UNIVERSIDAD NACIONAL DE CÓRDOBA (UNC)
COMISIÓN NACIONAL DE ACTIVIDADES ESPACIALES
(CONAE)

Geomática aplicada a un Sistema de Alerta Temprana

MASTER IN SPACE APPLICATIONS FOR EMERGENCY EARLY WARNING AND RESPONSE

Autor: **Gonzalo Sebastian Peralta**
Director: Mario Lamfri

Córdoba - Argentina
Julio del 2011

I would like to dedicate this thesis to my loving parents ...

Resumen

En el presente trabajo se plantea una Infraestructura Informática para dar soporte a la prevención y el control estratégico del vector del dengue en Argentina. La misma es parte de un complejo Sistema de Alerta Temprana (SAT) y es llevado a cabo por la Comisión Nacional de Actividades Espaciales (CONAE) bajo los estándares de la European Spatial Agency (ESA). La arquitectura, diseño, metodología y codificación pretenden ser componentes re-utilizables en cualquier infraestructura informática de soporte a SATs. En su desarrollo se utiliza Open Source Software (OSS) y Patrones de Diseño (Design Patterns) garantizando una herramienta tecnológica además de re-utilizable, flexible, mantenible y robusta. En el capítulo 1 se presenta una introducción acerca de la problemática del dengue y los beneficios de epidemiología panorámica aplicada al desarrollo de este trabajo. Capítulo 2 se describen los requerimientos de usuario y sistema al inicio del desarrollo. El Capitulo 3 presenta la arquitectura y diseños adoptados en la resolución del problema. El capítulo 4 es un resumen de las tecnologías investigada para la implementación de la solución. El capítulo 5 muestra los detalles de implementación mientras que el capítulo 6 describe la funcionalidad obtenida. Finalmente las conclusiones son abordadas en el capítulo 7.

Indice

Indice de figuras

Capítulo 1

Introducción

1.1. Motivación: Dengue en la Argentina

El Dengue es en la actualidad una de las enfermedades transmitidas por vectores de mayor prevalencia en el continente Sudamericano [Jennifer and Droll, 2001]. A partir de la década del 70 y en no más de 20 años, la región pasó a una situación de hiper-endemia, con circulación simultánea de hasta 4 serotipos del virus del dengue, Existen en el continente epidemias frecuentes con incremento significativo en el número de casos, reportándose notificaciones en su forma más grave: el Dengue Hemorrágico [Bejarano, 1979; Carcavallo and Martinez, 1968]. El agente causal de la enfermedad del dengue es un arbovirus[1] miembro del género Flaviviridae, el mismo es transmitido al hombre (agente susceptible), a través del mosquito Aedes aegypti principal vector en el continente sudamericano. Esta patología puede ser causada por uno de cuatro serotipos diferentes (DEN 1, 2, 3, 4), la infección con alguno de estos serótinos provee inmunidad contra su homólogo, pero no así para los tres restantes, desencadenando en su re-infestación la forma hemorrágica de la enfermedad [Estrada and Craig, 1995; Kouri, 1998]. Como todas las enfermedades de transmisión vectorial, el Dengue es un sistema complejo y dinámico en tiempo y espacio [Delgado and Ramos, 2007]. Múltiples factores ambientales tanto de carácter biofísico como social constituyen una compleja trama que condiciona o determina la proliferación del vector-enfermedad [Hales et al., 2002; Tauil, 2001; WHO, 2009]. En este sentido, factores bioclimáticos como demográficos y antrópicos actúan sobre las poblaciones de insectos, siendo las condiciones climáticas el factor regulador de la distribución espacio temporal de las poblaciones de artrópodos[2] y su abundancia [Micieli and Campos, 2003]. Cambios en el régimen de precipitaciones, en los niveles de humedad o en los valores de temperatura afectan la biología y ecología de los vectores. Por lo tanto existen diversos factores ambientales que actúan en diferente medida en la problemática del dengue, su re-emergencia, y difusión actual. Actualmente, los límites de distribución de la especie son la frontera norte del país, Santa Rosa (Provincia de La Pampa) al sur [Rossi et al., 2006] y el Departamento de Guaymallén en la provincia de Mendoza [Dominguez and Lagos, 2001]. (ver figura 1.1).

En el primer semestre del año 2009 se produjo una epidemia de dengue por el serotipo 1 que

[1] describe la forma particular de transmisión de estos virus a través de en un artrópodo hematófago, el que actúa como vector.

[2] animales que por sus caracteres morfológicos presentan patas y antenas.

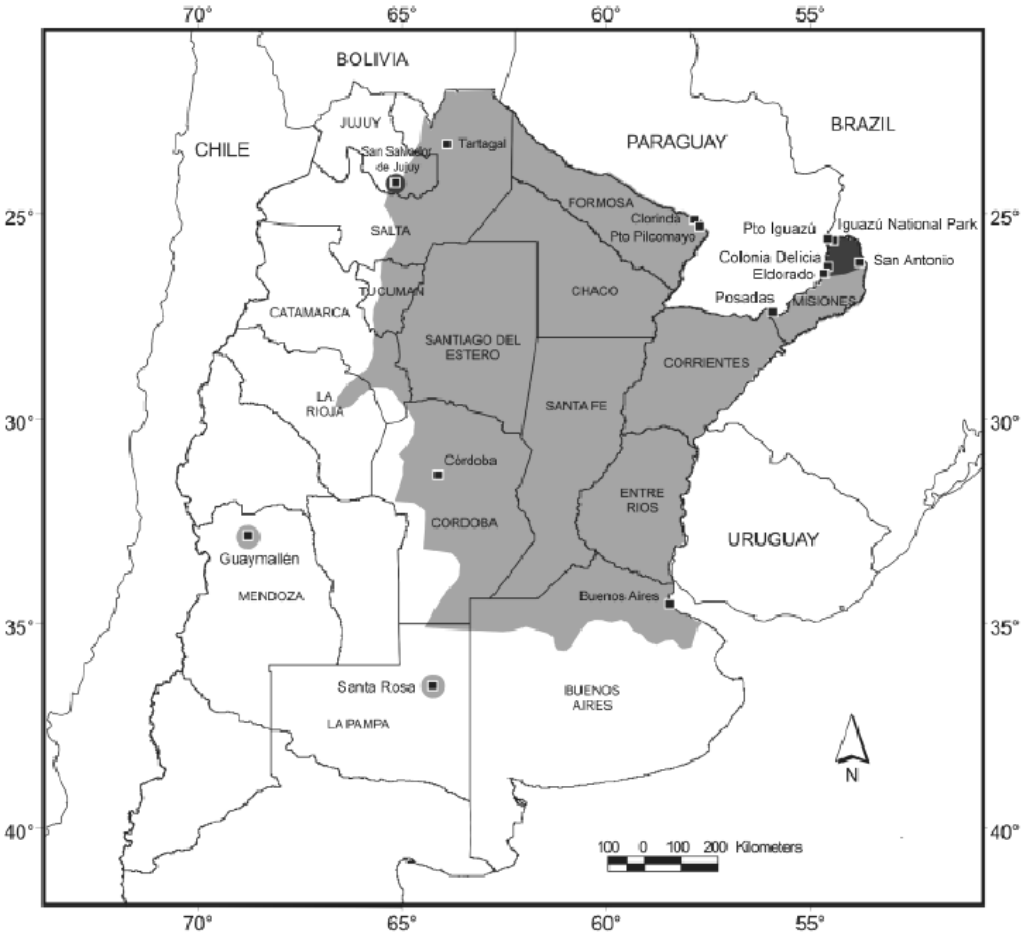

Figura 1.1: Distribución Geográfica estimada de Aedes aegypti (gris claro) y Aedes albopictus (gris oscuro) para el año 2008. Extraído de Vezzani y Carbajo (2008).

desde Salta y Jujuy se extendió hacia el sur y al este, llegando hasta el paralelo 35º. El total de casos confirmados llegó a 25.989, en catorce jurisdicciones (Provincias de Buenos Aires, Catamarca, Córdoba, Chaco, Entre Ríos, La Rioja, Santa Fe, Santiago del Estero, Tucumán, Salta, Jujuy, Corrientes, Misiones y la Ciudad Autónoma de Buenos Aires), once de las cuales no habían registrado nunca casos autóctonos de la enfermedad[Ministerio de Salud de la Nación, 2009]. Así mismo, fue la primera epidemia de Dengue en el país en la que se registraron muertes debido al Dengue, resaltando que el mecanismo involucrado en la gravedad de la enfermedad seguida de óbito, fue la primo-infección [Seijo, 2009]. Durante el año 2010, el Ministerio de Salud de la Nación, notificó un total de 1706 casos de Dengue, 1185 correspondientes a la primer mitad del año [Ministerio de Salud de la Nación, 2010a] y 521 en el Informe de Vigilancia de Dengue con actualización al 21 de Enero de 2011 [Ministerio de Salud de la Nación, 2010b] .

1.2. Epidemiología Panorámica

En los años sesenta se definió el *Remote Sensing* o sensado remoto (SR) como la técnica que permitía el estudio de un objeto sin un contacto directo con el mismo. Actualmente SR se basa en la medición de radiación electromagnética emitida, radiada o reflejada específicamente por cada objeto en la superficie de la Tierra [Scavuzzo et al., 2000]. De esta forma, el sensado remoto hace referencia a la energía captada por sensores a bordo de satélites espaciales, donde se incluyen mediciones provenientes de distintos sensores, fotografías aéreas, láser, receptores de radio, sistemas de radar y sonar, sismógrafos, etc [Porcasi, 2009]. La aplicación de la información espacial en salud, particularmente la Epidemiología Panorámica, es una inter disciplina relativamente nueva que involucra la caracterización de áreas eco-geográficas donde las enfermedades se desarrollan [Scavuzzo et al., 2006]. La misma puede ser entendida como parte de una segunda generación de aplicaciones que usan datos de sensado remoto, donde el vector o reservorio de la enfermedad puede no ser detectado directamente en imágenes satelitales. De esta manera, la Epidemiología Panorámica tiene en cuenta las relaciones e interacciones entre los distintos elementos del ecosistema y la dinámica biológica tanto del hospedador como de la población del vector facilitando el estudio de estas relaciones entre los elementos del paisaje tales como temperatura y vegetación.
Los Sistemas de Alerta Temprana (SAT) tiene fundamento en aquellas enfermedades con variación temporal, que producen picos epidémicos que puede ser disparados por la variabilidad de las condiciones climáticas [WHO, 2009], como es el caso del dengue. Es así que el principal objetivo de la Epidemiología Panorámica, es el desarrollo de mapas de riesgo de enfermedades específicas para lograr la formulación de sistemas de alerta temprana (o usando su denominación en ingles: EWS *Early Warning Systems*) en salud; mejorando de esta manera las acciones en programas de control y prevención de enfermedades [Porcasi, 2009]. De esta manera los SATs tienen como objetivo principal el soporte tanto a la toma de decisión como a la prevención de enfermedades altamente relacionadas con el ambiente espacial. Con respecto a la problemática en el presente trabajo, la necesidad de un SATs en el caso específico de la Fiebre del Dengue es acentuada aun mas debido a la inexistencia por el

momento, de vacuna o tratamiento específico para humanos; siendo la única solución plausible para la enfermedad, la prevención y el control estratégico del vector.

1.3. Objetivo

El Instituto de altos estudios espaciales Mario Gulich perteneciente a CONAE - Universidad Nacional de Córdoba, tiene como principal objetivo el generar Sistemas de Alerta Temprana en emergencias ambientales usando información Espacial proveniente de distintos sensores remotos. De ello y bajo el complejo marco del Dengue en Argentina se desprende la necesidad de contar con la correcta infraestructura informática para dar soporte a los Sistemas de Alerta Temprana. Así, el presente trabajo tuvo como objetivo el el desarrollo integral de una infraestructura informática para alertas tempranas en materia de prevención y control estratégico del Dengue: El "Sistema de Alerta Temprana y Estratificación de Riesgo de Dengue Nacional y Urbano": SAT/ERDNU. A partir de ello se desarrolló una arquitectura robusta y re-utilizable para SATs de cualquier tipo gracias al uso de un correcto diseño (Patrones de Diseños) y frameworks [1] Open Source. Mientras el incorporar Open Source Software (OSS) en el desarrollo es un punto clave en este trabajo y requiere un complejo análisis de los mismos, los Patrones de Diseño proveen una manera eficiente de crear software orientado a objetos [Booch et al., 2008] mas flexible, elegante y re-utilizable. Por lo tanto su correcta aplicación en el diseño de los sistemas influirá significativamente en su reusabilidad, flexibilidad, expansibilidad y mantenibilidad. Así mismo, se debe tener en cuenta que el crecimiento exponencial de las aplicaciones web a dado lugar a un complejo conjunto de las mismas, donde el mantenimiento, la flexibilidad y la expansibilidad se ha vuelto extremadamente difícil [Shuang and Chen, 2009], acentuando mas la necesidad de una correcta elección en el diseño.

[1] En programación, un framework de software es una abstracción en la cual el software provee funcionalidad genérica que puede ser modificada por el desarrollador para proveer una aplicación de software especifico. En general, están compuestos por un conjunto de librerías que proveen una interfase de programación (API - Application Programming Interface).

Capítulo 2

Requerimientos

Como parte de las buenas prácticas en la formulación de proyectos, y cumpliendo con los estándares de la ESA (European Space Agency) [European Space Agency, 2011], el proyecto SAT/ERD-NU comenzó con la formulación de un conjunto de requerimientos de alto nivel (requerimientos de usuario) que luego fueron especificados a nivel de detalle del sistema (requerimientos del sistema). De esta manera se generaron diferentes tipos de requerimientos tanto funcionales como no funcionales que se describen a continuación. Para un mayor detalle de los requerimientos puede consultarse el documento técnico Requierements Baseline Document (RBD) de CONAE revisado y aprobado por el usuario: El Ministerio de Salud de la Nación (Coordinación para el Control de Vectores (DETVs)). El sistema tiene dos escalas de trabajo (un sub-sistema de nivel nacional y otros sub-sistemas a escala local) que funcionarán en una plataforma SIG, para la integración y visualización de los datos. Como objetivos principales del sistema podemos mencionar:

- Estratificar el riesgo de Dengue a escala nacional mediante un modelo multifactorial (medioambiental, demográfico, eco-epidemiológico) capaz de asignar el riego de circulación viral de Dengue a cada localidad del país.
- Generar cartografía de riesgo de Dengue a escala Urbano, a partir de la caracterización de hábitat preferenciales para el desarrollo del mosquito Aedes aegypti y la dinámica de transmisión del virus.
- Visualizar localidades de riesgo a escala nacional.
- Generar cartografía de áreas de riesgo a escala Urbana a partir de datos espaciales, que complementados con datos de campo, optimicen las acciones de control.
- Integrar la información histórica de la circulación viral de dengue y de otros datos epidemiológicos (flujo poblacional, capacidad de respuesta ante eventos epidémicos, entre otros)
- Actualizar frecuentemente los datos ambientales provenientes de sensores remotos.
- Integrar los datos en una plataforma cartográfica amigable, fácil de utilizar y robusta.

2.1. Requerimientos del Usuario

2.1.1. Estratificación del Riesgo del Dengue a escala Nacional: ERDN

Para la estratificación del Riesgo de Dengue a escala Nacional cada localidad esta representada en una capa vectorial de puntos, por un par único de coordenadas (latitud,longitud) al que se asociará información alfa-numérica (en formato .dbf) correspondiente a bloques de factores condicionantes del riesgo:
Caracterización macro-ambiental: será determinada principalmente por datos de temperatura de superficie obtenidos del sensor MODIS, por datos de Índice de Vegetación del mismo sensor (o humedad ambiente) y de altura de la localidad. Estos datos, inicialmente en formato raster, se resumirán a un valor por localidad (promedio de píxeles).
Los siguientes datos se derivarán de una planilla de riesgo que deberá ser ingresada por el usuario conforme a un proceso de validaciones automáticas.
Control vectorial: este bloque caracteriza a localidad en función a la periodicidad en que realizan acciones orientadas al control de la densidad de poblaciones del mosquito A. aegypti Caracterización de la circulación viral: aquí se concentra la información relacionada a circulación autóctona del virus del Dengue en cada localidad.
Riesgo entomológico: este bloque considera la presencia del vector Aedes aegypti en la localidad y otras medidas asociadas a la densidad del mismo (índices entomológicos).
El riesgo final de circulación de Dengue a escala Nacional para cada localidad es definido a través de la aplicación de un modelo matemático en función a los valores obtenidos en cada grupo. El sistema en su conjunto permitirá flexibilidad tanto para la inclusión de nuevos datos (según indicaciones), como en la ponderación o peso de cada variable en la asignación final de riesgo.

2.1.2. Estratificación del Riesgo del Dengue a escala Urbano: ERDU

Para el caso de la Estratificación del Riego a escala Urbano y a partir de imágenes satelitales, datos históricos de circulación viral, índices aédicos (LirAa, ovitrampas, descacharrado, etc.) y cartografía de base en formato vectorial, se estiman anomalías y variaciones espaciales de variables de relevancia para la estratificación de riesgo de Dengue y Dengue Agudo dentro de cada localidad. Las localidades piloto, han sido seleccionadas por solicitud expresa de los referentes epidemiólogos participantes en el diseño de esta herramienta, en base a la disponibilidad de datos de campo y epidemiológicos, a la existencia de eventos epidémicos históricos y recientes, a su localización con respecto a países limítrofes y áreas de circulación viral frecuente.

El sistema deberá entregar tanto para la visualización como para la manipulación una seria de productos que pueden ser resumidos en la tabla 2.1.2.

	ERDN	**ERDNAmb**	**ERDNfact**	**ERDU**	**ERDUamb**	**ERDUFact**
	Mapa riesgo de nivel nacional	Subproducto de riesgo nacional ambiental	Estado de los factores no ambientales	Escala Urbana		
Producto	Vector	Raster	Vector	Raster	Raster	Vector
Area	Argentina	Argentina	Argentina	Localidad	Localidad	Localidad
Resolución temporal	semestral	Bi-mensual	semestral	mensual	anual	mensual
Resolución espacial/ escala de mapa	regional	Km	regional	10 - 30 m/cuadras	10 - 30 m/cuadras	10 - 30 m/cuadras
Datos de entrada	RS, planilla estratificación*	RS, planilla estratificación*	RS, planilla estratificación*	Entomológicos – ambientales –serológicos - Servicios etc.	TBD	TBD
Datos de RS	MODIS, SRTM	MODIS, SRTM	-	– COSMO – Aster -	TBD	TBD

Tabla 2.1.2 : Resumen de los requisitos necesarios de los productos principales del sistema. donde:

- Mediante la planilla de estratificación el usuario ingresa su análisis, datos que serán usados posteriormente en el cálculo de los algoritmos.
- ERDN: Estratificación Riesgo de Dengue Nacional.
- ERDU: Estratificación Riesgo de Dengue Urbano.
- ERDNAmb: Estratificación Riesgo de Dengue Nacional Ambiental.
- ERDNfact: Estratificación Riesgo de Dengue Nacional Factorial.
- ERDUAmb: Estratificación Riesgo de Dengue Urbano Ambiental.
- ERDUfact: Estratificación Riesgo de Dengue Urbano Factorial.

La documentación técnica de los requerimientos del sistema fue elaborada por personal del Instituto Gulich y se encuentra disponible para su consulta en [Porcasi, 2011]. No obstante a continuación se muestra un resumen codificado de los requerimientos del usuario (tabla 2.1.2). Para un completo entendimiento de la codificación usada ver el Apéndice A, la misma se corresponde a la nomenclatura usada para proyectos desarrollados en CONAE.

