Highcharts* en *Java*. Para esto se realizó lo siguiente:

- Se construyó una jerarquía de clases que mapea las configuraciones JSON de la librería. Luego se utilizó la librería *Java Gson*, para transformar las configuraciones a JSON.

- Un componente *JSF* que recibe las configuraciones y se encarga de invocar a *HighCharts* para renderizar el indicador.

En la Ilustración 19 e Ilustración 20 se puede apreciar un ejemplo de indicador angular, mostrando las configuraciones JSON, el mapeo correspondiente, el uso del componente *JSF* y el resultado final.

```
chart: {
    type: 'gauge'
},
title: {
    text: 'Medición SO2'
},
pane: {
    startAngle: -150,
    endAngle: 150,
    background: [.....]
},
yAxis: {
    min: 0,
    max: 200,
    //.....
    tickPixelInterval: 30,
    plotBands:
    [{ from: 0, to: 120, color: '#55BF3B'
  }, { from: 120, to: 160, color: '#DDDF0D'
  }, { from: 160, to: 200,color: '#DF5353'}]
},
series: [{
    data: [80],
    tooltip: {
        valueSuffix: ' ug/m3N'
    }
}]
```

```
public Graph getSampleChart(){
    AngularGaugeGraph gauge;
    gauge = new AngularGaugeGraph();
    gauge.setTitleText("Medición SO2");
    gauge.getAngularYAxis().setMax(200);
    PlotBand[] plotbands;
    plotbands = gauge.getAngularYAxis().getPlotBands();
    plotbands[0].setFrom(0);
    plotbands[0].setTo(120);
    plotbands[0].setColor("#55BF3B");
    plotbands[1].setFrom(120);
    plotbands[1].setTo(160);
    plotbands[1].setColor("#DDDF0D");
    plotbands[2].setFrom(160);
    plotbands[2].setTo(200);
    plotbands[2].setColor("#DF7D0D");
    List<Serie> series = new ArrayList<>();
    Serie serie= new Serie();
    serie.setName("ug/m3N");
    serie.setData(new Double[]{80.0});
    series.add(serie);
    gauge.setSeries(series);
    return gauge;
}
```

Ilustración 19. Configuración de un indicador angular en JSON (izquierda) y Java (derecha)

```
<acomp:graph graph="${sampleBean.sampleChart}"   height="250px" width="200px" />
```

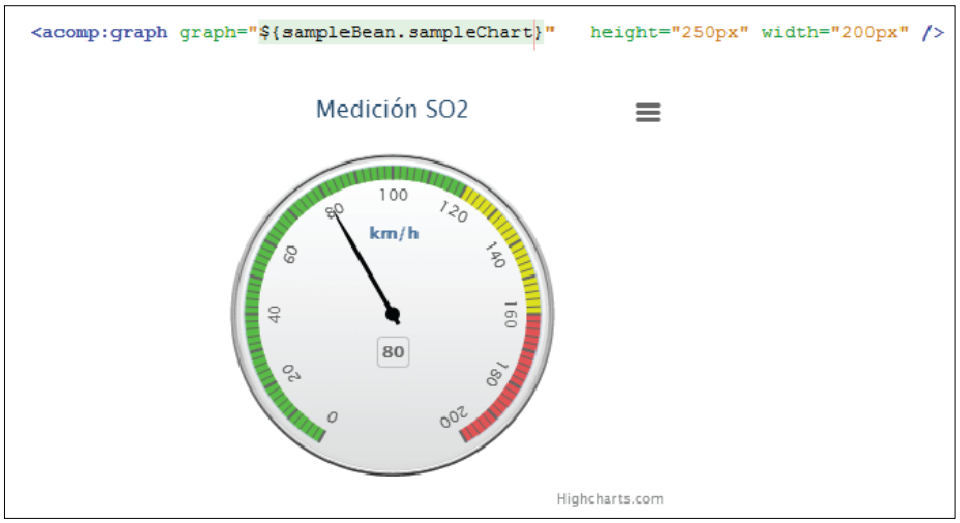

Ilustración 20. Uso del componente JSF (arriba), resultado final (abajo)

Posteriormente, se refactorizó el proyecto utilizando esta estrategia en otros módulos del sistema donde se generan gráficos con *HighCharts.*

3.5.5. Envío de Notificaciones y Asincronía

Un aspecto importante en la implementación del módulo de alarmas fue el envío de notificaciones para las alertas levantadas. Independiente del medio, o medios de notificación que se implementen, el requisito de usuario *RU008* establece que el tiempo de envío no puede exceder a un minuto, esto para cada usuario a notificar.

Existen diversos motivos por los cuales un envío podría retrasarse, como por ejemplo un error de comunicación o retraso por la saturación de algún servicio. Independiente de la causa de retraso, si los envíos son procesados de manera serial basta que un envío se retrase, para que todos los que siguen también se retrasen.

Para evitar el caso anterior, los envíos fueron implementados creando un *thread* por cada uno de ellos, para esto nuevamente se utilizó *Quartz*. En específico, se utilizó la anotación *"@Async"* que realiza la tarea antes descrita (ver Ilustración 21).

Por otra parte, al igual que para las normativas, el envío de correos fue implementado como un servicio de *Spring*. Para soportar otros mecanismos de notificación en el futuro, se definió que todos los servicios de notificación deben implementar la interfaz llamada *NotificationSender* (ver Ilustración 21).

```
@Service("emailNotificationSender")
public class EmailNotificationSender implements NotificationSender {
    //...

    @Override
    @Async
    public void sendNotification(User user, DataAlarm alarm) {
        //...
    }
}
```

Ilustración 21. Anotación @Async en EmailNotificationSender

Los envíos por email fueron implementados utilizando el servicio de envío de correos de *Spring MailSender*, el cual es una implementación del estándar *JavaMail API*. Para el contenido de los mensajes de correo se utilizó el motor de templates *Apache Velocity*.

4. Evaluación del Sistema

Una vez implementado el sistema, se ejecutaron tres etapas en la evaluación del sistema, primero se testeó cada funcionalidad del sistema, se realizaron pruebas de rendimiento del motor de base de datos SQL y se recogieron las apreciaciones de los usuarios finales del sistema realizando un *focus group*. Cada uno de los puntos nombrados anteriormente son detallados en este capítulo.

4.1. Testing

Cada funcionalidad implementada pasó por una etapa de testing, la cual fue realizada por el equipo de desarrollo del proyecto, en conjunto con tres expertos en medioambiente de ASYT. El proceso de testing se llevó a cabo realizando las siguientes tareas:

- **Presentación de las Funcionalidades.** El equipo de desarrollo se reúne con los expertos en medioambiente y se realiza una breve charla en la que se describen las funcionalidades del módulo y se presentan las interfaces correspondientes.
- **Evaluación de Interfaces.** Los expertos en medioambiente utilizan el sistema, navegan por cada interfaz y hacen llegar sus comentarios, poniendo énfasis en los siguientes puntos:
    - **Intuitividad de las interfaces**. Que las interfaces sean entendibles y fáciles de usar, de modo que el usuario final pueda alcanzar rápidamente productividad en el sistema sin necesidad de entrenamiento.
    - **Consistencia de las interfaces.** Que las interfaces efectivamente representen lo que pretenden, presentando la información suficiente que el usuario necesite para su uso.
    - **Aspectos técnicos.** Que términos técnicos de medioambiente y nomenclatura asociada estén empleados de manera correcta en el sistema.
- **Evaluación de Funcionalidades.** Los expertos solicitan información adicional para realizar cálculos de la normativa en forma paralela a los realizados por el sistema, de manera de contrastar resultados. Los expertos hacen llegar sus resultados con las diferencias apreciadas y posibles errores detectados.

Con toda la información recogida, se procedió a realizar los cambios correspondientes respecto a la evaluación de las interfaces. Para el caso de la evaluación de las funcionalidades, se revisaron las diferencias en los cálculos y se realizaron reuniones posteriores con los expertos para determinar el origen de las diferencias y realizar las correcciones correspondientes.

4.2. Evaluación de Performance

Se realizaron pruebas de rendimiento del motor de la base de datos respecto a las consultas que el *Módulo de Alertas* ejecuta, de manera de asegurar que el sistema tendrá una performance aceptable tanto al momento de instalar el sistema, como en el futuro. En esta sección se detallan los experimentos realizados, resultados obtenidos y medidas tomadas.

4.2.1. Pruebas realizadas y sus Resultados

Primero, se escogió una muestra de 3 consultas distintas en rango de fechas y agregación, dentro del conjunto de consultas generadas por el módulo. Un ejemplo es la consulta que se presenta a continuación, la cual retorna el promedio diario de las mediciones de un indicador (el 278), en un año móvil a contar del 18-06-2012.

```
SELECT avg(value)
FROM measurements
WHERE parameter_id=278 AND date_time >= '2012-06-18' AND date_time < '2013-06-18'
GROUP BY date_part('day',date_time), date_part('month', date_time) ,date_part('year', date_time)
ORDER BY min(date_time) ASC;
```

Luego, se prepararon cuatro instancias de bases de datos con diversos volúmenes de datos, instancias con 1, 2, 3, y 4 años de datos, de manera que los experimentos sean representativos de la situación futura del sistema. Para que el experimento resultase más representativo, las instancias fueron generadas en base a un *dump* de 6 meses de todas las mediciones que almacena EMIS, por lo que cada instancia contiene no sólo datos de los indicadores a consultar, sino que de todos los indicadores del sistema, incluso de indicadores no afectos a normativa. Los 6 meses de datos en *EMIS* suman alrededor de 35 millones de registros, por lo que las instancias de 1, 2, 3 y 4 años quedaron con 70, 140, 280 y 350 millones de registros, respectivamente.