Codigo	**Descripción**	**Comentarios**
CAE-HAP-FUN-L0-0001	Deberá ser un Sistema Integrado Multiescala para estratificación de riesgo de Dengue	Utiliza información en diferentes formatos y en diferentes escalas de análisis

CAE-HAP-FUN-L0-0002	El Sistema deberá ser automatizado y permitir carga de datos de Usuarios específicos	Los usuarios son responsables de la carga de estos datos
CAE-HAP-QUA-L0-0003	Deberá prever un método de control de fallos de carga datos, formatos y de sistemas de coordenadas	Los usuarios necesitan capacitación para esto
CAE-HAP-RES-L0-0004	Deberá ser capaz de incorporar datos ambientales socio-económicos y epidemiológicos a nivel nacional y urbano	Los datos vienen en diferentes formatos: rasters, vectores o alfanuméricos.
CAE-HAP-FUN-L0-0005	Deberá generar y permitir la visualización de mapas/subproductos a nivel nacional y urbano vía WEB	Con funcionalidad de SIG
CAE-HAP-DES-L0-0006	Deberá ser capaz de hacercálculos de riesgo integrando la información ambiental y epidemiológica	
CAE-HAP-DES-L0-0007	Deberá permitir flexibilidad para la modificación de algoritmos. Además permitir ejecución de algoritmos particulares durante la activación de estado de emergencia.	
CAE-HAP-QUA-L0-0008	Deberá generar productos de calidad (con la actualizaciónrequerida) durante 5 años	

Tabla 2.1.2: Versión técnica de los requerimientos planteados originalmente por los usuarios. Puede ser útil para verificar la compatibilidad de los requerimientos. Para una descripcion completa de los requerimientos del usuario ver [Porcasi, 2011].

2.2. Requerimientos del Sistema

Finalmente, a partir de los requerimientos por naturaleza vagos del usuario se definieron requerimientos mas concretos técnicos y detallados: los requerimientos del sistema. A continuación se presenta la matriz de requerimientos del sistema (tabla 2.2.1) con la misma codificación usada para los requerimientos del usuario explicada en el Apéndice A. Nuevamente, la misma pretende ser un resumen de los requerimientos del sistema los que pueden ser profundizados en [Porcasi, 2011].

2.2.1. Matriz de Requerimientos del sistema

Código	Descripción	Comentarios	Precursor
CAE-HAP-FUN-L0-0001	Sistema Integrado Multiescala para estratificación de riesgo de Dengue		
CAE-HAP-DES-L1-0001	Deberá contener un sistema de riesgo a nivel nacional	Una localidad representada por un punto	CAE-HAP-FUN-L0-0001
CAE-HAP-DES-L1-0002	Deberá contener sistema de riesgo a nivel urbano	Para las localidades seleccionadas y con detalle de información a nivel de manzana	CAE-HAP-FUN-L0-0001

CAE -HAP-FUN-L0-0002	El sistema deberá ser automatizado y permitir carga de datos de usuarios específicos		
CAE-HAP-FUN-L1-0003	El sistema deberá ser capaz de incorporar datos numéricos de variables de riesgo desde diferentes localidades del país	Localidades/municipios con acceso a Internet	CAE -HAP-FUN-L0-0002
CAE-HAP-DES-L1-0004	Deberá tener a usuarios definidos y habilitarlos para el ingreso de datos y/o consultas	Categorías de usuarios: Visualización, carga de datos y operador SIG	CAE -HAP-FUN-L0-0002
CAE-HAP-OPE-L1-0005	Se deberá capacitar a usuarios del MSAL para el ingreso de datos		CAE -HAP-FUN-L0-0002
CAE-HAP-FUN-L1-0006	El sistema deberá incorporar datos en formato vectorial y raster		CAE -HAP-FUN-L0-0002
CAE-HAP-PER-L1-0007	El sistema deberá funcionar de manera automatizada (compatible para instalación en el CUSS)	Bajo condiciones estándar generar los productos sin intervención sistemática del operador	CAE-HAP-FUN-L0-0002
CAE -HAP-QUA-L0-0003	Deberá prever un método de control de fallos de carga datos, formatos y de sistemas de coordenadas		
CAE-HAP-QUA-L1-0008	Deberá detectar y notificar fallas en las cargas de datos alfa-numéricos		CAE -HAP-QUA-L0-0003
CAE-HAP-QUA-L1-0009	Deberá detectar falta de actualización de datos	Esta condición se visualizara en el sub-producto de riesgo	CAE -HAP-QUA-L0-0003
CAE-HAP-QUA-L1-0010	Deberá verificar el formato de datos de entrada (raster y vectoriales) y el uso de un sistema de coordenadas acordado	Lo hará manualmente un operador con acceso al sistema	CAE -HAP-QUA-L0-0003
CAE - HAP -RES-L0-0004	Deberá ser capaz de incorporar datos ambientales, socio-económicos y epidemiológicos a nivel nacional y urbano		
CAE-HAP-FUN-L1-0011	Deberá incorporar datos ambientales a nivel nacional	Principalmente rasters	CAE - HAP -RES-L0-0004
CAE-HAP-FUN-L1-0012	Deberá incorporar datos ambientales a nivel de ciudad.	Principalmente rasters	CAE - HAP -RES-L0-0004
CAE-HAP-FUN-L1 -0013	Deberá incorporar datos socio-culturales y de control vectorial a nivel de ciudad.	Principalmente vectores de polígonos y puntos	CAE - HAP -RES-L0-0004

CAE-HAP-DES-L1-0014	Todos los datos deberán ser alojados en una única base de datos centralizada.	Los datos serán consultados por el sistema o usuarios de manera remota. A pedido de MSAL los datos se centralizaran en el CUSS de CONAE.	CAE - HAP -RES-L0-0004
CAE-HAP-RES-L1-0015	Los datos deberán incorporarse a un SIG de base, con coordenadas geográficas (lat-long)	La base cartográfica definitiva del SIG usado deberá acordarse por ambas partes	CAE - HAP -RES-L0-0004
CAE – HAP-FUN-L0-0005	Deberá generar y permitir la visualización de mapas/ subproductos a nivel nacional y urbano via WEB		
CAE-HAP-DES-L1-0016	Los datos deberán visualizarse con el mismo SIG de base accesible via WEB	Con coordenadas geográficas (lat-long)	CAE - HAP-FUN-L0-0005
CAE-HAP-RES-L1-0017	Los datos adquiridos deberán ser guardados y quedar a disposición en un Archivo de datos	Para visualización y cálculo de algoritmos	CAE - HAP-FUN-L0-0005
CAE-HAP-FUN-L1-0018	El sistema deberá mostrar mapas de la valoración de riesgo a nivel nacional (ERDN)	Visualización de cada localidad como un punto codificada en colores según el riesgo.	CAE - HAP-FUN-L0-0005
CAE-HAP-FUN-L1-0019	El sistema también deberá permitir la visualización de sub-productos a nivel nacional (ERDNAmb y ERDNFact)	Para evaluar factores individuales que influyen en el riesgo de dengue nacional	CAE - HAP-FUN-L0-0005
CAE-HAP-FUN-L1-0020	El sistema será capaz de visualizar simultáneamente datos de entrada (capas) como casos e índices aédicos	Para evaluar factores que influyen en el riesgo a nivel urbano	CAE - HAP-FUN-L0-0005
CAE-HAP-FUN-L1-0021	El sistema deberá permitir descargar algunos de los productos pre-procesados en formato “JPEG”	Ya sean mapas o reportes	CAE - HAP-FUN-L0-0005
CAE-HAP-FUN-L1-0022	El sistema deberá permitir la visualización de resultados de algoritmos como clasificaciones de suelo y “Hot-spots”	Son sub-productos del riesgo a nivel urbano	CAE - HAP-FUN-L0-0005

CAE – HAP-DES-L0-0006	Deberá ser capaz de hacer cálculos de riesgo integrando la información ambiental y epidemiológica		
CAE-HAP-FUN-L1-0023	Deberá computar algoritmos para detectar “hot-spots” a escala de ciudad	“Hot-spot”: son áreas de mayor concentración de índices aédicos y/o casos de dengue	CAE - HAP-FUN-L0-0006
CAE-HAP-FUN-L1-0024	Deberá generar mapas de cobertura/uso del suelo	Subproducto a nivel urbano con actualización al menos anual realizado por un operador	CAE - HAP-FUN-L0-0006
CAE-HAP-FUN-L1-0025	El sistema deberá correr los algoritmos (que figurarán en el ATBD) para generar mapas de riesgo a nivel urbano	Combinando los factores ambientales y epidemiológicos	CAE - HAP-FUN-L0-0006
CAE – HAP-DES-L0-0007	Deberá permitir flexibilidad para la modificación de algoritmos. Además permitir ejecución de algoritmos particulares durante la activación de estado de emergencia.		
CAE-HAP-DES-L1-0026	Deberá poseer flexibilidad para realizar cambios menores en los algoritmos	En base a avances científicos se prevé una continua actualización de los algoritmos a aplicar para la estimación de riesgo	CAE-HAP-PER-L0-0007
CAE-HAP-DES-L1-0027	Deberá permitir el cambio a “estado de emergencia”	Aumentaría la frecuencia de actualización de los datos de entrada y las visualizaciones. Podrá incorporar herramientas avanzadas de análisis espacio-temporal. Se activará a pedido del MSAL	CAE-HAP-PER-L0-0007

CAE-HAP-PER-L1-0028	Deberá generar productos indicadores de riesgo aún con datos faltantes	Robustez. Deberá quedar registro del tipo de dato faltante en el correspondiente metadato.	CAE-HAP-PER-L0-0007
CAE -HAP-QUA-L0-0008	Deberá generar productos de calidad (con la actualización requerida) durante 5 años		
CAE-HAP-QUA-L1-0029	El sistema deberá tener productos con un error de georeferencia acotado.	A nivel urbano < 10m A nivel Nacional < 1Km	CAE-HAP-QUA-L0-0008
CAE-HAP-QUA-L1-0030	El sistema deberá actualizar los productos: Mapa Nivel nacional: semestral Productos nivel urbano: mensual Estado de Emergencia: TBD		
CAE-HAP-FUN-L1-0031	El sistema deberá ser capaz de incluir datos de nuevos sensores ópticos con características similares para garantizar una vida útil de al menos 5 años		CAE-HAP-QUA-L0-0008
CAE-HAP-QUA-L1-0032	Deberán incluirse herramientas de validación de los mapas para evaluar los productos generados	Para verificar los mapas con datos de terreno	CAE-HAP-QUA-L0-0008
CAE-HAP-QUA-L1-0033	Para cada producto, el sistema deberá generar metadatos que incluirá, entre otros atributos, la historia de producción del mismo.		CAE-HAP-QUA-L0-0008

Tabla 2.2.1: Versión técnica de los requerimientos del Sistema Para una descripción mas detallada ver [Porcasi, 2011].

Capítulo 3

Arquitectura elegida

3.1. La solución: Infraestructura Informática

Como se mencionó anteriormente la Comisión Nacional de Actividades Espaciales desarrolla software bajo los estándares de la European Space Agency (ESA) los que definen prácticas para el desarrollo de software espacial y deben ser aplicadas en este ámbito [European Space Agency, 2011]. La especificación de requerimientos, el diseño detallado del sistema, y la definición de sus interfaces son etapas que deben ser llevadas a cabo en el desarrollo de un correcto sistema informático que se desarrolle en el ámbito espacial. De este modo, cualquier software desarrollado en CONAE, y en concreto el SAT/ERDNU cumple con dichos estándares. El presente capítulo describe la arquitectura adoptada para el sistema informático desarrollado, la cual se concebió pensando principalmente en la necesidad de contar con una herramienta re-utilizable en cualquier tipo de SAT. Al concebir la arquitectura teniendo en cuenta el requrimiento CAE-HAP-PER-L1-0007 (el sistema deberá operar dentro del CUSS del CETT-CONAE ver [De La Torre, 2010; Oglietti, 2002] y cumpliendo con sus especificaciones) es necesario tener en cuenta que la operatividad de un sistema en cualquier ambiente se logrará facilmente si el mismo es altamente configurable y mantenible en el tiempo, lo cual se cumplirá solo si presenta una arquitectura modular, versatil y flexible. De esta manera un sistema será dividido en partes: subsistemas y unidades. La unidad es la minima porción del sistema que puede ser desarrollado independientemente. Mientras que un subsistema podrá agrupar a un conjunto de unidades con un objetivo en común.

3.2. Arquitectura

El SATERDNU esta compuesto por los subsistemas Data Translation Subsystem (DT), Algorithm Executor Subsystem (AE), Spatial Data Subsystem (SD), Visualization Subsystem (VZ) y el Alarm Trigger Subsystem (AT) (ver figura 3.1).

Subsistema DT (Data Translation), es el responsable de cargar en el servidor los datos enviados por el usuario; validar los datos enviados por el usuario y convertir los datos a XML. Una vez convertidos a XML, los datos son enviados al subsistema SD, para luegos ser consultados por el subsistema AE. Este subsistema contiene las unidades Uploader WebInterface, xls2xml, y las unidades translator (una por tipo de planilla a guardar).

Subsistema SD (Spatial Data). El subsistema SD, es el encargado de gestionar el repositorio de datos. Será el encargado de almacenar los datos de salida del subsistema DT y los datos de salida del subsistema AE de manera óptima para ser luegos usados por el subsistema VZ.

Subsistema AE (Algorithm Executor). El subsistema AE, será el responsable de ejecutar los algoritmos para producir los mapas de riesgo (un algoritmo por mapa de riesgo), para el período de tiempo especificado en las reglas, y con los datos injestados por el subsistema DT.

Subsistema VZ (Visualization) será el encargado de mostrar los datos de manera amigable al usuario, a travez de la web. Este Subsistema implementará la descarga de los mapas de riesgo y de las capas de información disponibles a la hora de la visualización. Además mediante este Subsistema se permitirá la impresion de mapas.

Subsistema AT (Alarm Trigger) será el encargado de activar la situación de emergencia; en esta situación el AE cambiará la frecuencia de ejecución y los parámetros de los algoritmos. El manejo del pedido de emergencia es gestionado por las unidades Emergency Interface y el Emergency Module.

A continuación se presenta una descripción detallada de los diferentes subsistemas y sus unidades.

3.2.1. Subsistema Data Translation

Las unidades que conforman el subsistema DT de HAP son:

Unidad Uploader WebInterface. Esta unidad es la encargada de brindar la interface gráfica al usuario para que el mismo suba los datos de planillas al servidor. Además esta unidad es la encargada de mostrar un mensaje de éxito o fracaso segun como haya sido la carga de los datos. En el caso de fracaso, se describirá donde se produjo el error.

Unidad xls2xml. Esta unidad es la encargada de convertir el archivo xls subido por el usuario autorizado a archivo xml. Es así que la función principal de dicha unidad es convertir los datos ingresados a un formato estandar y de fácil acceso para su posterior procesamiento. Dicha unidad también guarda el archivo xls en una carpeta input.

Unidades Translators. Existe un translator por cada tipo de planilla ingresada por el usuario. La función principal de estas unidades es la verificación de los datos cargados por el usuario y el almacenamiento en el subsistema SD de los mismos. En el caso en el cual los datos sean erroneos, esta unidad se encarga de disparar una respuesta al usuario.

En la figura 3.2 se muestra las unidades del subsistema DT y la manera en la cual ellos estan interconectados. Además se muestra las conexiones de estas unidades con los demás subsistemas de SAT/ERDNU.

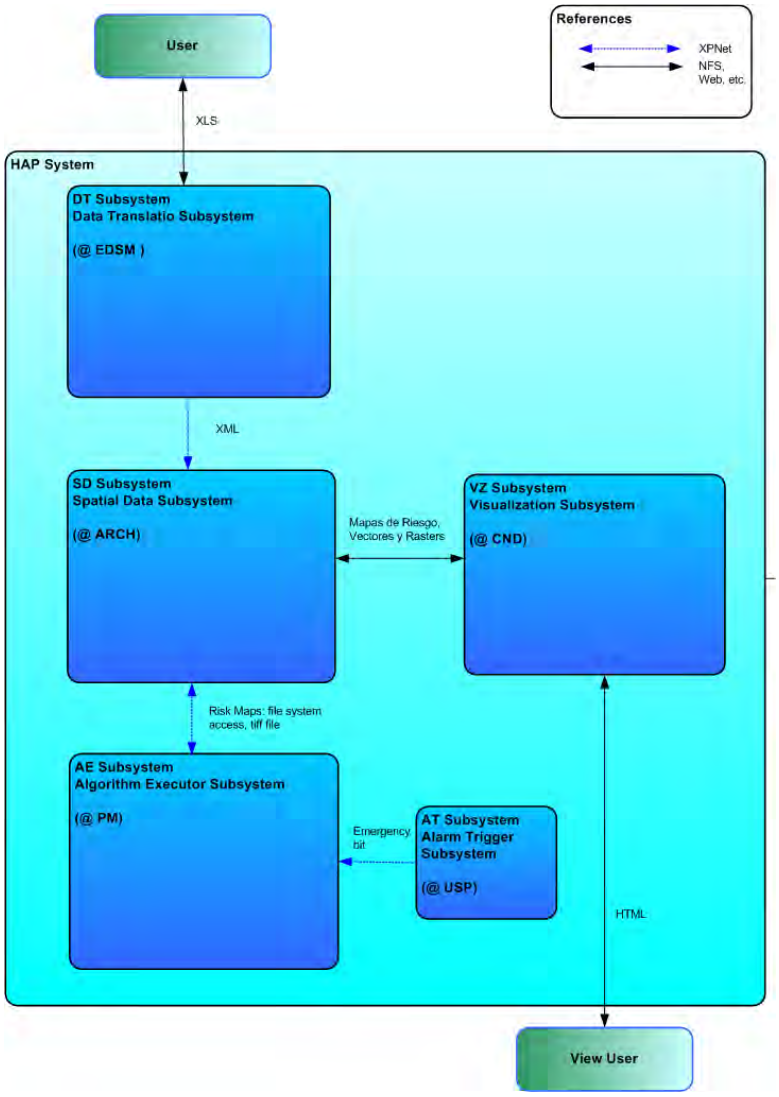

Figura 3.1: Representación del sistema HAP.

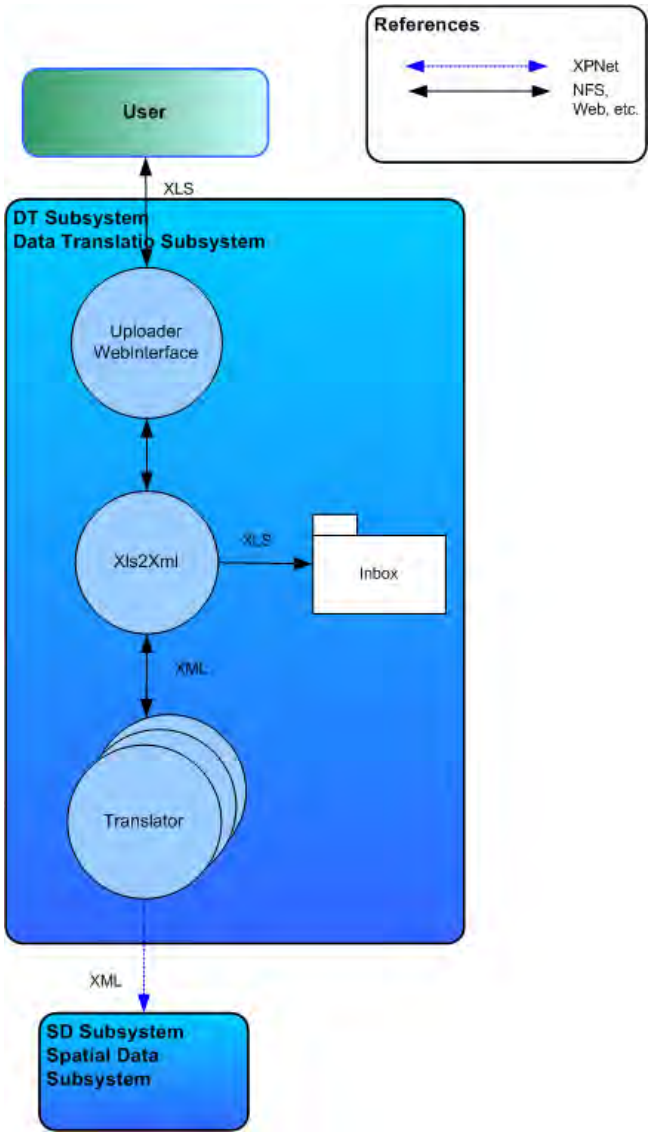

Figura 3.2: El subsistema DT

3.2.2. Subsistema Spatial Data

Este subsistema es el encargado, en primera instancia, de almacenar los archivos xml correspondientes a los datos ingresados por el usuario. Los cuales son necesarios para la ejecución de los algoritmos. En segunda instancia, almacena los resultados de la ejecución de los algoritmos en formato tiff para su posterior vizualización. Es así que el subsistema SD esta compuesta por una unidad que es la encargada de copiar los diferentes archivos en las carpetas de almacenamiento. Es requisito de la actual arquitectura del CUSS de CONAE, no disponer de bases de datos propietarias y tender de esta manera a manejar la mayor parte de los datos a traves de archivos presindiendo del uso de bases de datos. Por este motivo es que los datos tanto ingresados por el usuario como los producidos por la aplicación de los modelos se almacenan en archivos xml y tiff respectivamente. Dicha unidad maneja conflictos de accesos y atomicidad de operaciones. La figura 3.3 muestra dicho subsistema

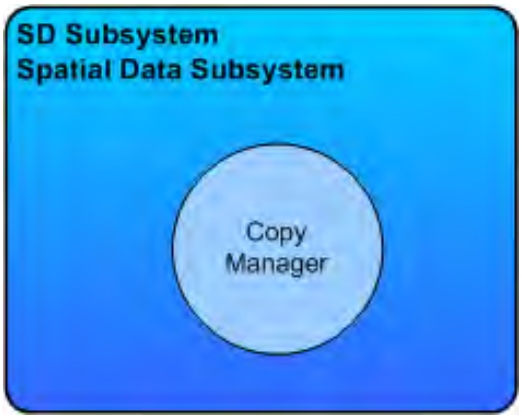

Figura 3.3: El subsistema SD

3.2.3. Subsistema Algorithm Executor

Las unidades que conforman dicho subsistema son:

Algorithm Execution Manager. Esta unidad gestiona la ejecución de los algoritmos específicos para la producción de los mapas de riesgo. Las reglas son archivos de configuraciones donde se especifican los distintos parámetros necesarios para la ejecución correcta de los distintos algoritmos. Esta unidad guarda en el subsistema SD los mapas de riesgo producidos por cada algoritmo.