A continuación, se ejecutaron las consultas escogidas en las instancias generadas, registrando el tiempo promedio de *runtime*. Para evitar que algún evento (como algún otro proceso del SO) distorsionase las mediciones, lo anterior fue repetido 5 veces, promediando los resultados. Los experimentos fueron realizados en una máquina con las siguientes especificaciones técnicas:

- Procesador Intel® Core i7® 2600 3.40GHz, 8MB Cache, 5GT/s.
- 6GB RDIMM, 1600MT/s, Low Volt, Dual Rank, x8 Data Width.
- Disco Duro 500GB 7.2K RPM SATA 3.5 pulgadas.

Los resultados obtenidos son resumidos en la siguiente tabla:

Tabla 3. Runtime de las consultas del módulo de alertas.

Consulta	Características	1 Año	2 Años	3 Años	4 Años
Consulta 1	Promedio Diario, 3 Años	-	-	120 seg	135 seg
Consulta 2	Promedio Horario, 3 Años	-	-	110 seg	130 seg
Consulta 3	Promedio Diario, 1 Año	1.9 seg.	68.0 seg.	100 seg.	140 seg
PROMEDIO		**1.9 seg**	**68 seg**	**110 seg**	**135 seg**

Se debe considerar que la máquina utilizada para los experimentos brinda un menor desempeño en comparación a los servidores de E-CL, lo que implica que las consultas se ejecuten más rápido en E-CL.

4.2.2. Análisis y Medidas Tomadas

De los resultados obtenidos se pudo apreciar que una vez que la base de datos alcance datos equivalentes a la instancia de 2 años, el tiempo medio de ejecución de las consultas generadas por el *Módulo de Alertas* no será lo suficientemente bajo como para asegurar el correcto funcionamiento del sistema. Lo anterior cobra más sentido considerando que los experimentos fueron realizados en condiciones ideales, distintas a las que tendrá el sistema completo en producción, puesto que:

- Se realizarán cálculos para múltiples normativas cada hora.
- Otros módulos también ejecutarán consultas, por ejemplo el *Servicio de Recolección de Datos*, módulo de Localización, Dashboard.
- EMIS tendrá usuarios utilizando el sistema de forma concurrente (se estima un máximo de 5).

Se realizó un análisis de lo implementado tanto en la construcción de las consultas, como en la estructura de la base de datos, con el fin de encontrar posibles optimizaciones que permitiesen mejorar el *runtime* medio de las consultas. Para la construcción de los campos y relaciones de la base de datos, no se encontraron optimizaciones en las que se pudiera apreciar una mejora significativa en los tiempos. En cambio, al analizar la estructura de tabla *Measurements* (encargada de registrar todas las mediciones de los indicadores), se pudo encontrar una optimización que produjo mejoras significativas.

La mejora consistió en crear un índice en la tabla *Measurements*, para las columnas (*date_time*, *parameter_id)* (ver 3.3 Modelo de Datos). La efectividad radica en que todas las consultas que el *Módulo de Alertas* ejecuta, en su cláusula *WHERE*, define condiciones para las columnas mencionadas, por lo que el motor de base de datos al utilizar este índice ejecuta las consultas más rápido. Esta optimización permitió reducir en promedio un 30% los tiempos de ejecución de las consultas.

4.3. Evaluación de Usuarios Finales

Una vez que el sistema se instaló en los servidores de E-CL y se realizaron las capacitaciones correspondientes (ver 5 Sistema Implementado), convocando a las mismas personas, se realizó un *focus group* en el que se recogieron las impresiones, comentarios y propuestas de mejoras de los usuarios.

Al momento de realizar el *focus group*, los usuarios del sistema ya habían tenido tiempo utilizando el sistema. La sesión fue iniciada indicando que el objetivo de la reunión era recoger la percepción del usuario y las posibilidades de mejora del sistema. Luego se revisó cada interfaz y funcionalidad del sistema y se recogieron los comentarios de los usuarios. Para cerrar el *focus group*, se hicieron preguntas generales del módulo de alertas como: ¿En qué medida el módulo de alertas le ayuda en su gestión ambiental? ¿Qué funcionalidades consideran que falta incorporar en el módulo de manera que mejore su productividad?

Dentro de los comentarios y opiniones de mejora que se recogieron por parte de los usuarios finales, vale la pena mencionar las siguientes:

- **Estaciones favoritas.** Debido a la gran cantidad de estaciones y a que los usuarios del sistema tienen diversos roles en la gestión ambiental, se propuso la mejora de que los usuarios puedan mantener un conjunto de estaciones favoritas, de modo que puedan acceder a las alertas asociadas de manera rápida.
- **Personalizar Formato Correo.** Que cada usuario pueda decidir qué información de las alertas desea que se muestre en el correo. También se desea que se pueda configurar un informe diario o semanal de las alertas detectadas.
- **Integración con Módulo de Dashboard.** el usuario pueda incluir un indicador angular de carga de algún indicador, de manera que cada vez que cargue su *Dashboard* personal, pueda ver información actualizada del indicador.
- **Restricciones de acceso.** Debido a que se espera que *EMIS* posea diversos tipos de usuarios, con diversos privilegios, se desea que el módulo de alertas permita la restricción de acceso a la información de alertas de ciertos indicadores que su información se considera confidencial y no accesible a la totalidad de los usuarios.

Se pudo apreciar que la gran mayoría de los comentarios recibidos son referentes a la manera en que se accede a la información. Los usuarios desean mejoras para visualizar la información de manera más rápida, segura y personalizable. Algunas de las sugerencias descritas anteriormente, son propuestas para ser implementadas en el futuro, lo que puede ser visto en el capítulo 6 *Conclusiones y Trabajo a Futuro*

5. Sistema Implementado

Luego de la construcción del sistema, se procedió a realizar la entrega oficial, para esto se realizaron las siguientes actividades: instalación del módulo de alertas, pruebas y entrega oficial, las que son detalladas en este capítulo.

5.1. Instalación del Módulo de Alertas

La modalidad de desarrollo de EMIS fue realizada de manera iterativa incremental, debido a esto, al momento de instalar el módulo de alertas fue necesario realizar una actualización del sistema, puesto que ya existían otros módulos instalados en calidad de prototipo funcional en los servidores de E-CL.

Para realizar la actualización, se realizó una conexión remota mediante un túnel *ssh* entre las instalaciones de ASYT y los servidores de E-CL ubicados en Tocopilla. Luego de actualizar, tanto el código compilado, como la base de datos, se realizó un recorrido por el sistema para comprobar que las funcionalidades del módulo se ejecutasen sin errores.

Las ilustraciones siguientes muestran capturas de pantalla de la instalación realizada. En la Ilustración 22 se puede apreciar la implementación de la interfaz de visualización de estaciones, los filtros fueron movidos a un panel lateral en el lado derecho (Ilustración 23).

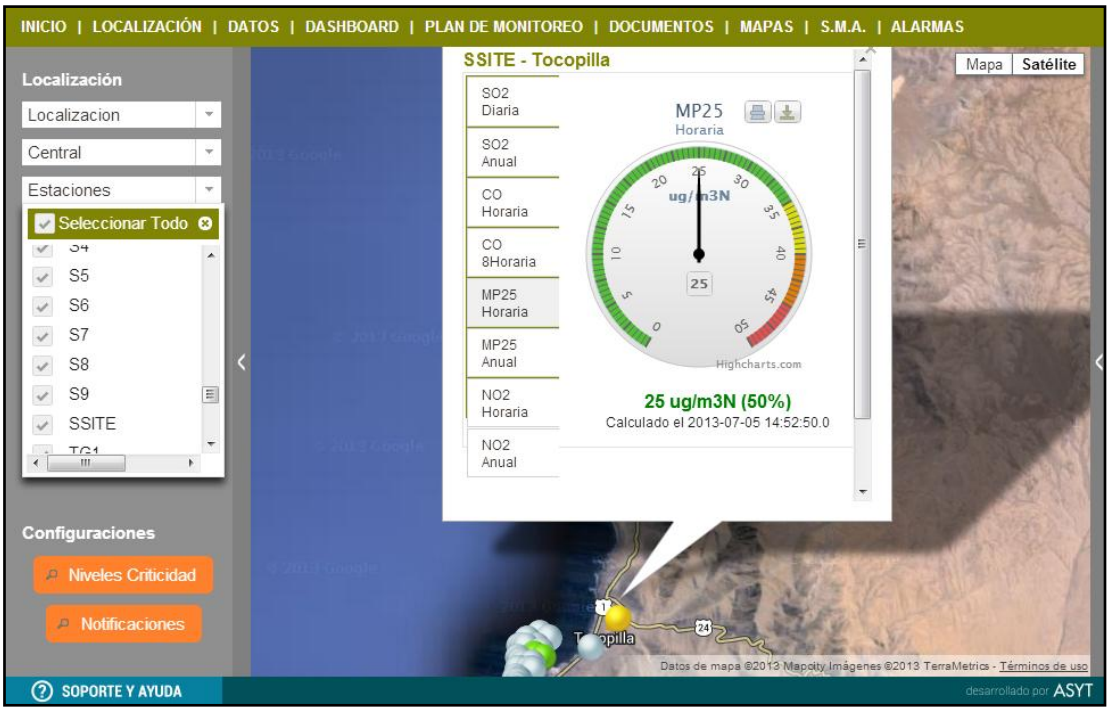

Ilustración 22. Implementación de visualización de estaciones

Debido al gran número de estaciones que posee el sistema, se agregó un buscador en que el usuario selecciona una estación y el mapa se reposiciona en las coordenadas correspondientes.

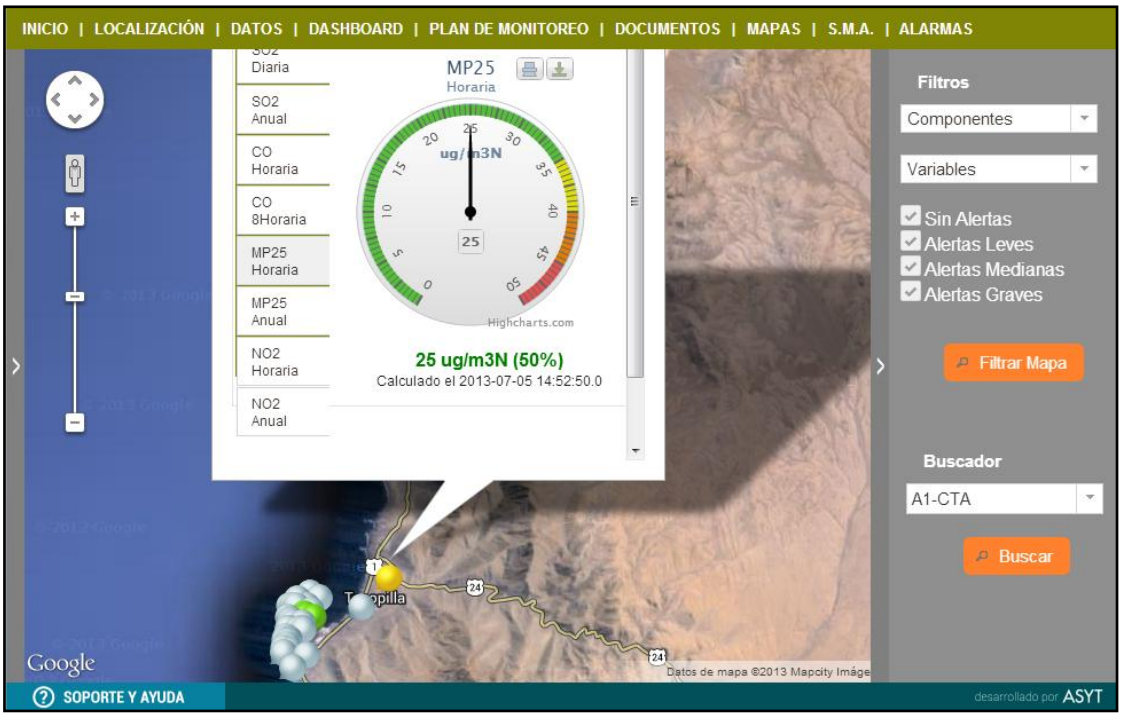

Ilustración 23. Implementación de filtros y buscador de estaciones

Las vistas de configuración de niveles de criticidad y las de configuración de notificaciones no sufrieron cambios en el desarrollo y validación de la solución, por lo que se implementaron tal como fueron diseñadas, ver *3.4 Diseño de la Interfaz de Usuario*.

Las vistas de histórico de alertas no fueron implementadas en este trabajo.

5.2. Reunión de Entrega

Para cerrar la entrega del módulo de datos, se procedió a realizar la entrega del prototipo funcional del módulo. Para esto se viajó a las instalaciones de E-CL ubicadas en Tocopilla para sostener una reunión en la que participaron:

- El memorista.
- Jefe del proyecto EMIS.
- Jefe de medioambiente de la *Central Térmica Mejillones.*
- Gerente de medioambiente de E-CL.
- Encargado de informática de E-CL.

En dicha reunión se realizó un recorrido completo por el sistema EMIS y luego se procedió a realizar una demostración del uso del módulo, en la que se revisaron las interfaces del usuario, se explicó el funcionamiento interno del módulo y como éste interactúa con los otras componentes del sistema. En la Ilustración 24 se puede apreciar una foto de esta reunión.

Posteriormente, fueron entregados los manuales de usuario del módulo de alertas y se realizó una capacitación, la cual consistió en un recorrido guiado por el sistema, donde cada uno de los asistentes contó con un portátil e inició sesión en EMIS para interactuar con las funcionalidades del sistema.

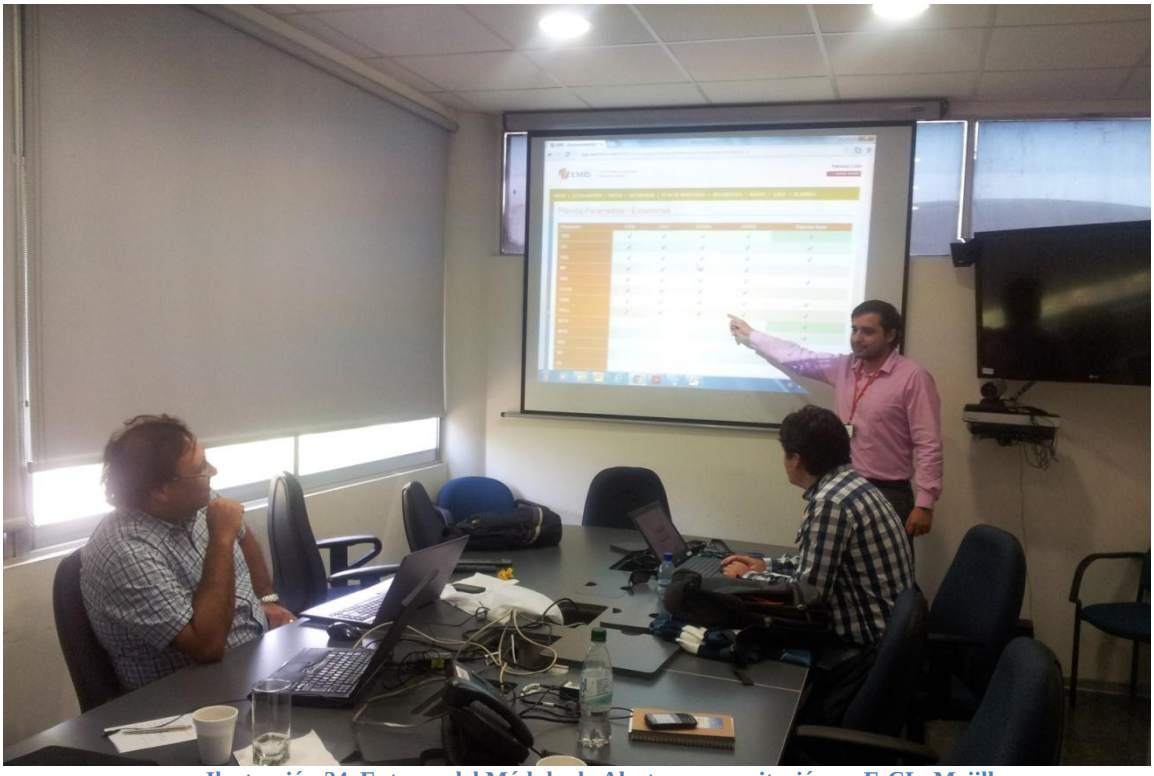

Ilustración 24. Entrega del Módulo de Alertas y capacitación en E-CL, Mejillones.

Finalmente, el sistema quedó instalado como prototipo funcional aprobado por el cliente. En etapas futuras se proyecta realizar una evaluación de performance del módulo y mejoras en la interacción con el usuario, las cuales son descritas en el próximo capítulo.

.

6. Conclusiones y Trabajo a Futuro

Hasta antes de contar con el módulo de alertas, la labor de calcular el estado de cumplimiento de las normativas era una tarea que consistía de muchos pasos y debía ser ejecutada de forma manual. La tarea debía ser realizada por un experto que tuviese conocimiento de los sistemas que registran las mediciones, conocimientos de la normativa vigente y conocimientos matemáticos que le permitiesen realizar los cálculos. Dada la situación anterior, el esfuerzo que demandaba el conocer el estado de cumplimiento de un indicador respecto a la normativa era alto. Considerando la gestión ambiental global de E-CL, el esfuerzo de llevar a cabo esta gestión es aún mayor, siendo proporcional a la cantidad de indicadores medidos, cantidad de normas vigentes y frecuencia.

En relación con el módulo de alertas, se diseñó, construyó e instaló dicho módulo del software EMIS para E-CL. Se logró recopilar la información adecuada para comprender las necesidades del cliente y así lograr definir requisitos de software que las representen. Se logró definir un diseño de software que soporte los requisitos definidos y que permita que el sistema sea mantenible y extensible en el futuro. En cuanto al diseño de las interfaces de usuario, mediante el proceso de validación y modificación de las mismas, se logró que fuesen intuitivas y que presenten la información relevante de manera clara, permitiéndole al usuario realizar las tareas más importantes de manera eficiente.

En cuanto a implementación, se seleccionaron tecnologías existentes de tal manera que permitieran construir la solución diseñada y que ahorraran tiempo de desarrollo. Junto con el uso de buenas prácticas de programación, se produjo una implementación robusta del sistema y que se apega a su diseño original.

Finalmente se logró analizar, diseñar, planificar, construir e implantar exitosamente una solución funcional que cumpla con las necesidades de gestión ambiental del cliente y entregándose en los plazos establecidos.

El desarrollo de la solución ha producido un impacto positivo en E-CL puesto que resuelve la situación mencionada anteriormente centralizando, estandarizando y automatizando la labor de calcular el estado de cumplimiento de la normativa vigente, eliminando barreras tecnológicas y posibles errores en los procedimientos de cálculo. La solución no solo resolvió aquel problema, sino que fue un paso más lejos permitiéndole al usuario ser notificado ante sucesos críticos. En definitiva, el uso del módulo de alertas ha sido un aporte para E-CL ayudándole a enfocarse en los aspectos importantes de su gestión ambiental y toma de decisiones.

La solución desarrollada contempla normativa de Calidad del Aire y Emisiones, sin embargo es extensible para incorporar otro tipo de normativas, como por ejemplo de: RILES (Residuos Industriales Líquidos), aguas, suelos, descargas marítimas, flora y fauna, ruido y olores.

La implementación de estas normativas permitiría la reutilización de EMIS y el módulo de alertas en otro tipo de industrias como: mineras, pesqueras, cementeras, etc. Esta solución podría ser implementada no sólo para empresas y normativas del ámbito Chileno, sino que también podría ser utilizada en otros países, en definitiva en cualquier lugar donde exista una normativa ambiental aplicable y un sistema de monitoreo de los indicadores asociados.