Process_1,Process_2,...,Process_n. Existe una unidad Process por algoritmo necesario a ejecutar. El mismo es robusto, es decir, se ejecuta en situaciones carentes de ciertos datos. De esta manera, simplemente agregando una unidad Process, se podrá agregar nuevos productos al sistema.

xmlParser. Esta unidad será la encargada de la interface entre el archivo xml y los datos necesarios para la ejecución de cada algoritmo.

La figura 3.4 muestra las subunidades de dicho subsistema

3.2.4. Subsistema Visualization

Las unidades que conforman dicho subsistema son:

GeoServer. Esta unidad sirve diferentes datos geográficos, entre ellos los mapas de riesgo generados por los algoritmos.

GIS Web Viewer. Esta unidad muestra los datos servidos por la unidad GeoServer, de manera amigable e interactiva al usuario autorizado. Esta funcionalidad es alcanzada mediante un sistema de información geográfico. Además esta interface permite la descarga de las capas visualizadas (el ususario necesita tener permisos de descarga) y permite la impresión de mapas (aquí tambien el usuario deberá tener permisos para tal fin).

La figura 3.5 muestra la composición del mismo.

3.2.5. Subsistema Alarm Trigger

Las unidades que componen el sub sistema AT son:

Unidad Emergency Interface. Esta unidad será la encargada de brindar la interface al usuario autorizado para que el mismo active la situación de emergencia.

Unidad Emergency Manager. Esta unidad será la encargada de modificar las reglas de los algoritmos en el subsistema AE, mediantes las cuales se procesan los datos.

La figura 3.6 muestra la composición del mismo.

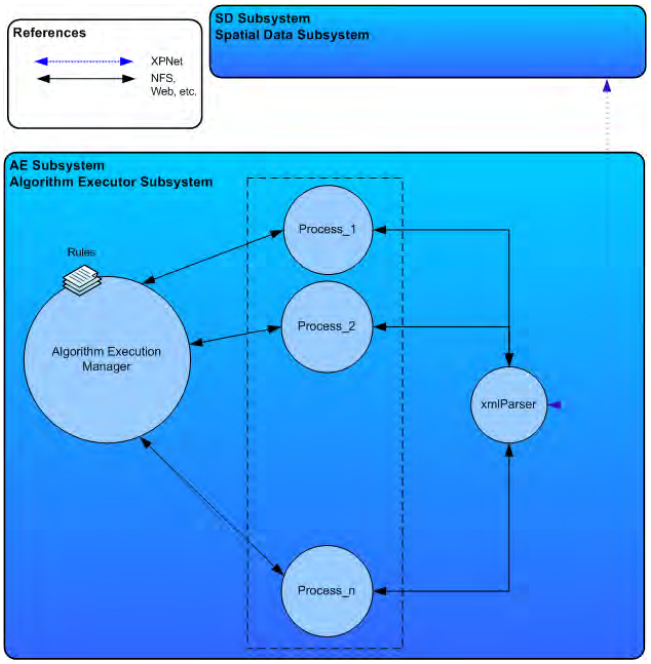

Figura 3.4: El subsistema AE

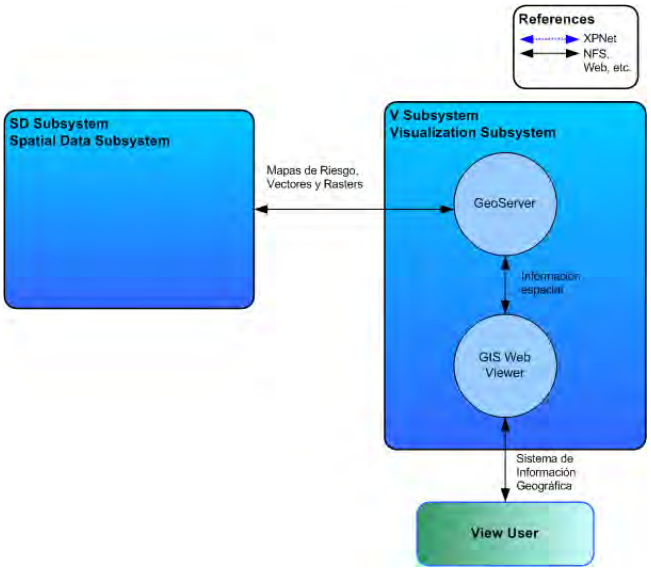

Figura 3.5: El subsistema VZ

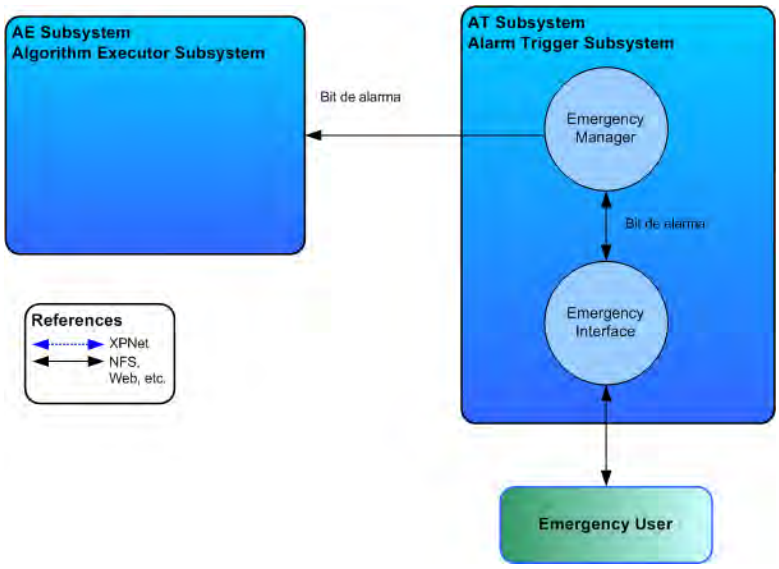

Figura 3.6: El subsistema AT

Capítulo 4

Evaluación de tecnologías de implementación

Los OSS pueden ser definidos como "Software donde el código fuente esta disponible para usar, estudiar, reusar, modificar, mejorar y redistribuir por los usuarios del mismo [Kennedy, 2010]. Por el contrario, en el software comercial el código fuente solo puede ser usado, modificado y redistribuido (en algunos casos) por sus propietario [Ullah and Khan, 2011]. Actualmente, instituciones y compañías tienen que enfrentarse al desarrollo de proyectos ampliamente distribuidos con requerimientos que cambian permanentemente. De igual manera que en [Torkar et al., 2011] en el presente trabajo se planteó la adopción de practicas OSS para mejorar el desarrollo del mismo, lo cual involucró un exhaustivo análisis de las tecnologías posibles, hecho que se describe a lo largo del presente capítulo. De esta manera pensar en la etapa de diseño usando OSS y Patrones de Diseño se asegura en cierta manera un alto porcentaje de éxito. Además de las ventajas de bajo coste, usando OSS la colaboración entre desarrolladores y usuarios promueven el compartimiento de ideas que producen una solución confiable. Mientras el uso de patrones beneficiará su mantenibilidad en el tiempo y re-usabilidad. La elección de OSS tiene un fundamento aun mayor si pensamos en que el 85 % de las compañías a adoptado a políticas de uso OSS mientras el restante 15 % esta moviendo a adoptarlas en los próximos años [Morasca et al., 2011].

4.1. Desarrollo Orientado a Objetos

Uno de los mayores beneficios del desarrollo orientado a objetos es la reusabilidad, lo cual no es trivial de alcanzar siendo necesario un correcto diseño para lograrlo. El diseño de software orientados a objetos es difícil: es necesario encontrar objetos adecuados, crear sus clases y definir jerarquías de herencia e interfaces, teniendo en cuenta sus interrelaciones. El diseño debería ser específico del problema pero lo suficiente general para poder reutilizar la solución en problemas y requerimientos futuros, evitando re-diseño o por lo menos reduciéndolo [Gamma et al., 1994]. El desarrollo de software orientados a objetos creció significativamente durante los últimos años producto del crecimiento en las técnicas de modelado, los Patrones de Diseño y finalmente al surgimiento de nuevos frameworks de trabajo [Shuang and Chen, 2009]. A partir de estas premisas y teniendo en cuenta la necesidad de incrementar la productividad disminuyendo complejidad se decidió integrar un conjunto de tecnologías Open Source disponibles a fin de facilitar la creación de la infraestructura necesaria de soporte al SAT/ERDNU. En general cada una de las tecnologías elegidas presentan un Patrón de

Diseño acorde a los resultados que se quiere obtener y dicho patrón esta bien implementado en la misma. La elección de las herramientas tecnológicas es un tema no menor en la construcción de todo el SAT/ERDNU siendo el diseño que las mismas presentan un parámetro clave para su elección debido al requerimiento fundamental de operatividad y mantenimiento que debe cumplir el sistema informático.

4.2. Spring

Spring es una tecnología dedicada a facilitar la construcción de aplicaciones usando POJOs[1]. Objetivo que requiere de un sofisticado framework que oculte la complejidad al desarrollador [Johnson, 2005]. De esta manera Spring puede ayudar a implementar la mas simple solución posible al problema. En febrero del 2003 Spring se convierte en un proyecto Open Source nacido a partir del código publicado en el libro Expert One-on-One J2EE Design and Development, por Rod Johnson a fines del 2002. A partir de entonces, Spring esta hosteado en SourceForge con 20 desarrolladores liderando y ayudando a los diversos contribuidores del proyecto. Existen muchas aplicaciones en producción usando Spring. Usuarios del framework son inversores y organizaciones bancarias, sitios de renombre dotcoms, consultorias, instituciones académicas, departamentos de gobiernos, diversas compañías aéreas, y organizaciones de investigaciones científicas (incluyendo CERN - European Organization for Nuclear Research) [Johnson, 2005]. Muchos de los usuarios usan todos los componentes de Spring, muchos otros, utilizan sus componentes aislados como por ejemplo el componente MVC web framework, que proporciona el framework para aplicaciones web, o el framework JDBC que proporciona una única abstracción de acceso a las distintas plataformas de bases de datos. Es decir que Spring hace que las diversas tecnologías existentes sean fácil de usar solucionando de esta manera muchos de los problemas existentes en J2EE. Además de lo mencionado anteriormente se desprende la capacidad brindada de manejar objetos de negocio y fomentar las buenas practicas de programación, tales como la programación de interfaces, en lugar de clases. En la figura 4.1 se puede observar la arquitectura general de este framework. Donde el paquete Core provee la Inyección de Dependencia (ver sub sección 4.2.1); el objeto BeanFactory provee la implementación del patrón factory [Gamma et al., 1994]. El paquete DAO provee la capa de abstracción a JDBC que remueve la necesidad de codificar las interfaces correspondientes al JDBC las cuales dependerán del propietario de la base de datos, lo que deviene en una tarea tediosa de codificación. El paquete ORM provee integración de tecnologías para el mapeo con APIs populares como JPA, JDO hibernates, de manejo de datos. El paquete AOP de Spring provee AOP (Alliance-compliant aspect-oriented programming). La programación orientada a aspectos o sus siglas en inglés: Aspect-Oriented Programming (AOP) complementa la programación Orientada a Objetos (sus siglas en ingles OOP) proporcionando otra manera de estructurar el programa. Mientras OO desacopla las aplicaciones en jerarquías de Objetos, AOP descompone los programas en aspectos o intereses. Lo que permite la modularización de dichos intereses tales como administración de transacciones, que podrían resultar en múltiples obje-

[1] POJO son las iniciales de *Plain Old Java Object*, que puede interpretarse como *Un objeto Java Plano y a la Antigua*. Un POJO es una instancia de una clase que no extiende ni implementa nada en especial. Son clases con atributos y operaciones simples.

tos interoperando. El paquete Web de Spring proporciona la integración de diversas funcionalidades para la creación de aplicaciones web. El cual como punto principal contiene al paquete Spring MVC discutido en este trabajo mas adelante.

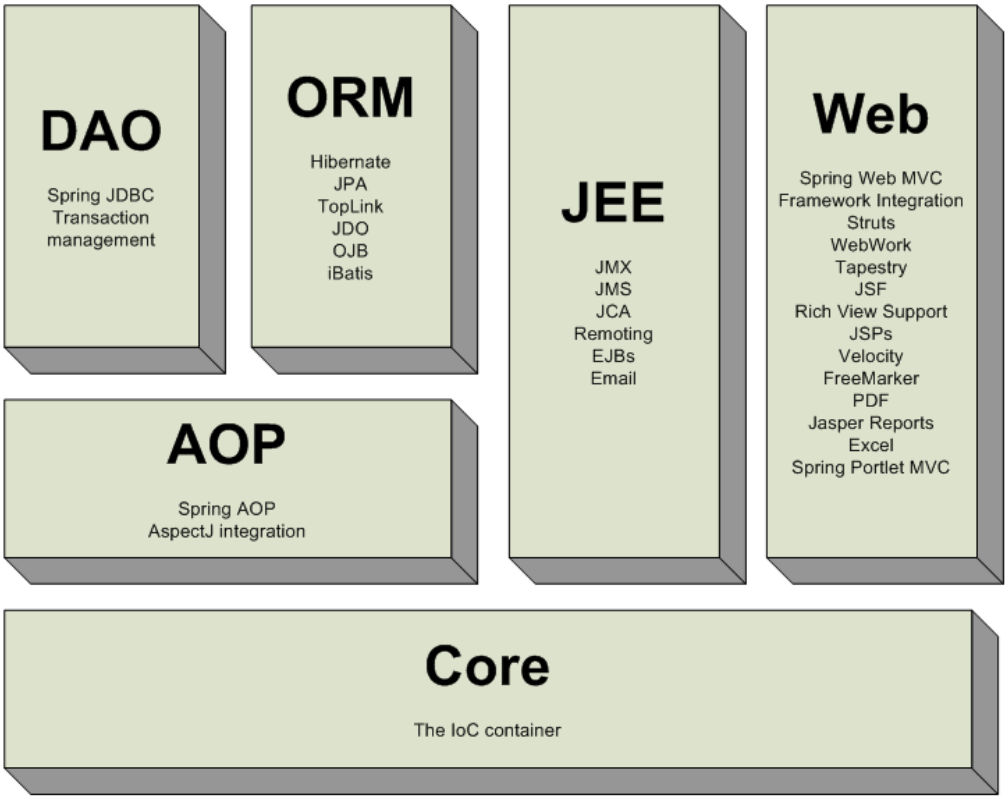

Figura 4.1: Arquitectura general de Spring, extraido de: [Johnson et al., 2011]

4.2.1. Inversión de Control IoC

La base arquitectural de Spring es el contenedor de Inversion de Control basado en el uso de propiedades JavaBean. Aunque Spring implementa un tipo de Inversion de Control denominado por Martin Fowler, Rod Johnson y el equipo de PicoContainer como Inyeccion de Dependencias o sus siglas en inglés Dependency Injection. El concepto detrás de la Inversion de Control (IoC) puede ser resumido, como lo sintetizó Rod Johnson en [Johnson, 2005]: ”Don’t call me, I’ll call you.” El IoC parte de Spring adquiere la responsabilidad de hacer que esto ocurra, lejos del codigo de la aplicación. Mientras en la codificación tradicional las librerias son llamadas por el codigo, con el IoC es el framework quien llama al codigo. Inyección de Dependencias es una forma de IoC que remueve las dependencias explicitas de las APIs, donde metodos son usados para inyectar dependencias tales como objetos de soporte, valores de configuración dentro de instancias de objetos. Si se piensa en el ámbito de las configuraciones, en la arquitectura tradicional de cualquier aplicación usando EJB[1], un componente podría llamar al contenedor para preguntar por el objeto X que necesita para realizar su trabajo. Con Inyección de Dependencias el contenedor se da cuenta que el componente necesita el objeto X para su ejecución, el cual se provee en tiempo de ejecución. El contenedor realizará esta ref-

[1]Enterprise JavaBean es una clase que agrupa funcionalidades para una aplicación, que existe dentro de un ambiente de ejecución o contenedor.

erencia basado en propiedades JavaBean[1], constructores, y datos de configuración tales como XML. Los dos puntos claves de la Inyección de Dependencias en Spring son la Inyección via JavaBeans setters y la Inyección de Dependencia via los argumentos del constructor (ver [Johnson et al., 2011]). De este modo Spring usa el contenedor de IoC como su bloque de construcción básico.

4.2.2. MVC web framework

MVC consiste en tres tipos de objetos. El modelo (Model) es el objeto aplicación, la vista (View) es su presentacion, y el controlador (Controller) que define la manera en que la interface de usuario reacciona a las entradas del mismo. MVC desacopla estos objetos para incrementar flexibilidad y reuso. Dicho patrón aisla la lógica de negocio[2] de las consideraciones de la interfase del usuario, de esta manera es fácil modificar o bien la apariencia visual o bien las reglas de negocio [Gamma et al., 1994]. Spring proporciona un framework altamente configurable para la construcción de aplicaciones web que implementa el patrón de diseño MVC. Como ventajas en el uso de Spring para la aplicación del MVC es posible mencionar que: provee una clara división entre controladores, el modelo JavaBeans, y la vista. Spring MVC es View-agnostic, es decir, no es necesario usar JSP en el View, siendo posible el uso de XLST u otra tecnología de visualización. Los controladores son configurados vía el IoC lo que permite versatilidad. Y finalmente la capa de visualización web, de esta manera, se transforma en una fina capa superior que envuelve los objeto de negocio lo que fomenta las buenas practicas de programación.