Otra posible aplicación es el uso de esta solución por parte de la autoridad. Como se mencionó en la introducción, actualmente están conformados el *Ministerio del Medio Ambiente*, la *Superintendencia del Medio Ambiente* y el *Tribunal del Medio Ambiente*, encargados generar normativa y de velar por el cumplimiento de la misma. En este contexto, al menos para las centrales termoeléctricas, la legislación vigente exige la implementación de CEMS (*Continious Emissions Monitoring System)* validados de acuerdo a normas internacionales, que midan ciertos indicadores críticos. Debido a lo anterior, se considera que existe tanto el marco legal, como tecnológico, para que un sistema como EMIS y en específico el Módulo de Alertas, pueda ser utilizado por la autoridad para facilitar su labor de fiscalización del cumplimiento de las normas y prevención de eventos riesgosos con potencial impacto en la ciudadanía.

Se propone además extender la actual implementación del sistema para incorporar la interfaz de histórico de alertas (ver 3.4.2). Además, un aspecto interesante para la mejora del software consiste en integrar el Módulo de Alertas con el Módulo de Dashboard. El usuario actualmente en Dashboard tiene la opción de configurar su propio panel, en el que puede agregar diversos tipos de componentes. La idea es desarrollar un componente para Dashboard en el cual el usuario pueda seleccionar un indicador de alguna estación y una normativa específica. Con esto se muestra en su dashboard personal, un detalle del estado actual del indicador respecto a la normativa y el histórico de alertas ocurridas (ilustración 25).

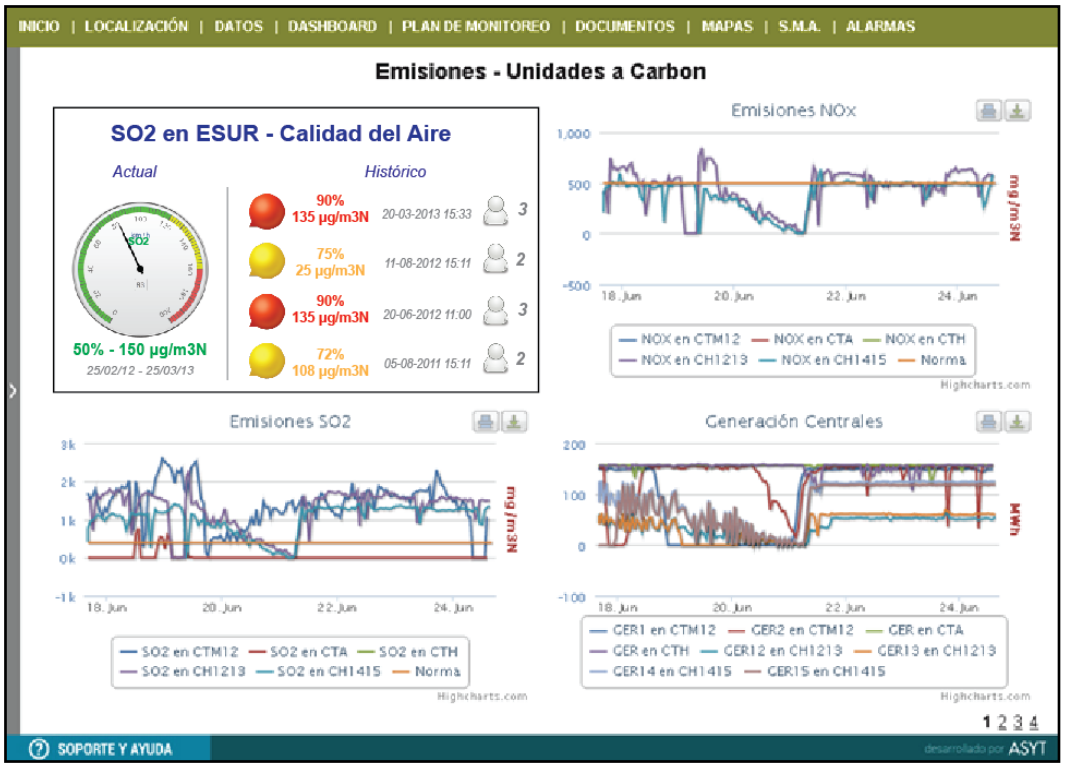

Ilustración 25. Integración con módulo de Dashboard

Se propone además realizar pruebas a la solución implantada en dos aspectos principalmente, la performance de las consultas y la validez de los cálculos realizados debido a datos no válidos. Se considera importante realizar estas actividades en una situación real de la operación del software puesto que las condiciones de producción podrían diferir mucho de las de desarrollo.

En cuanto a performance, se propone realizar pruebas de rendimiento de cálculo de normativa similares a las descritas en *4.2 Evaluación de Performance* en los servidores de E-CL, de manera de determinar y resolver posibles problemáticas futuras. Se propone también evaluar el uso de herramientas *NoSQL*, que son sistemas de gestión de bases de datos no RDBMS (no relacionales), en el que muchos de ellos están optimizados para consultar en tiempo real cálculos complejos de grandes volúmenes de información, ejemplos de estos sistemas son: *HBase, MongoDB, Hypertable, Accumulo*.

En cuanto a la validez de los cálculos realizados, se propone realizar una revisión de los datos recolectados en EMIS referente a los indicadores que se ven afectados por normativas, en busca de posibles eventos que pudiesen alterar la validez de las alertas. En particular se busca:

- **Valores Fuera de Rango.** Este evento podría darse por mal funcionamiento de algún equipo de monitoreo. Por ejemplo, las concentraciones habituales medidas de SO2 no debieran superar los 80 ug/m3N, un valor de 2000 ug/m3N podría ser considerado fuera de rango.
- **Valores Cero.** Muchas veces existen por problemas de comunicación o desconexión de equipos, sean estos eventos programados o no. Ante esta situación podrían llegar valores cero (o cercanos al cero) no debieran ser registrados en el sistema.

La ocurrencia repetida de alguno de estos eventos podría llevar a que se levanten falsas alertas o a que no se levanten otras que si debieran, por lo que es de suma importancia detectar y manejar correctamente estos eventos.

7. Bibliografía y Referencias

1. *"ESA Software Engineering Standards"*. PSS-05-0 Issue 2. ESA Board for Software Standardization and Control (BSSC) - European Space Agency. (1991). URL: www.ess.co.at/ECOSIM/ESA.txt
2. Roger S. Pressman.*Software Engineering: A Practitioner's Approach.*7th Ed. McGraw Hill. (2009).
3. Ian Sommerville. *Software Engineering.* 7th Ed. Addison-Wesley. (2004).
4. Clarence Ho, RobHarrop. *Pro Spring 3.* 1st Ed. Apress. (2012).
5. Giulio Zambon. Begining JSP, JSF and Tomcat: Java Web Development.Apress (2012).
6. Oleg Varaksin, Mert Caliskan. Primefaces Cookbook. Packt Publishing (2013).

Anexo A. Requisitos de Usuario

A continuación se listan los requisitos de usuario identificados en el presente trabajo.

RU001 – Alertas para módulo de DATOS	
Necesidad	El sistema debe permitir al usuario definir alertas para cuando algún parámetro que se mida en el *Módulo de Datos de EMIS* salga de la norma.
Fuente	E-CL
Prioridad	Alta
Estabilidad	Alta

RU004 – Notificación por correo electrónico	
Necesidad	Al levantarse una alerta, el usuario asociado podrá recibir la notificación mediante correo electrónico, según se haya configurado (RU007)
Fuente	E-CL
Prioridad	Alta
Estabilidad	Media
Req. Relacionados	RU007

RU005 – Notificación por SMS	
Necesidad	Al levantarse una alerta, el usuario asociado podrá recibir la notificación mediante mensaje SMS en su teléfono móvil, según se haya configurado (RU007)
Fuente	Consultora ASYT
Prioridad	Baja
Estabilidad	Media
Req. Relacionados	RU007

RU006 – Notificación en interfaz web	
Necesidad	Al levantarse una alerta, el usuario asociado podrá recibir la notificación al momento de iniciar sesión en el sistema Emis
Fuente	Consultora ASYT
Prioridad	Media
Estabilidad	Alta

Req. Relacionados	RU007

RU007 – Notificaciones a usuarios	
Necesidad	El usuario administrador del módulo podrá, por cada alerta, definir a qué usuarios notificar pudiendo seleccionar usuarios individuales. Por cada una de estas opciones el usuario podrá seleccionar por qué métodos se notificará.
Fuente	E-CL, Consultora ASYT
Prioridad	Alta
Estabilidad	Media
Req. Relacionados	RU004, RU005, RU006

RU008 – Tiempo envío notificaciones	
Necesidad	Al levantarse una alerta, el sistema tendrá a lo más un minuto para realizar el envío de las notificaciones correspondientes
Fuente	ASYT Consultora
Prioridad	Alta
Estabilidad	Baja

RU009 – Contenido de una notificación	
Necesidad	Al enviarse una notificación, esta debe contener información que le permita al receptor conocer la real dimensión del inconveniente y su origen. La información debe ser la suficiente como para que el usuario notificado pueda tomar alguna medida de acción.
Fuente	ASYT Consultora
Prioridad	Alta
Estabilidad	Baja

RU010 – Alertas de Normativa Vigente	
Necesidad	Las alertas del módulo de datos deben estar referidas a la normativa vigente de Calidad del Aire y Emisiones.
Fuente	ASYT Consultora
Prioridad	Alta

Estabilidad	Alta
Req Asociados	RU001

RU011 – Histórico de Alertas	
Necesidad	Todas las alertas detectadas en el sistema deben ser guardadas en un histórico.
Fuente	E-CL
Prioridad	Alta
Estabilidad	Alta
Req Asociados	RU001

Anexo B. Requisitos de Software

A continuación se listan los requisitos de software identificados a partir de los requisitos de usuario e información recopilada.