4.3. Web Services

Años atrás quedaron los días en que la informática se basaba en programas monousuario corriendo en computadoras que no eran capaces de manejar mas de una tarea a la vez. Desde hace años las aplicaciones comenzaron a manejar con facilidad multi-usuarios y multi-tareas y con el crecimiento de internet la arquitectura mayormente comenzó a ser cliente-servidor, donde las aplicaciones se dividían en una parte que interaccionaba con el usuario y otra parte destinada al procesamiento de información. En este acercamiento se logró que cada una de las partes que constituían la aplicación pudiera residir en computadoras distintas, y con el paso del tiempo, no solo aumentó la capacidad sino también la necesidad de cómputo llegando la era de las aplicaciones distribuidas en las cuales los procedimientos se realizaban en diferentes unidades. Hoy en día se pueden pensar a los web services como un paso adelante en la computación, donde una computadora ya no se considerara como un núcleo de computo sino como un repositorio de servicios de n aplicaciones distribuidas por la red. El World Wide Web Consortium (W3C) en [W3C Working Group, 2004] define a un web service como un sistema de software diseñado para soportar interacciones interoperativas de maquina a maquina sobre la red. Este tiene una interfaz descripta en un formato procesable por la computadora (WSDL). En la mayoría de los casos la interacción se logra usando mensajes SOAP,

[1] JavaBeans son componentes de software resusables (clases) que sigue las convenciones definidas por Sun. Basicamente, la clase tiene que tener un constructor, los atributos de la clase deben ser accesibles via los metodos setters y getters y por ultimo la clase de ser serializable

[2] lógica de negocio se denomina a la lógica del domino en particular

estableciendo HTTP con una serialización de XML en unión con otros estándares Web relacionados. Los servicios Web son muy prácticos y pueden aportar gran independencia entre la aplicación que usa el servicio Web y el propio servicio. De esta forma, los cambios a lo largo del tiempo en uno no deben afectar al otro. Esta flexibilidad será cada vez más importante, dado que la tendencia a construir grandes aplicaciones a partir de componentes distribuidos más pequeños es cada día más utilizada y en particular se corresponde directamente con el sistema desarrollado.

4.3.1. Agentes y Servicios

Un Web service es una abstracción implementada por un agente concreto (ver Figura 4.2). El agente es la unidad de software o hardware que envía y recibe mensajes. Mientras el servicio es el recurso caracterizado por la abstracción del conjunto de funcionalidad provista. Es posible implementar un particular Web service usando un agente, y cambiarlo con el paso de tiempo usando otro agente y otra forma de implementación, quizás cambiando hasta el lenguaje de programación.

4.3.2. Consumidor y Proveedores

El propósito de un Web Service es dar funcionalidad a su creador. El proveedor (provider) de un Web Service es la entidad que provee un agente que implementa un servicio en particular (ver 4.2). El consumidor (consumer) es la entidad que desea hacer uso del Web Service provisto. Este necesitará un agente consumidor para intercambiar los mensajes con el agente proveedor de la entidad proveedora. Para que el intercambio de estos mensajes sea exitoso, el consumidor y el proveedor deben estar de acuerdo con la semántica y el mecanismo de intercambio estableciendo el contrato (WSLD[1])

4.3.3. Descripción del servicio: Contrato

El mecanismo de intercambio de mensajes debe ser documentado en la descripción del Web Service (WSD) (ver figura 4.2) o contrato. El WSD es una especificación de las interfaces del Web service escrito en WSDL. Este define el formato de los mensajes, el tipo de datos, los protocolos de transportes, entre otras cosas que deben ser usadas entre el provider y el consumer. Este además especifica la ubicación donde el provider debe ser invocado.

4.3.4. Spring Web Services

Existen dos modos de desarrollar un Web Service: primero el contrato (contract-first) y ultimo el contrato (contract-last). Con el modo contract-last, se escribe primero el servicio a brindar dejando para ultimo momento el contrato, el cual es creado a partir de la implementación del servicio. Con el modo contract-first, se comienza primero definiendo de manera correcta el WSDL o contrato y usando el mismo para definir el servicio. El principal defecto del contract-last recae en el hecho de que convertir objetos Java a XML no es trivial. El mapeo parece simple: crear un elemento XML a partir de un objeto Java convirtiendo las propiedades y campos en sub elementos o atributos del

[1] WSDL o sus siglas en inglés: Web Services Description Language. Un archivo WSDL es un documento XML que describe un servicio Web. Este especifica la ubicación del servicio y las operaciones que el mismo brinda.

XML. Sin embargo existe diferencias fundamentales entre los lenguajes jerarquizados como XML y el modelo presente en Java (ver [Johnson et al., 2011]) lo que dificulta esta conversión. Además usando el contract-last, un cambio en la implementación del Web service devendrá en un nuevo contrato, perdiendo iteroperatividad entre los consumidores del servicio. Aun teniendo en cuenta que el contrato debe existir en el tiempo tanto como sea posible, el mismo puede cambiar. Cambiar usando el modo contrat-last típicamente resultará en una nueva interfase Java y una nueva implementación de esa interface, además de mantener el viejo servicio por cuestiones de compatibilidad, lo que genera un pesado trabajo de gestión de cambios. Más aun, usar contract-first significa perder el acoplamiento entre el contrato y la implementación, manteniendo interoperatividad de las partes independientemente del mismo. A través del uso de Spring la aplicación del modo contract-first es asegurado, ya que es la única modalidad adoptada por dicho framework.

4.4. Google Web Toolkit (GWT)

El problema de desarrollar aplicaciones web usando como lenguaje de programación de back-end a Java, es que necesitamos implementar la interfase de usuario (UI por sus siglas en ingles User Interface) usando otro lenguaje de programación soportado por el navegador web. Java no es intrínsecamente compatible con las interfaces gráficas manejadas por la web, para ello es necesario implementar la parte a ejecutarse en el cliente o navegador web, usando algún otro tipo de lenguaje como JavaScript, PHP, etc. perdiendo robustez y facilidad de desarrollo. El Google Web Toolkit (GWT)

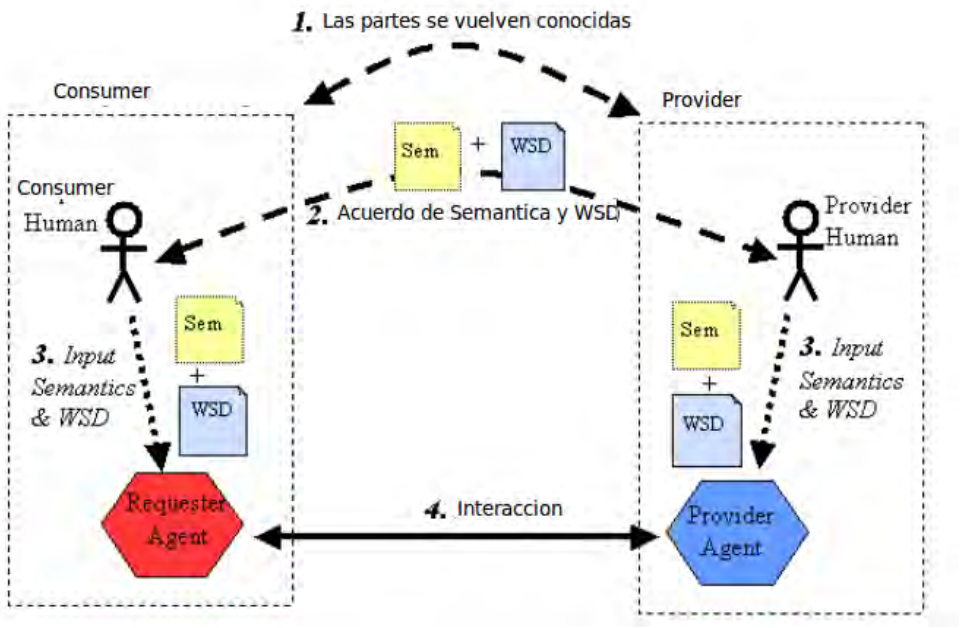

Figura 4.2: Arquitectura general de un Web Service. Extraído de: [W3C Working Group, 2004]

permite crear aplicaciones AJAX[1] en el lenguaje de programación Java que son compiladas posteriormente por GWT en código JavaScript ejecutable optimizado que funciona automáticamente en los principales navegadores. Es decir que, mediante el GWT el desarrollador se abstrae del problema de incompatibilidad entre Java y el navegador Web implementando su solución usando solo un lenguaje: Java. El GWT presenta una gran cantidad de librerías gráficas que permiten la implementación de widgets las cuales darán funcionalidad a la solución. El compilador de GWT realiza optimizaciones y un análisis estático completo de toda la base de código de GWT y, frecuentemente, genera código JavaScript que se carga y ejecuta con mayor rapidez que el código JavaScript equivalente creado de forma manual. Por ejemplo, el compilador de GWT suprime de forma segura todo el código no utilizable para asegurarse de que el archivo de secuencias de comandos compilado sea lo más pequeño posible. Una aplicación desarrollada con GWT tiene dos lados fundamentales: el lado cliente, y el lado servidor. El lado cliente implementará la capa de visualización (UI) mediante widgets, mientras que la parte servidora implementará la lógica fundamental de la aplicación. En el lado cliente no se permite el uso de todas las librerías Java lo cual es obvio si se tiene en cuenta las limitaciones de visualización presente en los navegadores.

4.4.1. Porque usar programación Java y desarrollos Web AJAX

- Chequeos de tipos estático aumenta productividad reduciendo errores.
- Errores en JavaScript son fácilmente atrapados en tiempo de compilación en lugar de en tiempo de ejecución.
- Diseños orientados a Objetos Java son mas fáciles de comunicar y entender, haciendo código AJAX mas comprensible con menos documentación.

4.4.2. Arquitectura

GWT tiene 4 componentes un compilador Java-to-JavaScript un navegador web embebido, y dos librerias de clases Java. La libreria GWT emuladora de jre y la libreria Web UI (ver figura 4.3).

- **El compilador Java to JavaScript:**, que convierte el codigo escrito en Java en JavaScript
- **Navegador Web GWT embebido:** permite correr y ejecutar aplicaciones GWT sin compilar a JavaScript.
- **La libreria jre emulada:** GWT contiene implementaciones JavaScript de una gran cantidad de clases de las librerias Java Standard incluyendo java.lan y un sub conjunto del paquete java.util. El resto de los paquetes (por ejemplo java.io) no son soportados pues no aplican a aplicaciones web puesto que acceden a la red y al sistema de archivos local.
- **La libreria de UI de GWT:** es un conjunto de interfaces personalizadas e interfaces que permiten crear los widgets, como botones, area de textos, etc.

[1] Ajax, definición encontrada en Wikipedia: acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y reusabilidad en las aplicaciones.

4.5. Web Map Service y Web Feature Service

El Open Geospatial Consortium (OGC) a definido varios servicios Abiertos para acceder a datos geográficos. El Web Feature Service (WFS) y el Web Map Service (WMS). Mientras el WFS es relacionado con el acceso directo a los datos: leer, escribir y actualizar las características de los mismos (features). El WMS es relacionado con la transformación de los datos en un mapa (imagen). Una feature es un objeto que abstrae el fenómeno real. Este objeto tiene un conjunto de propiedades asociadas, cada una de las cuales tiene un nombre, un tipo, y un valor. Un ejemplo de feature podría ser un camino con un nombre una ubicación (linea geométrica), límite de velocidad, etc. Típicamente estas features son almacenadas en una base de datos espacial un shapefile o cualquier otro formato vectorial.

4.5.1. Web Feature Service

El Web Feature Service (WFS) es un estándar creado por el OGC que establece la manera de enviar y recibir datos geo-espaciales a través de HTTP. Lo cual logra a través del Geography Markup Language (GML) un subconjunto de XML. La versión actual de WFS es la 1.1.0, existen notorias diferencias entre las versiones aunque la sintaxis permanece en general siendo la misma. La principal diferencia existente entre WMS y WFS radica en que WMS envia los datos a través de HTTP luego de que los datos geográficos han sido procesados en forma de imagen. Se puede pensar a WFS como el código fuente de los mapas enviados que pueden ser vistos mediante WMS. Cada uno de los protocolos soporta diferentes tipos de operaciones sobre los datos, para WFS las operaciones permitidas son:

- **GetCapabilities:** recibe una lista de datos que redicen en el server operaciones y parámetros.
- **DescribeFeatureType:** recibe información y atributos sobre un conjunto de datos en partícula.

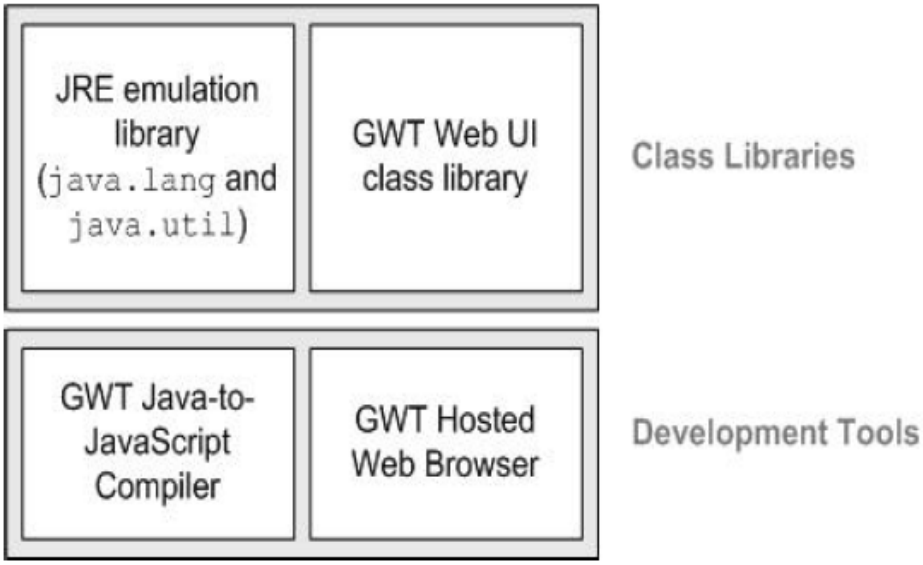

Figura 4.3: Arquitectura general de GWT.

- **GetFeature:** recibe los datos actuales incluyendo el atributo geométrico y los atributos de referencias.
- **LockFeature:** previene que un tipo feature sea editado.
- **Transaction:** edita los tipos features creando nuevos, actualizándolos o borrándolos.
- **GetGMLObject:** presente solo en la versión 1.1.0; recibe elementos dentro del XML a través de identificadores.

4.5.2. Web Map Service

WMS provee una interfase estándar para enviar imágenes geo-espaciales a través de internet. El principal beneficio es que los clientes no procesan demasiada información, los clientes solo deberán procurar mostrar correctamente la imagen que envió el servidor. El estándar asegura que es posible solapar distintos mapas en una sola imagen para su procesamiento. Las operaciones permitidas por WMS son:

- **GetCapabilities:** recibe una lista de datos que residen en el server y parámetros WMS permitidos.
- **GetMap:** recibe el mapa pedido por el usuario.
- **GetFeatureInfo:** recibe los datos asociados al mapa solicitado.
- **DescribeLayer:** Indica si el cliente requiere información adicional al mapa.
- **GetLegendGraphic:** Obtiene la leyenda asociada al mapa solicitado.

4.6. Geomajas

Geomajas es un framework Open Source enterprise-ready GIS para aplicaciones de mapas en la web [Geomajas, 2011]. Esta descripción establece que es realizado para su uso en empresas/instituciones que requieran productividad y operatividad inmediata a bajo coste, requerimiento fundamental en el presente trabajo. Tiene una buena implementación de políticas cliente-servidor para mostrar y editar datos geográficos [Geomajas, 2011], lo que cual se traduce en una arquitectura bien definida que facilita la visualización de datos de cualquier origen. Geomajas tiene seguridad integrada y es altamente escalable. Es compatible con estándares OGC (Open Geospatial Consorsium) tales como WMS, WFS, y soporta base de datos espaciales. Geomajas, provee funcionalidad “out-of-the-box” mediante plug-ins. Al aprovechar GWT en el lado cliente, el desarrollo es en java y mas eficiente según el sistema a implementar. El Objetivo principal de Geomajas es proporcionar una plataforma para la integración de diferentes capas de datos geo-espaciales, permitiendo que múltiples usuarios controlen y gestionen dichos datos. En esencia, Geomajas proporciona un conjunto de bloques de construcción de gran alcance, desde los cuales se pueden construir, complejas aplicaciones GIS. Cuenta con herramientas de edición, medición, consultas avanzadas y análisis. Desde un punto de vista técnico es implementado en Java sobre Spring 4.2 lo cual nos asegura un diseño y una implementación correcta desde el punto de vista de diseño. Un GIS que sirve sus datos a través de Internet es intrínsecamente complejo en lo que se refiere a performance y programación. Complejidad que se

desprende principalmente de la gran cantidad de tipos de interfaces necesarias. Debido a esto es necesario tener particular cuidado de la arquitectura y diseño a adoptar. El uso de Geomajas y junto con el Spring, permite despreocuparnos de detalles de diseño bien resueltos en este tipo de frameworks, ligando dicha responsabilidad a Geomajas y con el, a Spring. Geomajas implementa el MVC usando el framework Spring, donde para la implementación del objeto View, presenta la posibilidad de escoger entre 2 tipos de tecnologías: el Geomajas dojo-face o el Geomajas gwt-face (ver 4.4). La primera resuelve la interfase del usuario usando la librería de widgets en JavaScript dojo, la cual se provee en la actualidad únicamente por cuestiones de compatibilidad. Hasta la versión 1.4 de Geomajas, esta fue la única face disponible, sin embargo, y a partir de la version 1.5 Geomajas también provee la GWT face. Permitiendo así que todo el desarrollo sea realizado en Java y GWT es el encargado de manejar la conversión a JavaScript para el código que correrá en un navegador [Google, 2011]. Obviamente esta face se integrará mejor con aplicaciones basadas en GWT aunque puede ser combinada con otros frameworks de igual manera (Este hecho puntual es clave en la elección posterior de la tecnología usada en la implementación de la unidad uploader del subsistema DT). En resumen, Geomajas se integra muy bien con dojo pero usando esta face existe la desventaja de tener que programar tanto en JavaScript (lado cliente) como en Java (lado servidor) complicando la tarea de programación y debugging. Desventaja no presente usando el GWT-face.

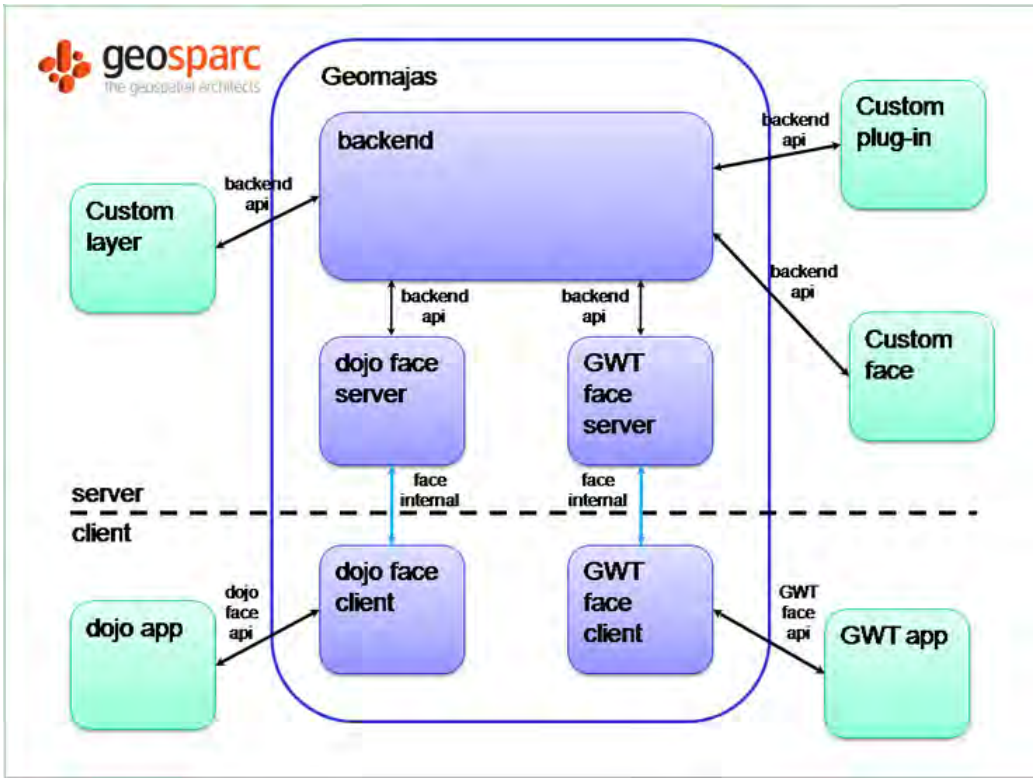

Figura 4.4: Arquitectura de Geomajas, extraido de: Geomajas user guide for developers, capítulo Architecture

Como se puede observar en la figura 4.4 Geomajas presenta un módulo back-end donde son configurados los mapas, capas y atributos o características (features) de los mismos. El back-end esta en el lado servidor y tiene una API para la interacción con el mundo exterior mediante extensiones denominadas plug-ins. El principal objetivo del back-end es contener el código para proveer vectores,

bitmaps, y datos acerca de las características (features) de la información geográfica a manipular. Sin embargo, no contiene código alguno referente a la interfase con el usuario dejando el mostrado y editado de información como tarea para las faces. El back-end es agnóstico de la web o de otro modo de visualización. La modularidad de Geomajas permite, por ejemplo construir una aplicación java swing usando el back-end, o de otra manera mejorar las faces existentes a logrando aun mayor versatilidad. A su vez, las faces son separadas en dos módulos, un modulo servidor, que dialoga directamente con el back-end usando java, y serializa los datos al cliente; y un lado cliente que dialoga solamente con el lado server del mismo. La comunicación entre ambas sub partes es privada a la face, la cual provee una API del cliente. Esta API provee detalles acerca de widgets, parámetros para estos y otras posibles interacciones como cola de mensajes. Como podemos ver (figura 4.5) el back-end es construido sobre diversos módulos asociados usando el framework Srping. El módulo Geomajas-Api y el módulo Geomajas-impl son módulos que facilitan el acceso al back-end.

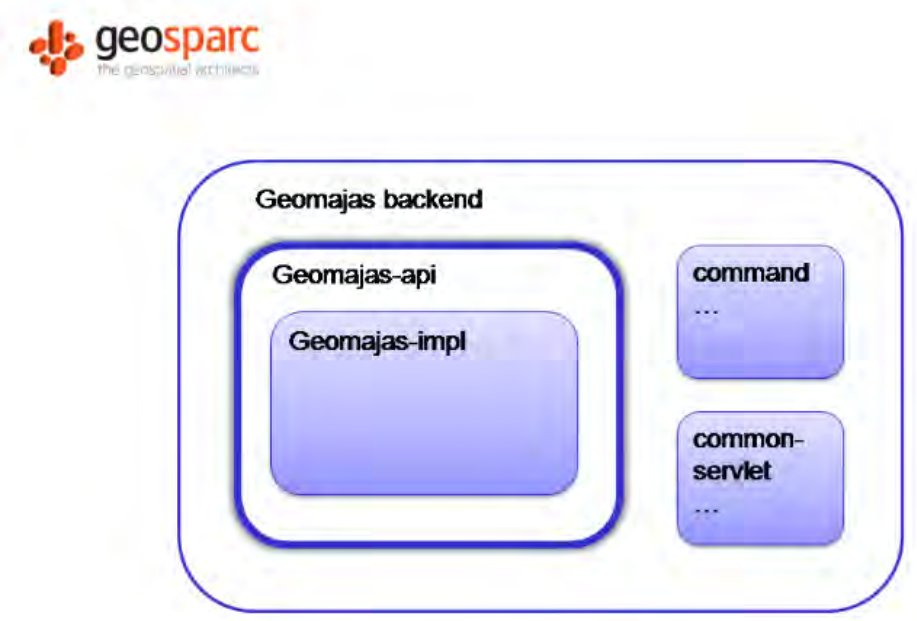

Figura 4.5: Back-end de Geomajas, extraido de: Geomajas user guide for developers, capítulo Architecture

Existen cuatro posibles formas de extender el back-end:

command: los comandos son usados como punto de interacción entre las faces y el back-end. El envío de mapas y features, los cálculos y actualizaciones en las features y funcionalidad que este en el lado cliente es realizada usando comandos

layer: agrupa un conjunto de opciones de acceso a los detalles de las capas de un mapa. Una capa puede ser raster o vector, puede ser editable o no, las features que describen los objetos que son partes de las capas vector son accesibles mediante el "feature model"que convierte features genéricas en cosas que Geomajas pueda usar

pipeline: todos los servicios de back-end que se refiere a capas son implementados usando pipelines. Configurando pipelines se puede cambiar el método de dibujado de una capa, o pasos extras en el dibujado (cacheo de features)

security: desde aquí se puede configurar los servicios a los cuales determinado usuario puede acceder. Estos objetos de autorización pueden leer políticas de seguridad desde un almacén de políticas (policy store).

4.7. GeoServer

GeoServer es un servidor de mapas escrito en Java mediante el cual es posible publicar datos a través de la web. Permite gran interoperatividad usando estándares abiertos. Es un proyecto de desarrollo abierto (OSS) GeoServer es la implementación de referencia del OGC, WFS y el WMS.

4.8. Maven

La gran cantidad de tecnologías convergentes, hace necesario una manera práctica y robusta de manejar todo el proyecto. Como es posible ver a lo largo de todo este capítulo se describieron las distintas tecnologías abordadas a fin de implementar la solución al problema principal de proporcionar una infraestructura informática para el SAT/ERDNU. Si bien hoy en día el objetivo principal en el desarrollo de todo sistema informático es la modularidad y por consiguiente el re-uso, es necesario tener en cuenta que gran modularidad genera problemas de mantenimiento de las interacciones entre las tecnologías intervinientes. El problema de versionado de las mismas, junto con sus incompatibilidades son solo dos de la numerosa cantidad de cuestiones a abordar a la hora de implementar la solución. Con Maven desarrollos de estas características se simplifican enormemente gracias a sus características de gestión y construcción de proyectos Java. Maven es un proyecto del Apache Software fundation y utiliza un modelo de configuración simple, basado en formatos XML. Su arquitectura se basa en plug-ins que permiten extender la funcionalidad del mismo en caso que se necesite, aunque el uso y soporte en lenguajes distintos de Java es mínimo. Actualmente existe además del plug-in para Java un plug-in para .Net. Maven utiliza un Project Object Model (POM) para describir el proyecto de software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos. Viene con objetivos predefinidos para realizar ciertas tareas claramente definidas, como la compilación del código y su empaquetado. Una definición informal extraída de [The Apache Software Foundation, 2011] establece que Maven es una manera fácil de publicar información y compartir Jars entre distintos proyectos. Maven está construido alrededor de la idea de reutilización de la lógica de construcción y como los proyectos generalmente se construyen en patrones similares, una elección lógica es reutilizar los procesos de construcción.

4.9. Consideraciones

En general, proyectos OSS presentan una pobre documentación en sus desarrollo, falencia que se agrava en la ausencia de comentarios dentro del código Ullah and Khan [2011]. Es obvio que desarrollos en estas condiciones requieren de mayor esfuerzo para codificar nuevas funcionalidades. No es un caso aparte el presente trabajo, donde es necesario un completo análisis de las herramientas

OSS a utilizar. En Geomajas, por ejemplo, existen manuales en su sitio web Geomajas [2011] aun no terminados que son importantes a la hora de su configuración, desarrollo y mantenimiento. Otro inconveniente técnico abordado es la dificultad en converger un entorno estable de desarrollo. La gran cantidad de tecnologías que intervienen, deben ser entrelazadas armónicamente para que puedan interactuar correctamente, pero mientras mayor es el numero de las mismas, mayor es la complejidad para lograr su correcta operatividad. El testing es otro de los puntos débiles con los que aun cuentan los OSS. Los mismos son frecuentemente entregados al público con un menor testing. En Ullah and Khan [2011] se establece como justificación a este punto el hecho de que en OSS el contribuidor (participante) tiene la libertad de trabajar en cualquier componente del proyecto. Ningún trabajo es asignado y ningún tiempo limite es asignado. Por lo tanto, en general, el participante tiende a estar interesado en tareas como el desarrollo, dejando de lado al testing, provocando que releases del producto esten carentes de testing.

Capítulo 5

Implementación

A continuación se describe de manera técnica la implementación de cada uno de los componentes (subsistemas y unidades) del SAT/ERDNU como así también sus interacciones. A lo largo del presente capitulo se utiliza la información descripta en el capítulo anterior sobre las características de las tecnologías para justificar y dar soporte a las decisiones tomadas en la implementación.

5.1. Subsistema DT

El subsistema DT es el encargado de recibir las planillas ingresadas por el usuario, las cuales mantienen información importante a la hora del cálculo de algoritmos producto del relevamiento y evaluación del mismo; validar dichas planillas y manejar los mensajes de retorno acerca la aceptación o no de la misma. Realizar las Validaciones en la Planilla ingresada por el usuario, guardar dicha planilla en formato xml para ser leída por el subsistema SD, enviar un correo de verificación al encargado de los datos acerca de la aceptación o no de dicha planilla y comunicar el resultado de dicha validación al usuario son algunas de las tareas que en particular se realizan para satisfacer los requerimientos a nivel Nacional (ERDN). Toda esta funcionalidad fue implementada mediante un Web Service que brinda dichos servicios. Si pensamos en unidades, la unidad Uploader es la encargada de subir el archivo xls al servidor. Luego, un analizador de xls (unidad xls2xml) traduce la planilla a formato xml a ser enviado por un Web Service consumer al Web Service provider. Una nueva planilla en el sistema es fácilmente agregada, mediante la creación de un nuevo Web Sevice. Un cambio en alguna planilla existente, es fácilmente incorporado modificando el contrato entre en consumer y el provider de dicho Web Service. De esta manera se garantiza versatilidad y re-uso de cada unidad en el subsistema y de su arquitectura. Incluso el proceso de validación puede cambiar y aun así, la arquitectura ajustará. Supongamos el caso en que es necesario la validación de la planilla por parte de una encargado en el proceso. Sabemos que datos de salud son sensibles y pensar en que el sistema puede requerir que una persona valide los datos por planilla no es descabellado. En este caso, la planilla se transformará, hipotéticamente, en una orden que será procesada y que necesitará tiempo para tal fin. En este caso la arquitectura será la misma, el web service será el mismo, siendo solo necesario implementar nuevamente el servicio de respuesta a la validación para que devuelva una respuesta no inmediata cuando el estado de la orden sea aceptada por el operador.

5.1.1. Unidad uploader

La unidad uploader (figura 3.2) es la encargada de brindar la UI de ingreso de las planillas epidemiológicas ERDN (un prototipo de dicha planilla puede ser vista en la figura 5.1). La planilla epidemiológica ERDN, resume información por localidad acerca de la circulación viral, el control vectorial, y el riesgo Entomológico de dengue del lugar. Es así que aproximadamente datos de 3500 planillas de localidades serán procesadas para, una vez almacenados correctamente, ser procesadas para junto con información ambiental proveniente del Sensado Remoto calcular el Riesgo de Dengue a Nivel Nacional. En términos técnicos dicha unidad es la encargada de subir la planilla xls a una carpeta temporal en el servidor y mostrar la respuesta de la validación de los datos al usuario. La figura 5.2 resume de una manera pseudo java gráfica sus atributos y métodos. Esta unidad es suficientemente modular y es independiente de la planilla a subir.

5.1.2. Unidad xls2xml

La arquitectura de la misma presenta 2 módulos principales (ver figura 5.3). El módulo xls2xml, es el encargado de transformar los datos de la planilla epidemiológica ERDN XLS en un formato XLM. Para realizar el mapeo necesario entre objetos Java y elementos XML se utilizaron las librerías Castor y Castor Marshalling de Spring. El segundo módulo, el módulo sender, es la implementación a través de Spring del lado consumer del Web Service que realizará las validaciones correspondientes en los datos ingresados y se encarga de enviar la planilla usando el protocolo SOAP al lado provider del Web Service. El punto clave para adoptar dichas tecnologías fue la alta probabilidad de cambio en el modo de ingreso de los datos. Puede darse el escenario donde sea necesario cambiar la manera de ingreso de los datos y aún así el subsistema DT soportará el cambio. Puesto que internamente los datos son validados independientemente del formato de los mismos para lo cual son mapeados de manera inmediata a elementos XMLs. Dicho escenario implica, por ejemplo, que los datos sean ingresados mediante un formulario web, gracias al diseño logrado, el único módulo a cambiar será el xls2xml digamos por un nuevo modulo form2xml. De esta manera, claro esta, prevenimos el impacto adverso que tienen cambios en unidades funcionales que ya han sido aceptadas y por consiguiente testeadas.

5.1.3. Unidad Translator

En principio y por el momento existe una sola unidad Translator, el mismo es un Web service provider implementado con Spring que realiza toda la lógica de validación de los datos de la planilla epidemiológica nacional. En un futuro cercano, existirá una serie de translators, uno por cada planilla a validar, mediante archivos XML correspondiente a la planilla en cuestión. Los servicios brindados por el actual translator pueden resumirse en: validación de datos, envío de correo de verificación de la información, y persistencia de los datos validados. Al definir correctamente el esquema XSD que generó el contrato WSDL entre el consumer y el provider, la arquitectura y diseño elegidos aseguran intrínsecamente una correcta validación de los datos, facilitando la codificación de la misma; pues definiendo correctamente las restricciones sobre los datos en el XSD, el uso de Web Services

	A	B	C	D	E
1		***Estratificación de Areas de Riesgo para Dengue***			
2					
3	**Localidad:**	Oran		**Riesgo:**	**2,4**
4	**Departamento:**	Oran		**Prioridad:**	**MEDIA**
5	**Provincia:**	Salta		**Año:**	2011
6	**Valoración del riesgo:**	bajo		**Temporada:**	2.- Semestre Invernal: Mayo a Octubre
7	**Codigo:**	66126070			
8					
9	***Componente Circulación***			***Componente Circulación***	
10	**Antecedentes de transmisión viral**	4.- Con transmision viral autóctona previa		Valoración Antec. Transm. Viral	4
11	**Vigilancia sindromica**	2.- Se realiza notificación rutinaria semanal de SFAI		Valoracion Vigilancia sindromica	2
12	**Flujo Poblacional**	4.- Transito estacional y/o permanente desde zonas de riesgo		Valoración Flujo Poblacional	4
13	**Localidad fronteriza c/Bolivia, Paraguay y Brasil**	4.- SI		Valoración Area o Localidad Fronteriza	4
14	**Densidad Poblacional**	1.- Hasta 2000 personas/Km2		Valoración Densidad Poblacional	1
15				**Valoración Componente Riesgo Transmision Viral**	3,00
16					
17	***Control Vectorial***			***Control Vectorial***	
18	**Periodicidad del Control de criaderos**	1.- Permanente como actividad rutinaria		Valoración Personal afectado al Control Vectorial	1
19	**Personal permanente afectado al Control Vectorial/Viviendas**	1.- Un técnico especializado cada 800 viviendas (aprox. 25 manzanas urbanas)		Valoración Periodicidad del Control de Criaderos	1
20				**Valoración Componente Riesgo Control Vectorial**	1,00
21					
22	***Riesgo Entomológico***			***Riesgo Entomológico***	
23	**Presencia del vector**	4.- SI		Valoración presencia/ausencia del vector	4
24	**Indices Aedicos**	4.- > al 20%		Valoración Indice Aedicos	4
25	**Provision de agua de red**	2.- Hasta 80 % de viviendas con provision permanente de agua de red		Valoración Provisión de Agua de Red	2
26				**Valoración Componente Riesgo Entomologico**	3,33

Figura 5.1: Planilla epidemiológica ERDN

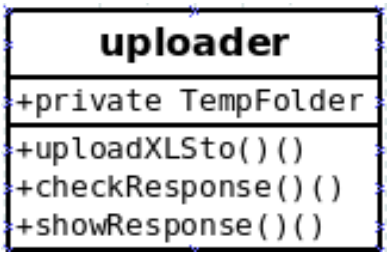

Figura 5.2: Diseño Técnico de la unidad uploader

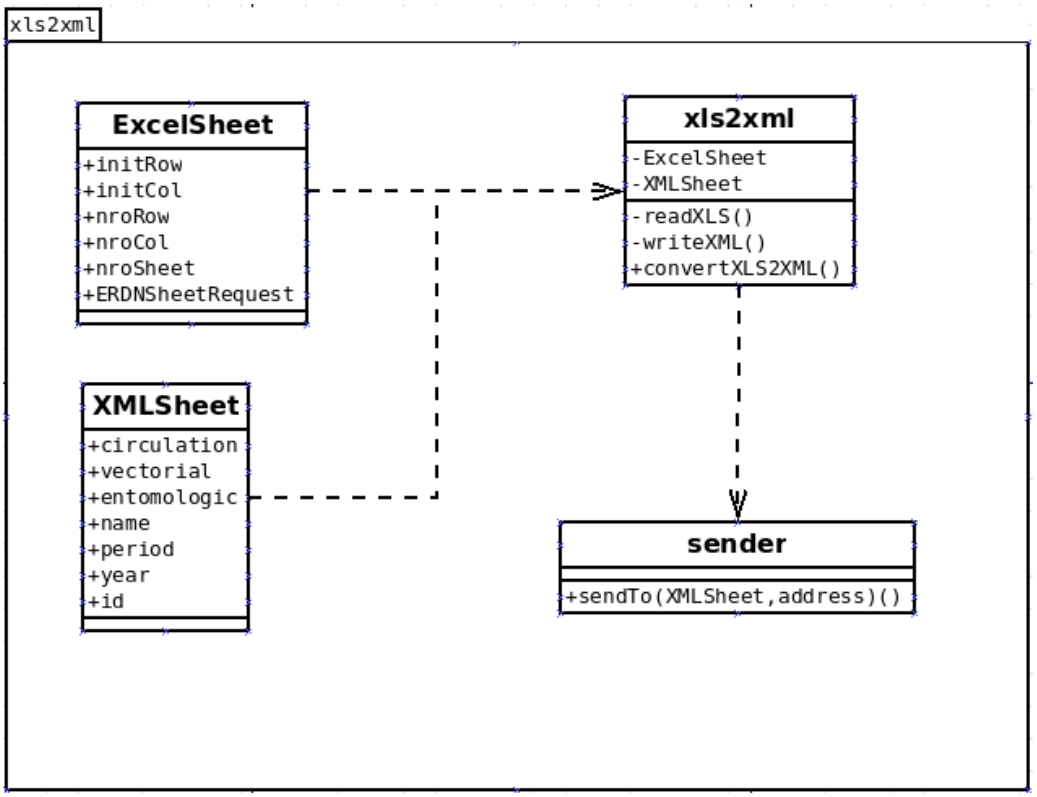

Figura 5.3: Diseño Técnico de la unidad xls2xml

soluciona gran cantidad de problemas correspondientes a largas validaciones. De esta manera, la imposibilidad de enviar datos entre consumer y provider, que no se correspondan con el XSD definido hace casi directa la implementación de la validación. De esta manera y usando contract-first de Spring un cambio en el tipo de datos en la planilla epidemiológica implicará un simple cambio en el esquema, sin la necesidad de re-codificar la unidad. La figura 5.4 muestra a modo de ejemplo restricciones establecida en el tipo de datos del componente de circulación de la panilla epidemiológica. El servicio de envío de mail, consiste en el envío de un correo electrónico de cada planilla xls ingresadas al jefe encargado de gestionar los datos en cada localidad. Dicho servicio fue implementado usando IoC de Spring para lograr una alta capacidad de configuración. Por último el servicio de persistencia de los datos se encarga de, una vez validados los mismos, almacenarlos en el servidor en formato XML. Formato que es adoptado debido a restricciones existentes en la arquitectura destino de todo el SAT/ERDNU: el CUSS de CONAE. Y que gracias a la alta modularidad de todo el subsistema DT se logró independientemente de la capacidad de soportar cambios, en este caso en la implementación de este servicio, de manera tal que soporte persistencia en bases de datos relacionales por ejemplo. A modo de resumen, el apéndice B expone el contrato publicado entre el consumer y el provider.