RS001 – Conexión con Módulo de Datos	
Descripción	Las alertas detectadas en el sistema deben ser calculadas en base a la información recogida por el Módulo de Datos, en específico por el Servicio de Recolección de Datos.
Tipo	Funcional
Prioridad	Alta
Estabilidad	Alta

RS002 – Tiempo de envío de notificaciones	
Descripción	Las notificaciones deberán ser enviadas (no necesariamente recibidas) como valor óptimo 30 segundos después de detectada la alerta, un máximo 90 segundos es aceptable. Esto debe ocurrir para el 90% de los casos.
Tipo	Performance
Prioridad	Alta
Estabilidad	Media

RS003 – Visualización en Mapa	
Descripción	Las alertas deben ser visualizadas en un mapa interactivo, en el que se muestren marcadores indicando las estaciones. Se deberá marcar cada alarma, de cada estación, con un color que identifique sus tipo.
Tipo	Interfaz
Prioridad	Media
Estabilidad	Alta

RS004 – Carga de un Indicador	
Descripción	El cumplimiento de la normativa debe ser llevada a una cuantificación. Se define la carga de un indicador respecto a una normativa dada, como el porcentaje del valor medido respecto al máximo establecido por la normativa.
Tipo	Funcional
Prioridad	Media
Estabilidad	Alta

RS005 – Configuración de cálculo de normativas	
Descripción	La forma en que se realizarán los cálculos de cada normativa y los indicadores que cada una de ellas tendrá asociados, vendrán precargados en el sistema. El usuario no tendrá que configurar ningún cálculo.
Tipo	Funcional
Prioridad	Alta
Estabilidad	Alta

RS006 – Niveles de Criticidad	
Descripción	Los niveles de criticidad que podrá tener cada alerta corresponden a Leve, Mediano y Grave, estando determinados cada uno por un porcentaje de carga (RS005). El sistema tendrá por defecto para Leva 70%, para Mediano 80% y para Grave 90%.
Tipo	Funcional
Prioridad	Alta
Estabilidad	Media

RS007 – Configuración de Niveles de Criticidad	
Descripción	Un usuario administrador del sistema podrá configurar por cada normativa, los tres niveles de criticidad asociados, indicando un porcentaje de carga asociado.
Tipo	Funcional
Prioridad	Alta
Estabilidad	Media

RS008 – Frecuencia de cálculo	
Descripción	La frecuencia con que se actualizará la carga de un indicador respecto a una normativa, será igual a la agregación que defina la misma. Por ejemplo, si una normativa considera los máximos diarios de un cierto contaminante, la frecuencia de cálculo será diaria.
Tipo	Funcional
Prioridad	Alta
Estabilidad	Alta

RS009 – Periodos móviles	
Descripción	Todas las reglas de cálculo de la normativas deben ser consideradas utilizando periodos móviles en vez de calendarios.
Tipo	Funcional
Prioridad	Media

Estabilidad	Alta

RS010 – Configuración de notificaciones	
Descripción	Un usuario administrador del sistema podrá configurar a qué usuarios se deberá notificar ante la detección de alertas Leves, Medianas o Graves, los siguientes niveles son posibles: • **Nivel Global.** Considera todas las alertas detectadas. • **Nivel Normativa.** Considera las alertas detectadas respecto a una normativa en específico. • **Nivel Estación.** Considera las alertas detectadas respecto los indicadores que están siendo medidos en una determinada estación.
Tipo	Funcional
Prioridad	Media
Estabilidad	Alta

RS011 – Persistencia de alertas	
Descripción	Cada vez que se detecte una alerta en el sistema, se deberá persistir un histórico de la ocurrencia indicando: valor medido, nivel de criticidad, fecha, normativa e indicador asociados.
Tipo	Funcional
Prioridad	Alta
Estabilidad	Alta

RS012 – Persistencia de cálculos	
Descripción	Cada vez que se actualice el estado de carga de un indicador respecto a una normativa, se deberá persistir un histórico indicando: valor medido, fecha, normativa e indicador asociados.
Tipo	Funcional
Prioridad	Alta
Estabilidad	Alta

RS013 – Alertas duplicadas	
Descripción	Las alertas duplicadas no deben ser persistidas. Se considera duplicada cuando: • No hay nuevos datos en el sistema y ya se ha registrado una alerta idéntica (con igual valor, fecha y hora, indicador, normativa) en el sistema. • Corresponde nuevamente actualizar el estado del indicador y se detecta una alerta del mismo nivel que la anterior y de manera

	consecutiva. Se asume que es parte del mismo evento y no representa una nueva alerta.
Tipo	Funcional
Prioridad	Alta
Estabilidad	Alta

RS014 – Cálculos duplicados	
Descripción	Los cálculos duplicados no son persistidos, si no hay nuevos datos de mediciones en el sistema para algún indicador, no se debe recalcular el estado de cumplimiento respecto a ninguna normativa.
Tipo	Funcional
Prioridad	Media
Estabilidad	Alta

RS015 – Envío de notificaciones	
Descripción	Una vez detectada una alerta, serán enviadas notificaciones mediante correo electrónico a los usuarios que aplique, según las reglas de notificación (RS010). El correo debe contener información acerca de: indicador, estación, normativa, valor medido, criticidad alerta.
Tipo	Funcional
Prioridad	Media
Estabilidad	Alta

RS016 – Programación de cálculos automáticos	
Descripción	El estado de cumplimiento de cada indicador respecto a cada normativa debe ser ejecutado mediante alguna tarea programada, es decir automáticamente. Los cálculos deben ser programados según se establece en RS008
Tipo	Funcional
Prioridad	Alta
Estabilidad	Media

RS017 – Visualización de Histórico	
Descripción	El usuario podrá consultar el histórico de alertas en la misma interfaz definida en RS003, se mostrará un listado con las ultimas alertas detectadas en el sistema por cada estación en el mapa. El listado estará ordenado por fecha de ocurrencia de la alerta. El listado indicara: valor medido, valor porcentual (carga), criticidad de la alerta, fecha de ocurrencia y usuarios notificados.
Tipo	Interfaz

Prioridad	Alta
Estabilidad	Media

RS018 – Visualización con indicadores angulares	
Descripción	En la visualización de estaciones, el estado de cada indicador respecto a cada normativa debe ser representado con un indicador angular (como un tacómetro), en el que se deben indicar con colores los niveles de criticidad.
Tipo	Interfaz
Prioridad	Baja
Estabilidad	Media

Anexo C. Módulos de EMIS

A continuación se describen brevemente los módulos que posee EMIS, incluyendo capturas de pantalla de los mismos.

C.1 Módulo de Localización

Permite la visualización georeferenciada (Ilustración 26) de datos de emisión de contaminantes o variables meteorológicas, provenientes de diversas fuentes, ya sean estaciones de monitoreo de calidad del aire, chimeneas u otras. Este módulo permite tener cada fuente de datos categorizada por diversos descriptores como Componente (Medio Marino, Medio Físico, Unida de Proceso, etc), Variable (Calidad del Aire, Emisiones, Medio Ambiente Marino), entre otros. Los datos pueden ser vistos en forma de gráficos y en tiempo real, teniendo diversas opciones como: agregación, horizonte temporal, tipo de gráfico, múltiples ejes, función de agregación, etc.

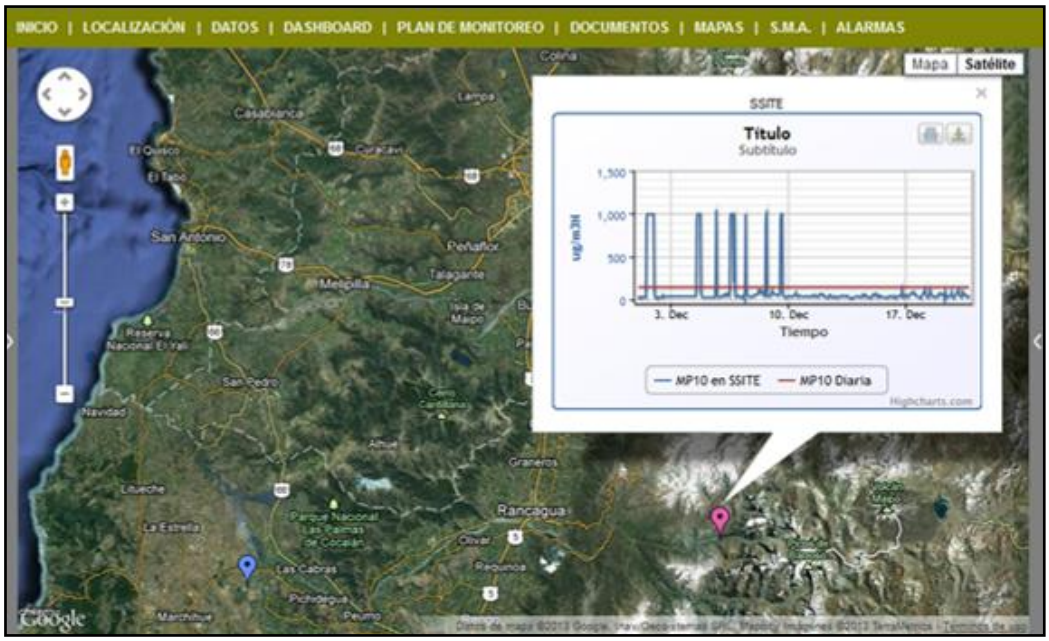

Ilustración 26. Módulo de Localización

C.2 Módulo de Datos

Este módulo permite visualizar (Ilustración 27) combinaciones de datos provenientes de diversas fuentes de medición y diversos parámetros. Posee filtros inteligentes que le permiten al usuario encontrar los parámetros que busca de manera fácil. Datos ofrece la flexibilidad de visualizar combinaciones de parámetros de diferente tipo (meteorológicos (Ilustración 29), emisiones, calidad del aire) y de diversas fuentes en un sólo gráfico o tabla (Ilustración 28), los que pueden ser posteriormente exportados como imagen o como archivos Excel. Por otra parte éste módulo permite combinar los datos de dirección y velocidad del viento para generar rosas de viento, con precisión de hasta minutos.