5.1.4. Interface: Uploader - xls2xml - Translator

En el proceso de ingreso de datos es necesario enfrentar la posibilidad de existencia de una numerosa cantidad de mensajes de retorno, los cuales pueden ser a su vez de distintos tipos y mas aun cuando se trata de una numerosa cantidad datos a validar e ingresar al sistema. Es por ello que la interfase entre el Web Service Consumer y la unidad Uploader se implementa a través del uso de respuestas xml. De esta manera las respuestas por parte del Web Service Provider es enviada al Web Service Consumer que luego escribe la misma en formato xml. Finalmente la unidad Uploader lee el repositorios de datos de respuesta y muestra la misma al usuario. La codificación xml de la respuesta esta compuesta por un estado y una descripción (figura 5.5). Dependiendo del error cometido y el lugar, el Web Service Provider establece la descripción del error a través de log4j.

```xml
<element name="componenteCirculacion">
    <complexType>
        <all>
            <element name="ValoracionAntecTransmViral">
                <simpleType>
                    <restriction base="string">
                    <pattern value="[0-4]"/>
                        <minLength value="1"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="ValoracionVigilaciaSindromica">
                <simpleType>
                    <restriction base="string">
                    <pattern value="[0-4]"/>
                        <minLength value="1"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="ValoracionFlujoPoblacional" >
                <simpleType>
                    <restriction base="string">
                    <pattern value="[0-4]"/>
                        <minLength value="1"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="ValoracionAreaLocalidadFronteriza" >
                <simpleType>
                    <restriction base="string">
                    <pattern value="[0-4]"/>
                        <minLength value="1"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="ValoracionDensidadPoblacional" >
                <simpleType>
                    <restriction base="string">
                    <pattern value="[0-4]"/>
                        <minLength value="1"/>
                    </restriction>
                </simpleType>
            </element>
        </all>
    </complexType>
</element>
```

Figura 5.4: Componente de Circulación Vectorial, restricciones intrínsecas al diseño que facilitan las validaciones

5.2. Subsistema SD

El subsistema Spatial Data (SD) es el encargado de almacenar todo los datos del SAT/ERDNU. Manteniendo una alta compatibilidad con la arquitectura presente en el CUSS de CONAE, la implementación de este subsistema es básicamente la capacidad de re-uso de componentes de software existentes en el subsistema ARCH del CUSS. El subsistema ARCH del CUSS es el subsistema encargado del almacenamiento de los datos espaciales del segmento del usuario de CONAE [De La Torre, 2010]. Este subsistema es compuesto de diversas unidades, una de las cuales es un file transfer manager: ftmanager que aporta la funcionalidad de copiar archivos entre diferentes maquinas dentro de una red. Entonces la unidad copy manager del subsistema SD es básicamente reusar la unidad ftmanager disponible a través del ARCH.

5.3. Subsistema AE

Los modelos matemáticos que luego se transforman en algoritmos que ejecuta el subsistema AE, son actualmente desarrollados por profesionales de distintas disciplinas. Por ende se hace difícil la unificación en un lenguaje de programación el desarrollo de los mismos. Para resolver este problema es necesario encapsular la implementación de los algoritmos brindando interfaces homogéneas a la unidad manager de algoritmos: wrappers. De esta manera se logra independencia de la plataforma, pues un wrapper brindará la interface necesaria para que el algorthm manager pueda cumplir su tarea y abstraerá de la tecnología usada en su implementación. Al momento de la presentación de este trabajo se desarrolló un algoritmo que consiste en calcular el riesgo de dengue a partir de datos ambientales provenientes del SR. El proceso genera una mapa raster con la estratificación del riesgo del dengue en la república Argentina. El mapa raster generado por el proceso ambiental será luego entrada de un proceso posterior para el cálculo del riesgo de dengue final. El cual además del mapa raster de riesgo ambiental recibirá como entrada los datos epidemiológicos almacenados en las planillas XMLs. Finalmente otro proceso es el encargado de generar un archivo vectorial con los datos de riesgo de dengue por localidad, dicho proceso raster2shp es genérico y es el encargado de convertir los archivos

```xml
<?xml version="1.0" encoding="UTF-8"?>
<oss:planillaERDNResponse xmlns:oss="http://conae.gov.ar/ws/schema/oss">
    <oss:code>SUCCESS</oss:code>
    <oss:description>Planilla aceptada</oss:description>
</oss:planillaERDNResponse>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<oss:planillaERDNResponse xmlns:oss="http://conae.gov.ar/ws/schema/oss">
    <oss:code>FAILURE</oss:code>
    <oss:description>Descripción del fallo en la planilla</oss:description>
</oss:planillaERDNResponse>
```

Figura 5.5: Codificación XML de la respuesta.

rasters de cualquier proceso dentro del subsistema AE (ver figura 5.6) a formato vectorial. A modo de ejemplo a continuación se muestra un java pseudo código de la unidad Process_RiesgoAmbiental que calcula el riesgo de dengue como se dijo anteriormente.

```
Module Process_RiesgoAmbiental(SerieDeTemperatura temp, MapaProbPresencia mapProb)

//Constants:
//grados centigrados
final int TEMP_MIN 12;
//numero de dias necesarios com temperatura mayor a  T_minima
final int DIAS_TEMP_MIN 20;
//dias
final int TIEMP_VIDA_MAX 20;
final int RESETEO 5 //grados

public RasterEstratificacion riesgoAmbiental(){
  int dia = 0;
  while (dia < size(Temp)){
    dia = evalDias(DIAS_TEMP_MIN,TEMP_MIN,Temp,RESETEO)
    while (Temp[dia]>5) {
        if (acumuladorPEIP(dia,TIEMP_VIDA_MAX, Temp,RESETEO)>0.99) {
          ciclos++;
        }
        dia++;
    }
  }

  ciclosnorm = normalizacion(ciclos);
  riesgo = sqrt(mapProb*ciclosnorm);
  riesgoambiental=  cuatroCategorias(riesgo);

private evalDias(DIAS_TEMP_MIN,TEMP_MIN,Temp, RESETEO){
  /*
    Devuelve un numero entero que representa el ultimo dia de la primer
    serie de DIAS_TEMP_MIN con Temp  mayor a  T_MIN. Si  Temp es menor
    a RESETEO se resetea la serie y se sigue buscando la siguiente serie.
  */
}

private int acumuladorPEIP(dia,TIEMP_VIDA_MAX, Temp, RESETEO){
  while (dia < dia+TIEMP_VIDA_MAX and temp > RESETEO){
    acumPEIP = acumPEIP + peip(dia);
  }
  return acumPEIP;
}


private peip(dia){
  /*
    calcula la proporcion del periodo de incubacion alcanzado en el dia
    peip significa proporcion del periodo de incubacion alcanzado
    cumplido
  */

private normalizacion(ciclos){
  /*
    normaliza la variable ciclos dividiendo por el maximo numero de
    ciclos.
  */

private cuatroCategorias(riesgo){
  /*
  divide la variable riesgo en cuatro categorias.
  */
}
```

La unidad Manager general es la encargada de ejecución de los algoritmos en modo batch. La figura 5.6 muestra de manera esquemática como es implementada la misma, haciendo uso del proceso Cron de sistemas operativos Linux. Donde dada una entrada de tiempo exacta (1 de abril), el algoritmo es ejecutado a través del archivo de configuración correspondiente (el crontab) para finalmente obtener el mapa de estratificación de dengue de la República Argentina.

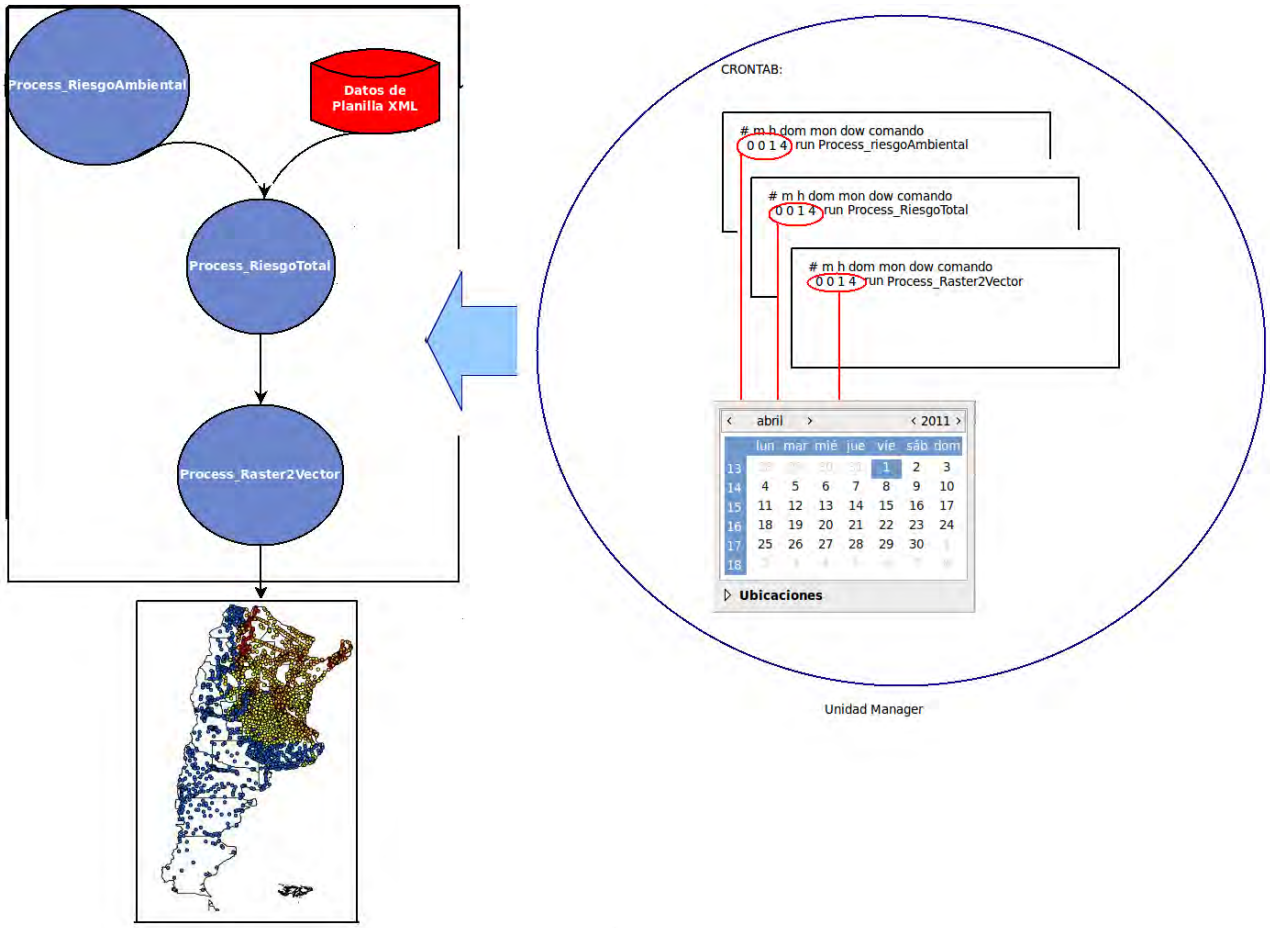

Figura 5.6: Proceso completo y ejecucion de los mismos por la unidad manager

5.4. subsistema VZ

Es requerimiento el correcto acceso y visualización de los datos procesados por el subsistema AE. Además de diversas capas bases, la infraestructura informática tiene que permitir la visualización de los datos de salida de los algoritmos de manera correcta. Para ello se implementó un complejo Sistema de Información Geográfica altamente configurable que permite el acceso a la información espacial validada por el subsistema DT, procesada por el subsistema AE y almacenada por el subsistema SD.

5.4.1. Unidad GeoServer

Esta unidad publica los datos a ser visualizados por la unidad GIS Web Viewer permitiendo su interoperatividad y versatilidad. La misma es básicamente un servidor de mapas que abstrae al usuario

de las dificultades de conexión propiamente asociadas a datos geográficos. Debido a lo ya mencionado en el capítulo anterior GeoServer es la tecnología ideal para la implementación de esta unidad. Nuevamente vale la pena mencionar, que es posible el cambio de la implementación de la unidad GeoServer sin impactar profundamente en todo el sistema gracias a la modularidad presente en todo el subsistema VZ. Otro factor importante a la hora de la elección de GeoServer como servidor de mapas es que el sistema es concebido para ser accesible en un ambiente productivo, lo que deberá garantizar la facilidad en la implementación de mínimos cambios como lo puede ser el agregado de capas bases al sistema. GeoServer presenta una UI amigable y de fácil uso que permite la administración de los datos geográficos a publicar.

5.4.2. Unidad GIS Web Viewer

Es necesaria la correcta visualización de los datos brindados por el Servidor de Mapas. Un GIS que sirve sus datos a través de Internet es intrinsecamente complejo en lo que se refiere a performance e implementación. Complejidad que se desprende principalmente de la gran cantidad de tipos de conexiones disponibles. Debido a esto es que es necesario tener particular cuidado de la arquitectura y diseño a adoptar para esta unidad. El uso de Geomajas y de Spring en la unidad de GIS Web Viewer, permite despreocuparnos de detalles bien resueltos en este tipo de frameworks, ligando la responsabilidad del diseño de dicha unidad a Geomajas y con el, a Spring. El estudio de Geomajas incluyó el uso de su aplicación por defecto, el arquetipo de Geomajas que puede ser descargado usando Maven2. Dicho arquetipo es la aplicación punto de partida para comenzar a usar Geomajas. Si es necesario la modificación del framework, el desarrollador debe acceder a los repositorios SVN en https://svn.geomajas.org/majas/ donde se encuentra alojado actualmente todo el proyecto con sus diversas releases. Un punto clave de geomajas es su versatilidad y flexibilidad lo que facilita la implementación de diferentes tipos de repositorios geográficos.

5.4.3. Configuraciones

Como se mencionó anteriormente la complejidad propia de todo GIS es su capacidad para manejar datos de cualquier fuente. Geomajas presenta varios tipos e interfaces para la correcta configuración de los accesos a través de Spring. La configuración es construida configurando las clases y es dividida en lado-cliente y back-end. Solo las clases back-end son necesarias para configurar el el comportamiento de las capas puesto que el mismo puede ser visto como un catálogo de capas generales. Las configuraciones en las clases del lado cliente son usadas para definir la aplicación y los mapas, los cuales son conceptos del cliente en la arquitectura de Geomajas. Cada archivo de configuración contiene una o mas definiciones de beans correspondientes a las instancias actuales de las clases Geomajas. Pueden ser establecidos tipos primitivos usando una representación string del valor. Cuando el valor de configuración sea otro bean puede ser definido in-line dentro del mismo Bean o puede usarse una referencia a el. El nombre del bean es único y la ubicación del mismo es agregado en el parámetro contextConfigLocation del archivo web.xml (ver figura 5.7)

5.4.3.1. web.xml

La configuración inicial debe ser efectuada usando el archivo *web.xml*. Es en este archivo web.xml, donde toda la configuración de la aplicación inicia. El archivo web.xml debe contener la lista de los archivos de configuración del proyecto. Que, si bien puede estar configurado íntegramente en un archivo, en la implementación de la unidad GIS Web Viewer fue separado en varios a fin de obtener configuraciones fáciles de entender. Las clases listeners inicializan el contexto de la aplicación (applicationContext) lo que vuelve necesario especificar los archivos que contienen el contexto de la aplicación en el parámetro contextConfigLocation.

5.4.3.2. applicationContext.xml

El archivo applicationContext.xml es usado para configurar especialmente tanto mapas y capas como para realizar alguna configuración adicional de seguridad y de plug-ins. Para implementar la configuración del GIS Web Viewer se adoptó la misma convención recomendada en el manual del desarrollador de Geomajas [Geosparc nv, 2011] El archivo de contexto esta en el directorio WEB-INF de la aplicación (src/main/webapp/WEB-INF) el cual contiene todo los archivos de configuración del GIS Web Viewer:

- **web.xml:** contiene, entre otras, configuraciones la lista de los archivos de configuración que se leen al iniciar el framework.
- **applicationContext.xml:** incluye el mapMain.xml, el overViewMap.xml y toda configuración general de la aplicación (pipelines, seguridad, tools, etc.).
- **mapMain.xml:** contiene la configuración principal del mapa y los Beans referidos a herramientas y a las capas a mostrar. Aquí es donde se establece la proyección y los niveles de zoom, junto con la configuración de herramientas como el árbol de capas y la visualización de las mismas.
- **capaXcliente.xml:** contiene la configuración las capas del lado cliente, incluyendo su nombre, visibilidad, leyenda, etc.
- **capaX.xml:** contiene la configuración denominada por Geomajas servidora de la capa de información geográfica. Es en este archivo donde se configura la proyección de la capa, el repositorio

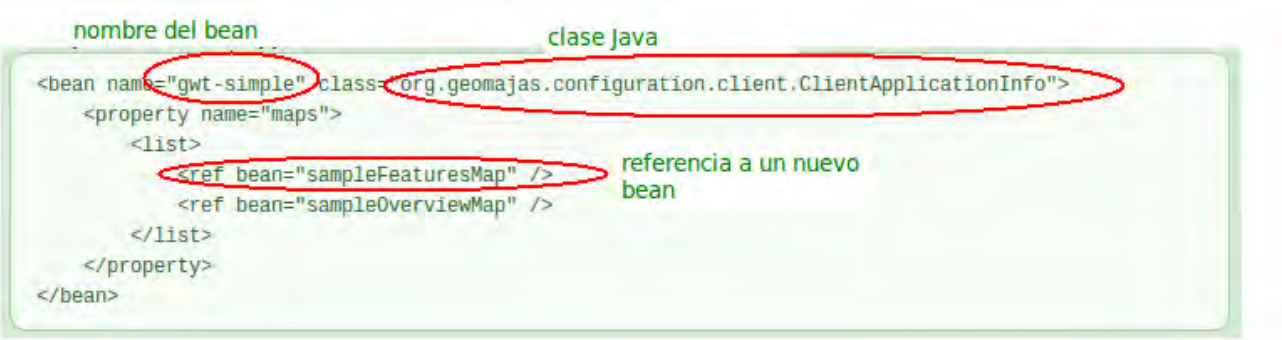

Figura 5.7: Configuración de un bean genérico. Extraído de Geomajas for Developers Manual

desde donde debe ser levantada, su estilo gráfico, su tipo (punto, linea, polígono, raster) y los atributos característicos (features) asociados a ella.

5.4.4. Configuración del Mapa: mapMain.xml

Toda capa base que muestra el subsistema VZ debe ser configurada por la unidad GIS Web Viewer, y al ser implementada por Geomajas, presenta gran versatilidad de configuración. La figura 5.8 resume los tipos de configuraciones posibles existentes para cada capa. Como se puede observar, las capas son configuradas de acuerdo a su tipo dependiendo del repositorio donde se encuentren y el plug-in usado para tal fin. Una capa Raster podrá ser accesible desde un repositorio local o usando las configuraciones WMS. A su vez es posible configurar las capas usando los plug-ins correspondientes provenientes del Open Street Map o Google mapas. Con respecto a las capas de tipo vector, pueden ser configuradas usando el plug-in GeoTool[1] y accedidas desde repositorios locales o mediante WFS y WMS. Además, Geomajas proporciona un plug-in para la abstracción de conexiones usando hibernate[2] siendo posible configurar a través de este plug-in, el acceso a bases de datos geográficas relacionales. Las siguientes secciones describen las configuraciones realizadas en las distintas capas bases de la unidad GIS Web Viewer.

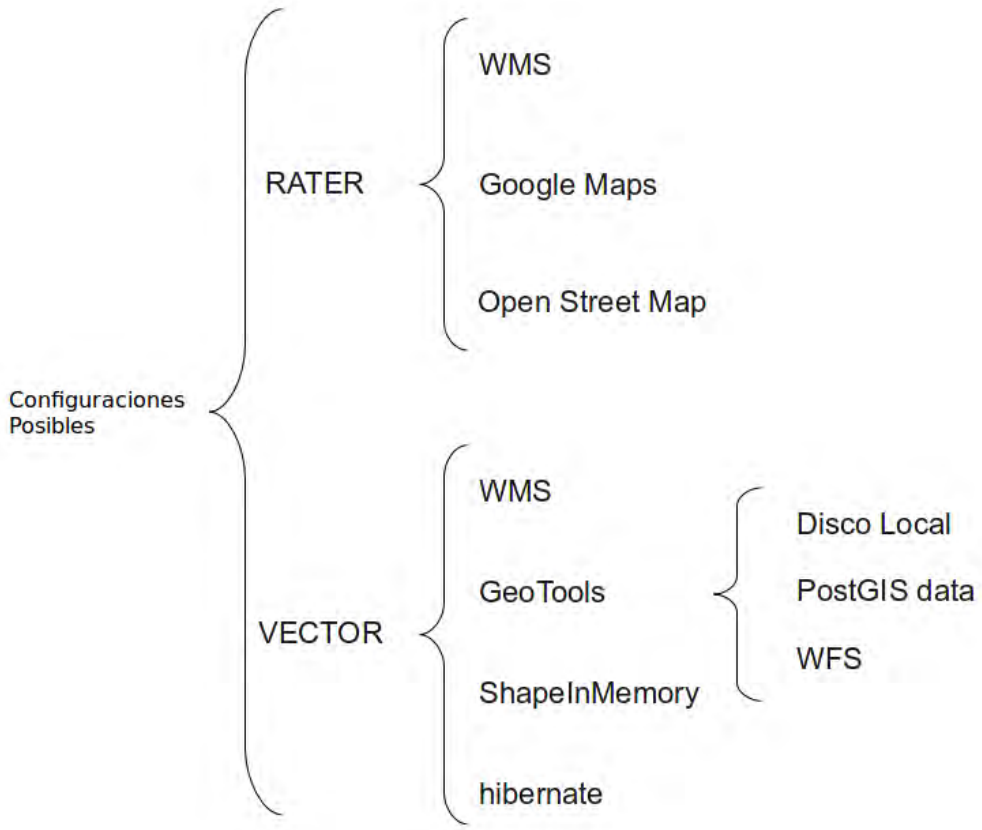

Figura 5.8: Resumen de los posibles tipos de configuración de la unidad GIS Web Viewer

[1] GeoTools es un librería Open Source ampliamente difundida y usada en el desarrollo de GeoServer
[2] hibernate es un framework dedicado a solucionar problemas de persistencia en bases de datos relacionales.

5.4.4.1. Configuración Rasters

Capas Rasters son capas basadas en imágenes que dependiendo del tipo pueden ser configuradas para recibir dichas imágenes desde servidores WMS, Google Maps o OpenStreetMap. En el presente trabajo se estudiaron las tres maneras de conexión, pero la implementada finalmente es la conexión WMS con la unidad GeoServer. Técnicamente, la capa raster implementa la interfase org.geomajas.layer.RasterLayer la cual provee acceso para objetos RasterLayerInfo para manejar los metadatos. La unidad GIS Web Viewer implementa la visualización de un mosaico construido con imágenes SAC-C la figura 5.9 y 5.10 muestran su configuraciones, donde se puede observar entre otras cosas la configuración del objeto org.geomajas.configuration.RasterLayerInfo el cual es definido para configirar la capa en el back-end (figura 5.10) correctamente.

```
<bean class="org.geomajas.configuration.client.ClientRasterLayerInfo"
    id="clientLayerSACC">
    <property name="serverLayerId" value="layerSACC" />
    <property name="label" value="Provincias Argentinas" />
    <property name="visible" value="false" />
    <property name="maximumScale" value="1:1000" />
    <property name="minimumScale" value="1:500000000" />
    <property name="style" value="1" />
</bean>
```

Figura 5.9: Configuración del Raster SAC-C back-end

5.4.4.2. Configuración Vectorial

Toda capa vector implementa la interfase org.geomajas.layer.VectorLayer estableciendo un modo de acceso para los meta-datos mediante el objeto VectorLayerInfo. Nuevamente las configuraciones son definidas en Beans de Spring como parte de las configuraciones de las capas. A modo de ejemplo (debido a la numerosa cantidad de capas base configuradas solo se presentan configuraciones de ejemplo quedando a disposición del lector el código de las configuraciones si así lo requiere) la figura 5.11 muestra las configuraciones realizadas sobre la capa vector de localidades de la República Argentina del lado cliente. Cuando se trata de una capa vector es necesario realizar configuraciones tanto en las características o features (figura 5.12) como en el estilo de la misma (figura 5.13)

5.4.4.3. Plug-ins

La unidad GIS Web Viewer provee un conjunto de funcionalidad como parte de su núcleo. A través de plug-ins esta funcionalidad es extendida logrando ser un contenedor de plug-ins. Para dar una definición precisa, es necesario pensar a los plug-ins como librerías opcionales que extienden la funcionalidad principal, aprovechando las ventajas de librerías públicas, como es el caso del plug-in de GeoTool, o el plug-in que se ocupa de roles y seguridad de los usuarios. En particular existe un subconjunto de plug-ins que se encargan del acceso a los datos actuales que necesitan ser mostrados como parte del mapa:

```
<beans
        xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:context="http://www.springframework.org/schema/context"
        xsi:schemaLocation="
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-2.5.xsd"
        >

     <bean name="SACCInfo" class="org.geomajas.configuration.RasterLayerInfo">
        <property name="crs" value="EPSG:4326"/>
        <property name="maxExtent">
            <bean class="org.geomajas.geometry.Bbox">
                <property name="x" value="-72.893"/>
                <property name="y" value="-54.805"/>
                <property name="width" value="30"/>
                <property name="height" value="40"/>
            </bean>
        </property>

        <property name="dataSourceName" value="HAPN:SAC-C-Argentina-latlon" />
        <property name="tileWidth" value="299"/>
        <property name="tileHeight" value="512"/>
    </bean>

    <bean name="layerSACC" class="org.geomajas.layer.wms.WmsLayer" >
        <property name="layerInfo" ref="SACCInfo" />
        <property name="baseWmsUrl" value="http://192.168.5.62:8080/geoserver/wms/"/>
        <property name="version" value="1.1.0"/>
        <property name="format" value="image/png"/>
        <property name="styles" value="" />
        <property name="parameters">
            <list>
                <bean class="org.geomajas.configuration.Parameter">
                    <property name="name" value="transparent"/>
                    <property name="value" value="true"/>
                </bean>
            </list>
        </property>
    </bean>

</beans>
```

Figura 5.10: Configuración del Raster SAC-C back-end

```
<bean class="org.geomajas.configuration.client.ClientVectorLayerInfo"
    id="clientLayerLocalidades">
    <property name="serverLayerId" value="layerLocalidades" />
    <property name="label" value="Localidades" />
    <property name="visible" value="true" />
    <property name="maximumScale" value="1:1000" />
    <property name="minimumScale" value="1:500000000" />
    <property name="snappingRules">
        <list>
            <bean class="org.geomajas.configuration.SnappingRuleInfo">
                <property name="distance" value="10000" />
                <property name="type" value="NEAREST_POINT" />
                <property name="layerId" value="localidadesLayer" />
            </bean>
        </list>
    </property>
    <property name="namedStyleInfo" ref="layerLocalidadesStyleInfo" />
</bean>
```

Figura 5.11: Configuración del cliente de la capa base localidades

```
<bean class="org.geomajas.configuration.FeatureInfo" name="layerLocalidadesFeatureInfo">
    <property name="dataSourceName" value="localidades_sinave_latlong"/>
    <property name="identifier">
        <bean class="org.geomajas.configuration.PrimitiveAttributeInfo">
            <property name="label" value="loc"/>
            <property name="name" value="LOC"/>
            <property name="type" value="STRING"/>
        </bean>
    </property>
    <property name="geometryType">
        <bean class="org.geomajas.configuration.GeometryAttributeInfo">
            <property name="name" value="the_geom"/>
            <property name="editable" value="false"/>
        </bean>
    </property>

    <property name="attributes">
        <list>
            <bean class="org.geomajas.configuration.PrimitiveAttributeInfo">
                <property name="label" value="loc"/>
                <property name="name" value="LOC"/>
                <property name="editable" value="true"/>
                <property name="identifying" value="true"/>
                <property name="type" value="STRING"/>
            </bean>
            <bean class="org.geomajas.configuration.PrimitiveAttributeInfo">
                <property name="label" value="nom_depto"/>
                <property name="name" value="NOM_DEPTO"/>
                <property name="editable" value="true"/>
                <property name="identifying" value="true"/>
                <property name="type" value="STRING"/>
            </bean>
            <bean class="org.geomajas.configuration.PrimitiveAttributeInfo">
                <property name="label" value="nom_locali"/>
                <property name="name" value="NOM_LOCALI"/>
                <property name="editable" value="true"/>
                <property name="identifying" value="true"/>
                <property name="type" value="STRING"/>
            </bean>

        </list>
    </property>
</bean>
```

Figura 5.12: Configuración del objecto FeatureInfo de una capa vector

```
<bean class="org.geomajas.configuration.NamedStyleInfo" name="layerLocalidadesStyleInfo">
    <property name="featureStyles">
        <list>
            <bean class="org.geomajas.configuration.FeatureStyleInfo">
                <property name="name" value="Localidades" />
                <property name="fillColor" value="#FF3333" />
                <property name="fillOpacity" value=".7" />
                <property name="strokeColor" value="#333333" />
                <property name="strokeOpacity" value="1" />
                <property name="strokeWidth" value="1" />
                <property name="symbol">
                    <bean class="org.geomajas.configuration.SymbolInfo">
                        <property name="rect">
                            <bean class="org.geomajas.configuration.RectInfo">
                                <property name="w" value="12" />
                                <property name="h" value="12" />
                            </bean>
                        </property>
                    </bean>
                </property>
            </bean>
        </list>
    </property>

    <property name="labelStyle">
        <bean class="org.geomajas.configuration.LabelStyleInfo">
            <property name="labelAttributeName" value="LOC" />
            <property name="fontStyle">
                <bean class="org.geomajas.configuration.FontStyleInfo">
                    <property name="color" value="#FEFEFE" />
                    <property name="opacity" value="1" />
                </bean>
            </property>
            <property name="backgroundStyle">
                <bean class="org.geomajas.configuration.FeatureStyleInfo">
                    <property name="fillColor" value="#888888" />
                    <property name="fillOpacity" value=".8" />
                    <property name="strokeColor" value="#CC0000" />
                    <property name="strokeOpacity" value=".7" />
                    <property name="strokeWidth" value="1" />
                </bean>
            </property>
        </bean>
    </property>
</bean>
```

Figura 5.13: Configuración del objecto StyleInfo de una capa vector

- **geomajas-layer-hibernate (vector):** permite el acceso a cualquier tipo de feature almacenado en una base de datos relacional. A tal fin las librerias de hibernate y hibernate-spatial open source son usadas.
- **geomajas-layer-geotools (vector):** permite el acceso a datos desde cualquier tipo de vector usando las librerias GeoTools (http://geotools.org/javadocs/org/geotools/data/DataStore.html).
- **geomajas-layer-google (raster):** permite el acceso a rasters Google (normal y satelite).
- **geomajas-layer-openstreetmap (raster):** permite el acceso a datos rasters provenientes del proyecto OpenStreetMap (http://www.openstreetmap.org/).
- **geomajas-layer-wms (raster):** permite el acceso a datos desde un servidor WMS
- **geomajas-layer-shapeinmem (vector):** permite el acceso a datos de tipo ESRI shape file los cuales son manejados en memoria, aunque el actual acceso a los datos se realiza usando GeoTools (http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf).

5.4.4.4. Configuración del mapa cliente del SAT/ERDNU: Esquema general

El mapa es configurado en la unidad GIS Web Viewer a través del lado cliente del face gwt proporcionado por Geomajas. Mientras el back-end de Geomajas y por consiguiente de la unidad GIS Web Viewer, trabaja casi exclusivamente con las capas, en el lado cliente de la unidad las capas se combinan para formar mapas. Para lo cual es necesario conocer entre, otras cosas, el sistema de coordenadas de referencia. Así, el back-end no conoce nada acerca del mapa: cosas como en que mapa la capa es mostrada, o detalles de construcción del mismo son innecesarias, aunque deba conocer, en contrapartida, el sistema de coordenadas de referencia usado. La figura 5.14 resume la configuración de proyección y coordenadas del mapa principal del SAT/ERDNU. El parámetro crs se refiere a las

```
<bean name="mapMain" class="org.geomajas.configuration.client.ClientMapInfo">
    <property name="crs" value="EPSG:900913" />

    <property name="initialBounds">
        <bean class="org.geomajas.geometry.Bbox">
            <property name="x" value="-12396251.49745" />
            <property name="y" value="-7739096.23874" />
            <property name="width" value="12513658.77288" />
            <property name="height" value="6124746.20158" />
            </bean>
    </property>

    <property name="layers">
        <list>
            <ref bean="clientLayerSACC"/>
            <ref bean="clientLayerProv"/>
            <ref bean="clientLayerDeptos"/>
            <ref bean="clientLayerCuerposAgua"/>
            <ref bean="clientLayerHIDRO"/>
            <ref bean="clientLayerRutas"/>
            <ref bean="clientLayerERDNWMS"/>
            <ref bean="clientLayerERDNWFS"/>
        </list>
    </property>
```

Figura 5.14: Resumen de los Objetos a Configurar en la unidad GIS Web Viewer

coordenadas de referencia del sistema. La propiedad initialbounds determina el área visible del mapa con la primer visualización. Mientras que la propiedad layers se refiere a la información de la lista

de capas a mostrar en el lado cliente. Adicionalmente un gran numero de configuraciones puede ser incluido en las configuraciones del mapa. Esto incluye información de color de fondo (background color) y estilos que se debe usar para puntos seleccionables, lineas y polígonos. Configuraciones de habilitación de botones de paneos y barra de escalas (ver figura 5.15). El mapa tiene una barra de herramientas configurada como en la figura 5.16, donde cada una de las herramientas es configurada mediante el establecimiento de alguno parámetros como se puede ver en la figura 5.17. Finalmente el mapa del SAT/ERDNU presenta un árbol de capas configurado como en la figura 5.18 que se conecta al mapa y establece la visibilidad o no de las capas configuradas. A modo de resumen, la figura 5.19 presenta el esquema UML de configuración de los objetos del mapa principal del SAT/ERDNU con sus inter-relaciones como se ha descripto en esta sección.

```
<bean name="mapMain" class="org.geomajas.configuration.client.ClientMapInfo">


    <property name="backgroundColor" value="#FFFFFF" />
    <property name="lineSelectStyle">
        <bean class="org.geomajas.configuration.FeatureStyleInfo">
            <property name="fillOpacity" value="0" />
            <property name="strokeColor" value="#FF6600" />
            <property name="strokeOpacity" value="1" />
        </bean>
    </property>
    <property name="pointSelectStyle">
        <bean class="org.geomajas.configuration.FeatureStyleInfo">
            <property name="fillColor" value="#FFFF00" />
        </bean>
    </property>
    <property name="polygonSelectStyle">
        <bean class="org.geomajas.configuration.FeatureStyleInfo">
            <property name="fillColor" value="#FFFF00" />
            <property name="fillOpacity" value=".5" />
        </bean>
    </property>
```

Figura 5.15: Configuraciones adicionales del mapa principal del SAT/ERDNU

```
<property name="toolbar">
    <bean class="org.geomajas.configuration.client.ClientToolbarInfo">
        <property name="tools">
            <list>
                <ref bean="ZoomToRectangleMode" />
                <ref bean="PanMode" />
                <ref bean="ToolbarSeparator" />
                <ref bean="ZoomPrevious" />
                <ref bean="ZoomNext" />
                <ref bean="ToolbarSeparator" />
                <ref bean="EditMode" />
                <ref bean="MeasureDistanceMode" />
                <ref bean="SelectionMode" />
                <ref bean="FeatureInfoMode" />
            </list>
        </property>
    </bean>
</property>
```

Figura 5.16: Configuraciones de la Barra de Herramientas

```xml
<bean name="ToolbarSeparator" class="org.geomajas.configuration.client.ClientToolInfo" />

<bean name="ZoomIn" class="org.geomajas.configuration.client.ClientToolInfo">
    <property name="parameters">
        <list>
            <bean class="org.geomajas.configuration.Parameter">
                <property name="name" value="delta" />
                <property name="value" value="2" />
            </bean>
        </list>
    </property>
</bean>
<bean name="ZoomOut" class="org.geomajas.configuration.client.ClientToolInfo">
    <property name="parameters">
        <list>
            <bean class="org.geomajas.configuration.Parameter">
                <property name="name" value="delta" />
                <property name="value" value=".5" />
            </bean>
        </list>
    </property>
</bean>
<bean name="ZoomPrevious" class="org.geomajas.configuration.client.ClientToolInfo" />

<bean name="ZoomNext" class="org.geomajas.configuration.client.ClientToolInfo" />

<bean name="ZoomToRectangleMode" class="org.geomajas.configuration.client.ClientToolInfo" />

<bean name="PanMode" class="org.geomajas.configuration.client.ClientToolInfo" />

<bean name="ZoomInMode" class="org.geomajas.configuration.client.ClientToolInfo" />

<bean name="ZoomOutMode" class="org.geomajas.configuration.client.ClientToolInfo" />

<bean name="ZoomToSelection" class="org.geomajas.configuration.client.ClientToolInfo" />

<bean name="PanToSelection" class="org.geomajas.configuration.client.ClientToolInfo" />

<bean name="DeselectAll" class="org.geomajas.configuration.client.ClientToolInfo" />

<bean name="MeasureDistanceMode" class="org.geomajas.configuration.client.ClientToolInfo" />
```

Figura 5.17: Configuraciones de cada uno de los parámetros de las Herramientas

```xml
<property name="layerTree">
    <bean class="org.geomajas.configuration.client.ClientLayerTreeInfo">
        <property name="tools">
            <list>
                <ref bean="LayerVisibleTool" />
                <ref bean="LayerLabeledTool" />
                <ref bean="ShowTableAction" />
                <ref bean="LayerRefreshAction" />
            </list>
        </property>
        <property name="treeNode">
            <bean class="org.geomajas.configuration.client.ClientLayerTreeNodeInfo">
                <property name="label" value="Layers" />
                <property name="layers">
                    <list>
                        <ref bean="clientLayerSACC"/>
                        <ref bean="clientLayerProv"/>
                        <ref bean="clientLayerDeptos"/>
                        <ref bean="clientLayerCuerposAgua"/>
                        <ref bean="clientLayerHIDRO"/>
                        <ref bean="clientLayerRutas"/>
                        <ref bean="clientLayerERDNWMS"/>
                        <ref bean="clientLayerERDNWFS"/>

                    </list>
                </property>
                <property name="expanded" value="true" />
            </bean>
        </property>
    </bean>
</property>
```