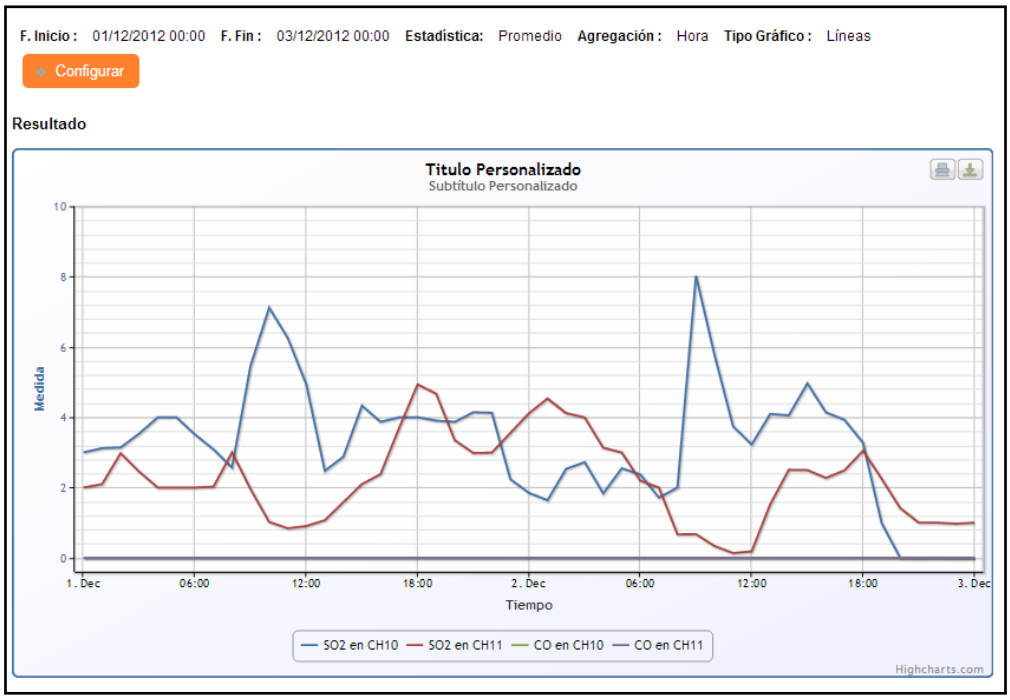

Ilustración 27. Gráfico para diversos indicadores, Módulo de Datos

Tiempo	SO2 en CH10 mg/m3N	SO2 en SSITE ug/m3N	CO en CH11 mg/m3N
01/12/2012 00:00	3.0	103.2	0.0
01/12/2012 01:00	3.1	77.0	0.0
01/12/2012 02:00	3.1	53.9	0.0
01/12/2012 03:00	3.5	68.3	0.0
01/12/2012 04:00	4.0	23.3	0.0
01/12/2012 05:00	4.0	124.8	0.0
01/12/2012 06:00	3.5	106.2	0.0
01/12/2012 07:00	3.1	73.7	0.0
01/12/2012 08:00	2.6	144.3	0.0
01/12/2012 09:00	5.5	69.9	0.0
01/12/2012 10:00	7.1	35.5	0.0
01/12/2012 11:00	6.3	24.3	0.0
01/12/2012 12:00	4.9	112.0	0.0
01/12/2012 13:00	2.5	438.4	0.0
01/12/2012 14:00	2.9	44.7	0.0
01/12/2012 15:00	4.3	8.9	0.0
01/12/2012 16:00	3.9	8.9	0.0
01/12/2012 17:00	4.0	8.9	0.0
01/12/2012 18:00	4.0	8.9	0.0
01/12/2012 19:00	3.9	8.9	0.0

(1 of 10) 1 2 3 4 5 6 7 8 9 10 5

Ilustración 28. Tabla de valores medidos, Módulo de Datos

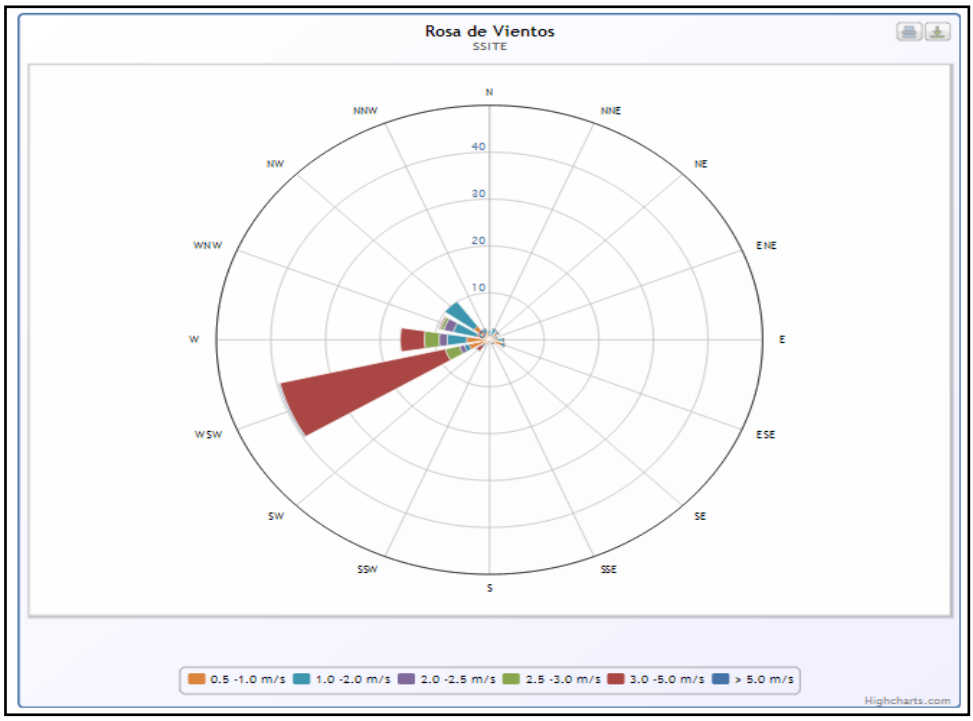

Ilustración 29. Rosa de vientos, Módulo de Datos

C.3 Módulo de Mapas

Este módulo permite al usuario generar modelos científicos de dispersión de contaminantes en tiempo real y visualizados en una vista satelital de la ubicación de la fuente. Estos modelos combinan datos meteorológicos, como velocidad y dirección del viento, con datos de contaminantes para generar las visualizaciones que se detallan a continuación.

- **Rosa de Contaminantes.** Permite visualizar las concentraciones registradas según la dirección del viento, mediante rangos que se calculan en base a las mediciones horarias del contaminante. Ver Ilustración 30.
- **Rosa Polar.** Permite visualizar las concentraciones registradas de un contaminante incorporando velocidad del viento (componente radial), dirección (componente angular) y concentración (color). Ver Ilustración 31.
- **Clustering.** Permite visualizar una partición del espacio indicando zonas que matemáticamente son significativamente distintas. Ilustración 32.
- **Anillo Temporal.** Permite visualizar las concentraciones registradas de un contaminante, por dirección (componente angular), por hora del día (componente radial) y valor registrado (color). Ver Ilustración 33.

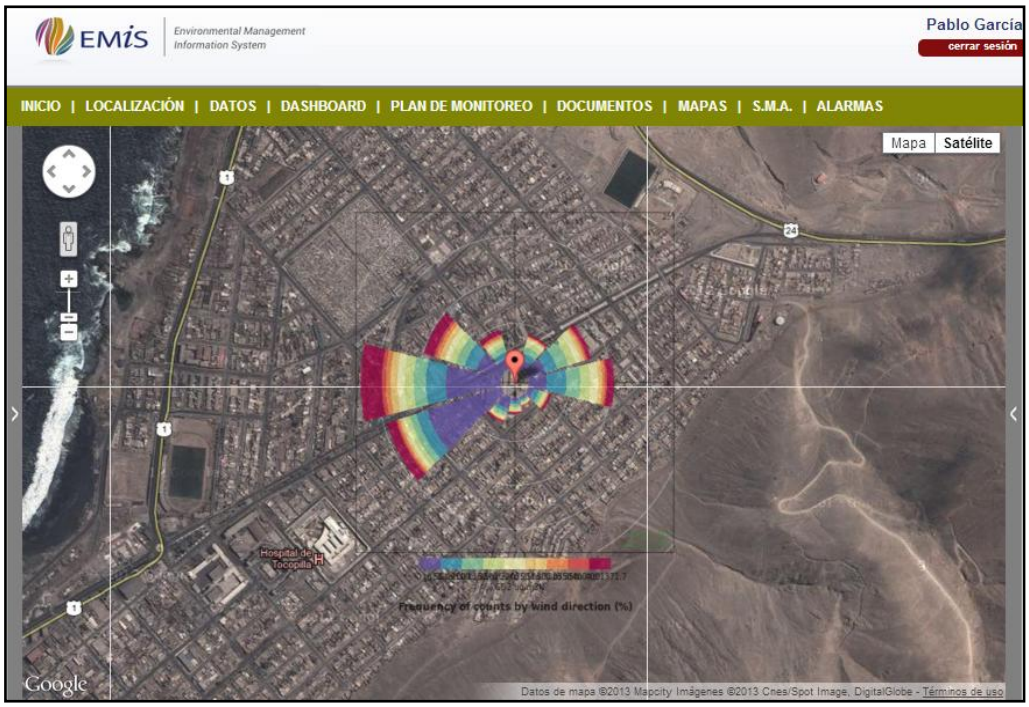

Ilustración 30. Rosa de contaminantes

Ilustración 31. Rosa polar, Módulo de Mapas

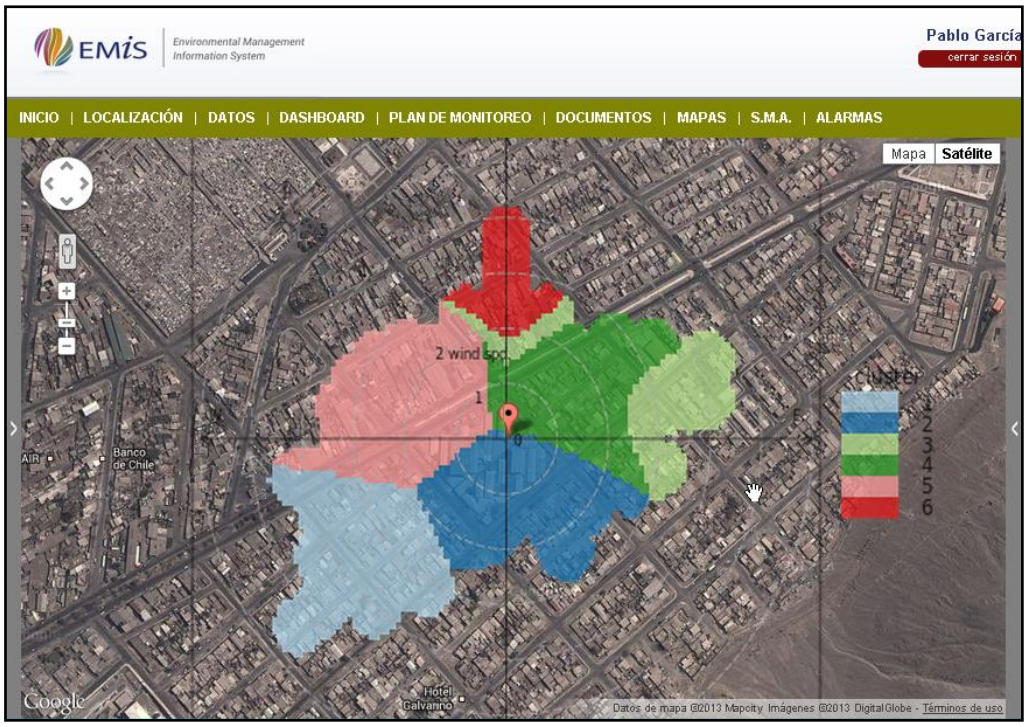

Ilustración 32. Clustering, Módulo de Mapas

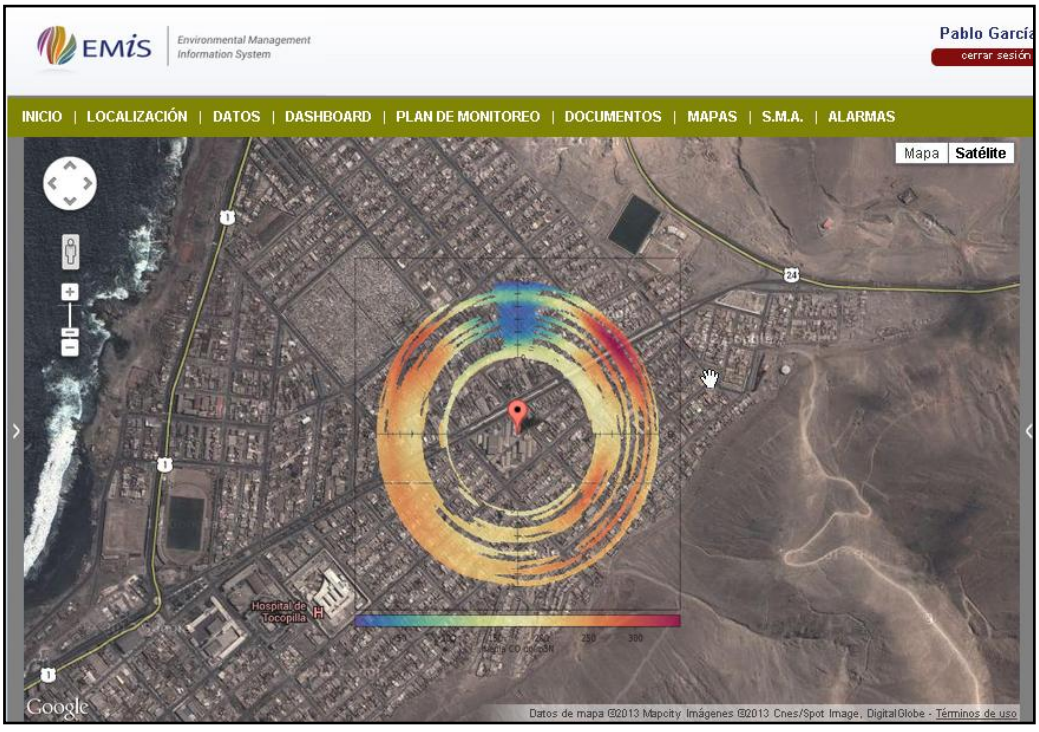

Ilustración 33. Anillo temporal, Módulo de Mapas

C.3 Módulo de Dashboard

Permite la visualización de indicadores de gestión predefinidos que se actualizan cada minuto. El usuario podrá consultar un Dashboard global, o bien crear su propio Dashboard personal, en el él mismo podrá definir sus indicadores de gestion.

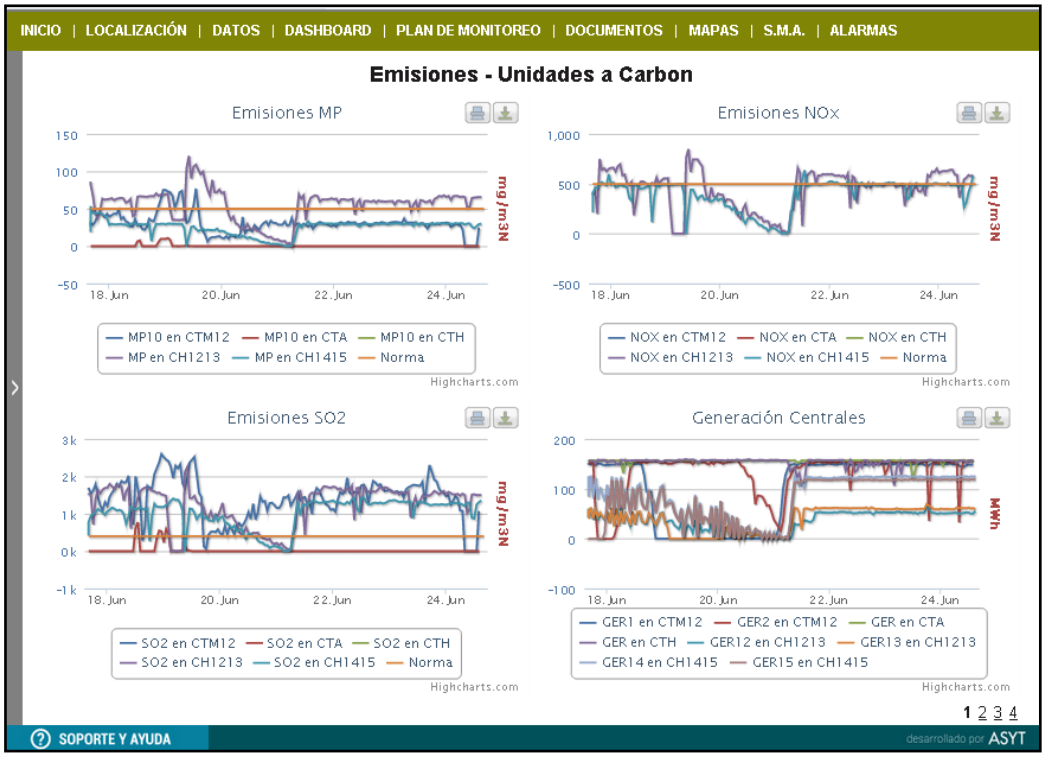

Ilustración 34. Módulo de Dashboard

C.3 Módulo de S.M.A

El *Módulo de S.M.A* es una herramienta de apoyo a la gestión del cumplimiento de las RCAs (Resolución de Calificación Ambiental) determinadas por la Superintendencia de Medio Ambiente. S.M.A permite al usuario de *EMIS* definir actividades para cada exigencia y pudiendo conocer el estado de cumplimiento de sus actividades asociadas (Ilustración 35). Por último, S.M.A permite al usuario manejar un repositorio de documentos legales relacionados con sus RCAs.

Ilustración 35. Listado de RCAs, Módulo de S.M.A.

C.3 Módulo Plan de Monitoreo

El Módulo Plan de Monitoreo es una herramienta de planificación y control de la gestión del cumplimiento de los acuerdos tomados con las autoridades locales, además de tareas operativas internas.

Este módulo funciona en base a "planes" los cuales están compuestos por actividades que son realizadas mensualmente, a las cuales se les debe asociar un estado de cumplimiento, además de la evidencia correspondiente.

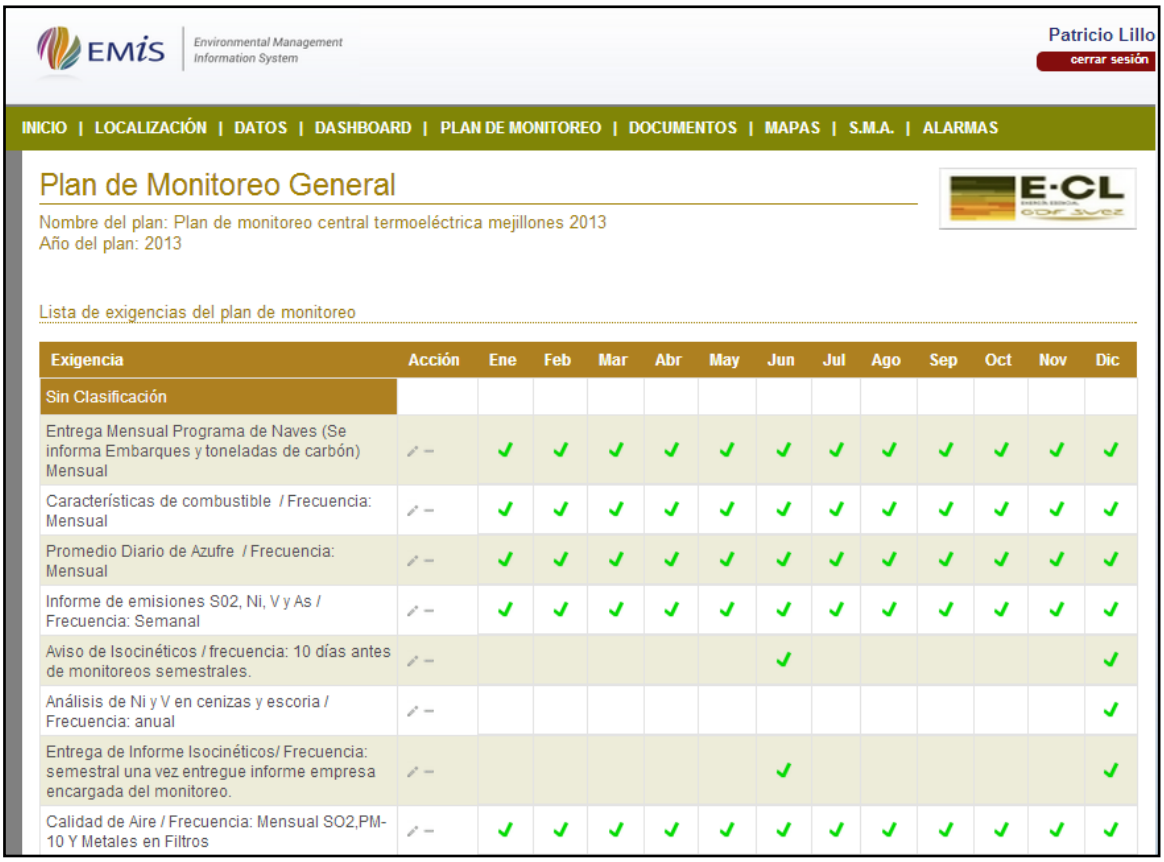

Ilustración 36. Módulo Plan de Monitoreo

Anexo D. Tecnologías Utilizadas en EMIS

La arquitectura base de EMIS utiliza *Apache Maven* y Spring como tecnologías base para la definición de cada capa e interacción entre ellas. En este anexo se listan las tecnologías utilizadas en el desarrollo de EMIS y el Módulo de Alertas. En la siguiente ilustración se puede apreciar un diagrama con las principales tecnologías utilizadas por cada capa.

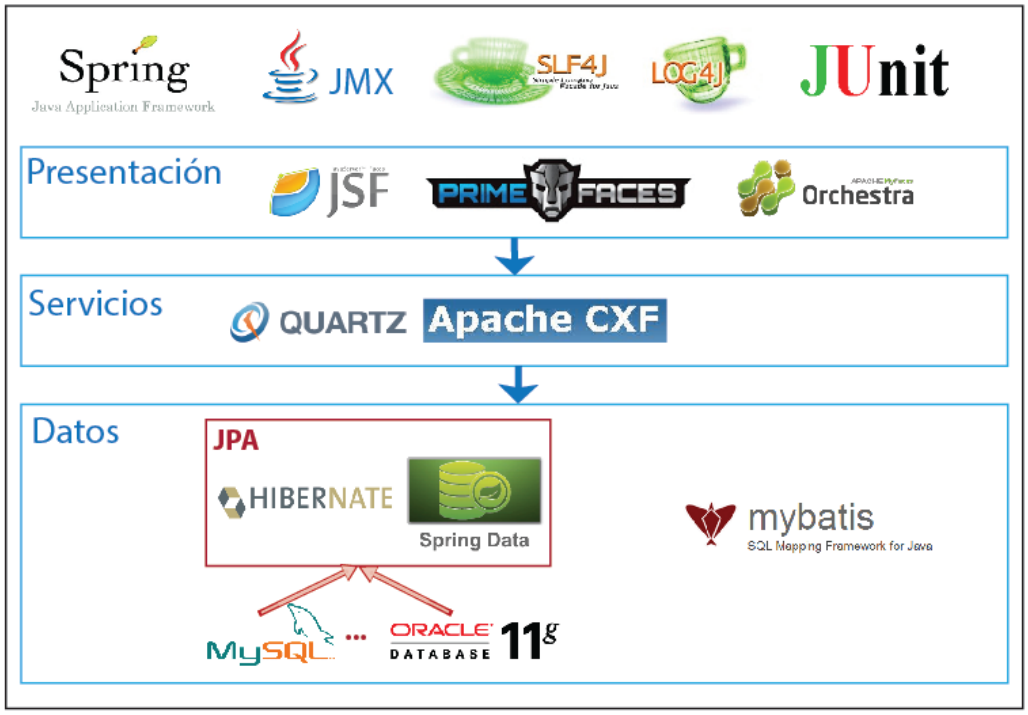

Ilustración 37. Principales tecnologías utilizadas en EMIS

- **Spring Framework.** Permite acoplar diversas componentes de un software de manera fácil utilizando IOC (*Inversión de Control)*, posee integración con diversas tecnologías (por ejemplo , Hibernate, Quartz), posee diversos módulos como seguridad, integración con redes sociales , acceso a datos.
  Todo lo anterior conversa plenamente con los estándares Java de datos (como JPA), de mensajería (como JMS), etc, además, esta tecnología es ampliamente utilizada en el sector empresarial alrededor del mundo y posee una gran comunidad que la soporta.
  Más información en http://www.springsource.org/

- **JMX (Java Management Extensions).** API para la monitorización y administración de aplicaciones JAVA. Es útil para poder hacer tracking de aplicaciones que están en producción. Con esto y través de los MBeans es posible conectarse remotamente a una JVM para observar la ejecución de una aplicación y poder controlar su funcionamiento.
  Más información en http://docs.oracle.com/javase/tutorial/jmx/

- **SLF4J y LOG4J.** SLF4J es una API para logging y LOG4J es una implementación de ella. Permite hacer logging de manera fácil con distintos niveles (INFO, ERROR, TRACE, WARNING, ETC). Una ventaja es que el logging puede ser activado o

desactivado en tiempo de ejecución, de manera de que éste no sea permanente y afecte la performance del sistema.

Más información en http://logging.apache.org/log4j/1.2/

- **JUnit.** Sirve para crear tests unitarios, los cuales permiten evaluar que el funcionamiento de las clases sea adecuado.

  Más información en http://junit.org/

- **JPA y Hibernate.** Hibernate, como implementación de la Java Persistence API, permite hacer un mapeo de clases o POJOs a tablas en una base de datos. Posee soporte para múltiples motores de bases de datos, tanto relacionales como NoSQL (Hibernate OGM). Hibernate se destaca por su flexibilidad y las amplias opciones que tiene para la generación de consultas, además de ser muy popular en el mercado y ser conocido por su robustez.

  Más información en http://www.hibernate.org/

- **Spring Data.** Permite crear DAOS de manera simple, de modo de que no es necesario escribir o implementar la lógica de consultas simples. Simplemente se definen en una interfaz y spring-data se encarga de inyectar la implementación correspondiente en tiempo de ejecución. Por otra parte, Spring Data permite exponer fácilmente la capa de acceso a datos como servicios REST.

  Más información en http://www.springsource.org/spring-data

- **MyBatis.** Muy útil para encapsular el acceso a datos cuando se tienen una diversidad de procesos almacenados y consultas específicas. Se recomienda utilizarlos cuando se deban vincular bases de datos legacy.

  Más información en https://code.google.com/p/mybatis/

- **QUARTZ.** Es una biblioteca que permite programar tareas utilizando *CRON Expressions*, tiene integración directa con Spring Framework. La gran ventaja es que es muy simple de utilizar gracias a sus anotaciones @Scheduled y @Async.
  Es de utilidad para la capa de servicios puesto que permite agendar tareas batch o ejecutar cierta lógica pesada gatillada por el usuario de manera asíncrona, sin que se vea afectado el tiempo de respuesta al usuario final.

  Más información en http://quartz-scheduler.org/

- **Apache CXF.** Permite exponer la lógica de la aplicación en servicios de diversos tipos como: SOAP, XML/HTTP, RESTful HTTP o CORBA. Además de soportar diversos protocolos de transporte como HTTP, JMS o JBI.

  Más información en http://cxf.apache.org/

- **Java Server Faces.** Framework que permite construir aplicaciones web, se basa en el concepto de componentes, que pueden ser referenciados en los templates XHTML. JSF se caracteriza por la rapidez y confiabilidad con que se pueden desarrollar las aplicaciones. Al ser un estándar, existen diversas implementaciones disponibles para utilizar.

  Más Información en http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html

- **PrimeFaces.** Es una biblioteca que contiene una gran variedad de componentes JSF para la interacción con el usuario, como acordiones, tabs, gráficos, mapas, además de los componentes típicos como cuadros de texto, checboxes, etc. PrimeFaces tiene la gran ventaja de que los componentes son referenciados tanto en el template como en el código java, por lo que los datos que recibe o muestran los componentes, son tratados puramente como objetos java.

  Más información en http://www.primefaces.org/

- **Myfaces Orchestra.** Corresponde a una biblioteca que implementa el llamado "*Conversation Scope*" para los *Beans JSF***,** el cual permite mantener viva una versión del PersistenceContext en cada conversación.

  Más información en http://myfaces.apache.org/orchestra/