Figura 5.18: Configuraciones del árbol de capas

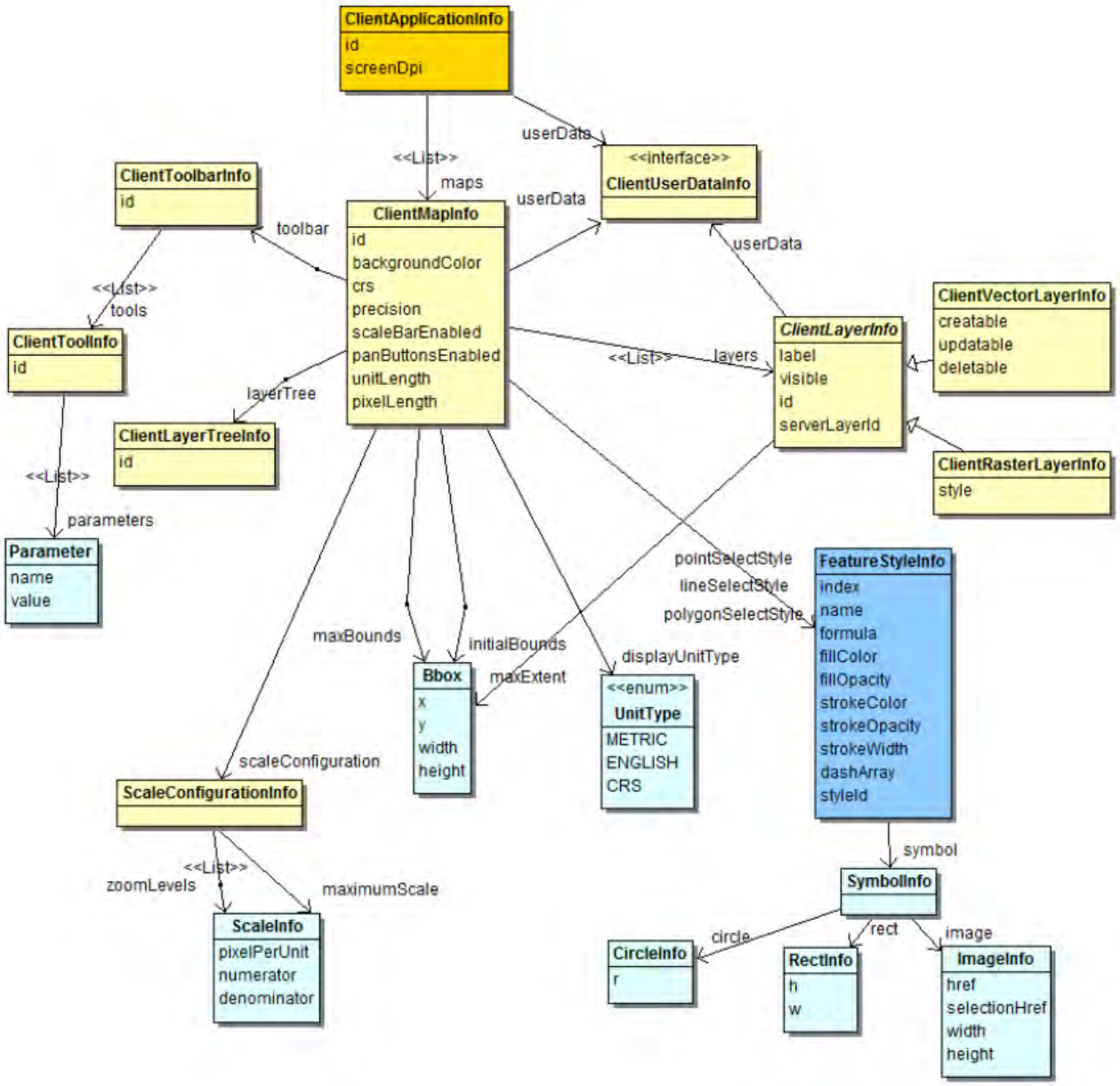

Figura 5.19: Resumen de los Objetos a Configurar en la unidad GIS Web Viewer

5.4.5. Interface GeoServer - Geomajas

La comunicación o interfase entre estas 2 unidades pertenecientes al subsistema VZ se realiza mediante el protocolo WMS y el protocolo WFS. De esta manera es posible concebir aplicaciones existentes como gvSIG, ArcMAP, Qgis, etc como un segundo tipo de cliente para acceder a los datos, aportando versatilidad al sistema. De esta manera se garantiza modularidad y flexibilidad. Además el tener datos geográficos en un servidor de mapas y solo visualizarlos usando la unidad GIS Web Viewer permite la centralización de estos, aspecto crítico teniendo en cuenta la sensibilidad en cuanto a difusión de los datos manejados por los SATs.

5.4.6. Funcionalidad Implementada en la Unidad GIS Web Viewer

COOMING SOON ACA IRIA LO DEL ACCESO A LAS CAPAS EN TIEMPO DE EJECUCIÓN Y LA RE-CONFIGURACIÓN DEL PROYECTO

Capítulo 6

Funcionalidad

A continuación se describe como los requerimientos del sistema, presentados en el capítulo de Requerimientos, son satisfechos por el diseño e implementación del sistema SAT/ERDNU. De esta manera será conveniente para el lector tener presente el capítulo requerimientos del este informe. La siguiente sección mapeará el código del requerimiento con una explicación de como el requerimiento es resuelto por el sistema implementado.

6.1. Funcionalidad del SAT/ERDNU

Con respecto al requerimiento CAE-HAP-FUN-L1-0001, el mismo es alcanzado logrando una correcta implementación de los procesos urbanos y nacionales en el subsistema AE. Además la escalabilidad del subsistema DT da soporte a la carga de datos necesarios para un sistema multiescala y multifactorial y por ultimo el modulo VZ da soporte a la visualización de los distintos tipos de datos permitiendo el análisis miltifactorial de los datos calculados. El requerimiento CAE-HAP-FUN-0002, y con el los requerimientos CAE-HAP-FUN-L1-0023, CAE-HAP-FUN-L1-0024 y CAE-HAP-FUN-L1-0025 serán satisfecho próximamente por el equipo de trabajo de salud del Instituto Gulich, pero vale la pena destacar que el sistema es completamente robusto, versátil y flexible para soportar una rápida implementación de los mismos. La tarea informática para soportar dicho requerimiento será simplemente crear una nueva unidad Translator para el ingreso y validación de datos provenientes del usuario y la programación de un nuevo proceso que calculará el riesgo a nivel urbano por localidad. Finalmente, será necesario modificar el archivo crontab para agregar una entrada que gestione la ejecución de dicho proceso. Permitiendo que cambios en las planillas sean implementados fácil y rápidamente en el sistema resuelve la capacidad de ingresar variables numéricas de variables de riesgo y permitiendo una planilla por localidad satisface el hecho de que esas variables numéricas sean ingresadas por cada localidad de la República Argentina (requerimiento CAE-HAP-FUN-L1-0003). El uso de GWT en la unidad uploader y de Geomajas en el GIS Web Viewer soluciona la implementación de manera directa y a través de configuraciones de la gestión de usuarios (CAE-HAP-DES-L1-0004). Tanto GeoServer como Geomajas soportan datos rasters, como vector logrando un sistema que resuelve el requerimiento CAE-HAP-FUN-L1-0006. Especial mención merece la pena el requerimiento CAE-HAP-PER-L1-0007, tal vez fue el principal requerimiento a satisfacer debido a la complejidad informática del mismo. Ajustar a una arquitectura existente completamente

funcional como la del CUSS de CONAE es una tarea compleja. Requiere la correcta modularidad e implementación de los distintos subsistemas y unidades que conforman el sistema y es por este requerimiento que se tomaron la mayor parte de las decisiones de diseño e implementación. De esta manera, y luego de numerosas revisiones todo la infraestructura desarrollada satisface los requerimientos arquitectónicos y de operatividad debido a su modularidad, flexibilidad y versatilidad. El requerimiento CAE-HAP-QUA-L1-0008 es satisfecho mediante las validaciones de los datos de entrada, mientras el CAE-HAP-QUA-L1-0010 es cumplido por la naturaleza GIS de la unidad GIS Web Viewer. El CAE-HAP-DES-L1-0014 es resuelto usando de la unidad GeoServer para centralizar los datos. La solución del requerimiento CAE-HAP-RES-L1-0015 recae sobre la plataforma usada para visualizar los datos: un sistema GIS como el logrado con la unidad GIS Web Viewer. El CAE-HAP-DES-L1-0017 es cumplido a través del almacenamiento interno de los datos en el subsistema SD. En un futuro estos archivos generados podrán estar disponibles, previa autorización, para ser pedidos al CUSS de CONAE. El CAE-HAP-FUN-L1-0018 es satisfecho a través del proceso de cálculo del riesgo de dengue a nivel nacional, mientras que el CAE-HAP-L1-FUN-0019 es satisfecho mediante la ejecución del proceso identidad, unidad xml2shp, existentes en el subsistema AE. Usando el protocolo WMS para la interconexión entre la unidad Geomajas y GeoServer se asegura el cumplimiento del requerimiento CAE-HAP-FUN-L1-0021 debido a que en realidad con WMS lo que se envía al usuario es una imagen de bits. La arquitectura del subsistema AE permite asegurar el cumplimiento del requerimientos CAE-HAP-DES-L1-0026, esto es, un cambio en alguno de los algoritmos impactara solo en la unidad que implementa el mismo y no a todo el sistema demostrando no solo flexibilidad sino también robustez. El requerimiento CAE-HAP-DES-L1-0027 aun no fue implementado pero fue tenido en cuenta en la realización de la arquitectura. De esta manera, además de resolver el requerimiento CAE-HAP-PER-L1-0007 permite la implementación directa del CAE-HAP-DES-L1-0027, a través de unidades ya existentes en el CUSS de CONAE. El CAE-HAP-QUA-L1-0030 es satisfecho mediante la unidad Manager de procesos estableciendo las fechas exactas en las que los procesos batch deben ser ejecutados. Así, el CAE-HAP-QUA-L1-0020, el CAE-HAP-QUA-L1-0031 y el CAE-HAP-QUA-L1-0032 son satisfechos de manera directa por la implementación de los algoritmos de cálculo. Los requerimientos que se encuentran en la tabla 2.2.1 no mencionados, pertenecen a un subconjunto de los aquí mencionados y son satisfechos de igual manera. En la figura 6.1 se puede observar como son satisfechos los requerimientos del usuario (QUE) por parte de la infraestructura informática obtenida (COMO). El subsistema DT (sistema de ingreso de datos) logra la automatización de la carga de datos, el control de fallos en la mismas y en menor medida, proporciona flexibilidad en la ejecución de los algoritmos ya que, mediante un cambio en la planilla, un algoritmo en particular puede ser modificado sin necesidad de una cambio en su implementación. El subsistema AE y las unidades que lo conforman definen el módulo de procesamiento de datos, mediante el cual se satisface la necesidad de integrar la información ambiental al modelo, beneficiando la correcta estratificación del riesgo de dengue. Información ambiental que es proporcionada a través de distintos sensores remotos (SAR y Ópticos) y que junto con la planilla de estratificación alimentan los modelos. De la misma manera el establecimiento de vectores como base del sistema de visualización produce un sistema integrado a multiescala que permite el análisis socio-económico a nivel tanto nacional como urbano y su relación con el riesgo de dengue presente. La encapsulación de los algoritmos us-

ando wrappers permite una mayor flexibilidad en sus inputs, lo que deviene en que la incorporación de inputs de cualquier tipo a los algoritmos sea posible. En particular incorporar imágenes SAR es factible y favorece la flexibilidad de los algoritmos (imágenes SAR muestran información tanto de día como de noche, en condiciones de nubes o sin ellas.) El servidor de mapas puede ser accedido a través de WMS o WFS independientemente de la plataforma Web GIS desarrollada. Esto es, el uso de herramientas GIS como GVSig, QGIS, GRASS es posible y de esta manera se consigue una mayor flexibilidad de uso. Pero el hecho de acceder a los datos mediante la plataforma de visualización WEB desarrollada garantiza el acceso desde cualquier plataforma con conexión a internet y un navegación web.

Requerimientos del Sistema COMO / Requerimientos del Usuario QUE?	Sistema de ingreso de datos	Modulo de procesamiento de datos	Base vectorial de datos	Imagenes Opticas LRes*	Imagenes Opticas HRes	Plataforma SIG (coordenadas geograficas)	SAR ImAges	Archivo de datos, metadatos y sub-productos (CUSS)	Basado en plataforma WEB	Acuerdo inter-institucional
Sistema Integrado Multiescala			9	9	9	9	3	3		
Automatizado /Usuarios y carga de datos	9		3					9	9	3
Control de fallos de carga datos y de coordenadas	9		9			9			3	
Incorporación datos ambientales, socio-económicos y vectoriales a nivel nacional/urbano		3	9	9	9	9	3	3		9
Visualización de riesgo a nivel nacional y urbano			3	3		9		3	9	
Cálculos e integración de información de amb y vectores		9	3		3	9		9	9	
Flexibilidad de algoritmos (activ. de estado de emergencia)	3	9			9		9	3	9	3

Figura 6.1: Lista de los requisitos de usuario principal y los requisitos del sistema que satisfacen los primeros. Proporciona una forma de analizar la posibilidad de dar una respuesta (o no) a cualquier exigencia del consumidor. Los números indican la posibilidad de respuesta.

6.1.1. GUI

El sistema obtenido facilita el ingreso de datos útiles a la hora del cálculo de los algoritmos y la ejecución de modelos mediante una interfaz de usuario amigable (figura 6.2). Mediante esta interfaz se informa al usuario acerca del éxito o fracaso en la carga. Además el sistema permite el manejo de usuarios con sus respectivos roles tanto para la carga de datos, visualización de los mismos, como así también su descarga. Con respecto a la visualización de los datos tanto de base, como los producidos por la ejecución de los modelos y algoritmos, pueden ser observados mediante la unidad Web GIS Viewer (figura 6.3). El usuario puede interactuar con los datos espaciales mediante herramientas base de cualquier GIS como lo son las de zoom, paneos y consultas geográficas (figura 6.4). Entre los datos

de base con los cuales el usuario puede interactuar se presentan: localidades, rutas, caminos, ríos y cuerpos de agua de la República Argentina. Entre los datos producidos por los modelos encontramos el vector y raster de la estratificación del Riesgo del Dengue a Nivel Nacional, el vector y raster representando el riesgo urbano, así como datos de imágenes satelitales (por ejemplo serie de Temperatura MODIS) utilizadas en el cálculo y el procesamiento de los modelos. Resumiendo podemos decir que cada cierto periodo de tiempo establecido en un archivo de configuraciones se ejecuta algoritmos tanto a nivel nacional como urbano para producir los mapas de riesgo de dengue estratificados. Esta funcionalidad es lograda mediante procesos batch desarrollados en Java y que son transparentes al usuario, es decir que el usuario del sistema solo interactúa con la visualización de los datos. Mientras que internamente el SAT/ERDNU maneja la ejecución de los modelos desarrollados por el equipo de trabajo (para mayor información sobre los algoritmos desarrollados ver [Porcasi and Lanfri, 2011]).

Figura 6.2: Sitio Receptor de Planillas: unidad Uploader

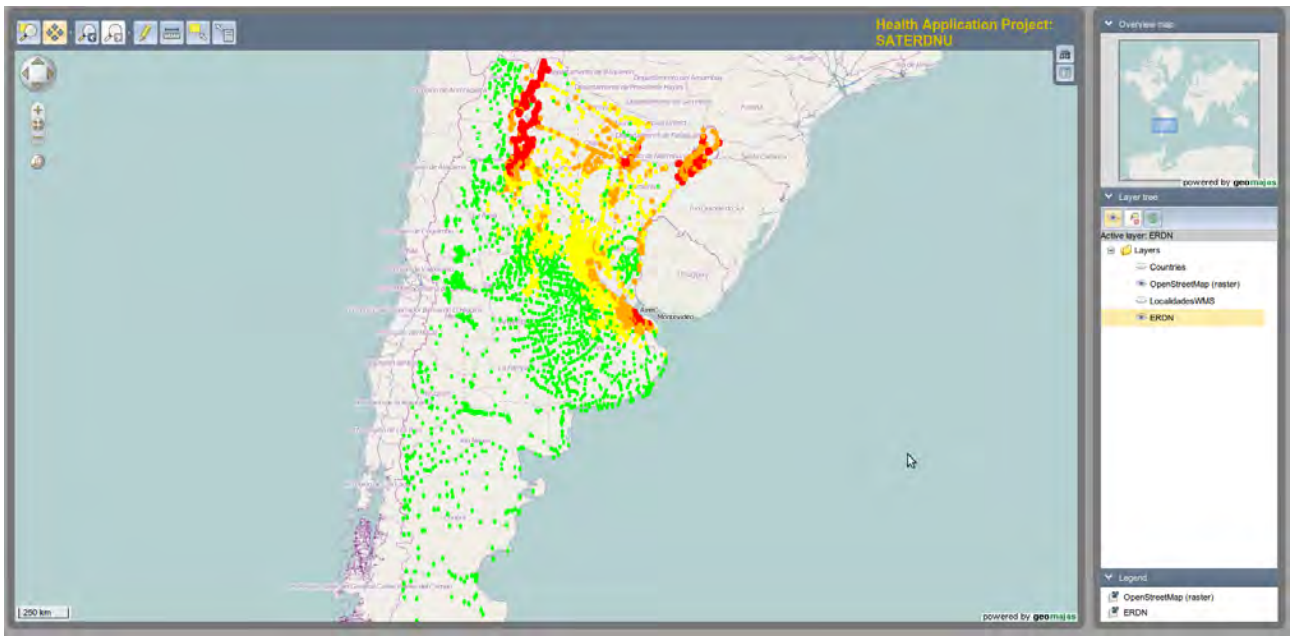

Figura 6.3: Sitio Visualizador: unidad Web GIS Viewer

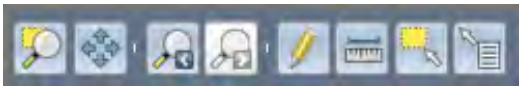

Figura 6.4: Herramientas del Visualizador Web

Capítulo 7

Conclusiones

Múltiples factores ambientales tanto de carácter biofísico como social constituyen una compleja trama que condiciona o determina la proliferación del vector-enfermedad del dengue. Factores bioclimáticos como demográficos y antrópicos actúan sobre las poblaciones de insectos, siendo las condiciones climáticas el factor regulador de la distribución espacio temporal de las poblaciones de artrópodos y su abundancia. A partir de la Epidemiología Panorámica, que tiene en cuenta dichas relaciones e interacciones entre los distintos elementos del ecosistema y la dinámica biológica tanto del hospedador como de la población del vector, se desarrolló un Sistema de Alerta Temprana que brinda el soporte tanto a la toma de decisiones como a la prevención de la enfermedad del Dengue en la República Argentina. El mismo se desarrollo en el marco de un convenio establecido entre el Instituto Gulich y el Ministerio de Salud de la Nación y consiste en una infraestructura informática de alerta tempranas en materia de prevención y control estratégico del Dengue: "Sistema de Alerta Temprana y Estratificación de Riesgo de Dengue Nacional y Urbano": SAT/ERDNU. La tecnología desarrollada para el ministerio de Salud de la Nación Argentina es una tecnología re-utilizable, flexible, mantenible y robusta. Y dada su gran modularidad y facilidad de configuración es integrada al CUSS de CONAE de manera directa (hecho corroborado a través de revisiones establecidas con los encargados del CUSS). Integrando herramientas Open Source (Spring, Geomajas, GeoServer, etc), se garantizó un desarrollo adecuado en tiempo y calidad, de bajo coste y reusable (el uso de OSS elimina las posibles restricciones de distribución que pudieran tener herramientas propietarias). El uso del frameworks ampliamente aceptados por la comunidad como Spring facilita el mantenimiento de la aplicación, el MVC y la inversión del control dan un diseño conocido a las unidades intervinientes, favoreciendo las posibles modificaciones (obviamente es mas fácil modificar sabiendo donde es necesario hacerlo). Otro punto clave que caracteriza la tecnología desarrollada es la portabilidad, que debido a la portabilidad de java, y al diseño realizado son satisfechas ampliamente. La tecnología es re-utilizable, puede ser usada en cualquier Sistema de Alerta Temprana donde el usuario alimente al sistema con datos de campo, el sistema procese dichos datos y finalmente exista la necesidad de visualizar los cálculos realizados a través de un Sistema de Información Geográfico espacial. Así esta infraestructura informática junto a sus componentes de software podrá ser re-utilizada por ejemplo en sistemas de prevención y control del chagas o malaria que involucren el estudio de la distribución de un vector-reservorio y sus relaciones ambientales, antropológicas, simplemente re-configurando el mismo. El sistema informático desarrollado se centró especialmente en robustez, flexibilidad y mantenibilidad

en el tiempo. Para su construcción se cumplió con los estándares para el desarrollo del software de la European Space Agency y con ello los de la Comisión Nacional de Actividades Espaciales (CONAE). A partir de ello, se le dio importancia a la estructura general del SAT/ERDNU, dejando en segundo plano funcionalidad extra que dadas las características logradas puede ser fácilmente implementada e integrada en un corto plazo. Al tratarse de un proyecto cuyo fin principal es su puesta en marcha en un sistema completamente operativo, gran parte del tiempo de desarrollo se invirtió en la investigación y análisis de tecnologías de punta que proporcionaran las bases para un correcto desarrollo. Aunque si bien el uso de dichas tecnologías es un hecho clave, la integración entre las mismas no es un tema trivial que se resolvió con éxito a lo largo de todo este trabajo, definiendo correctamente sus interfaces e interacciones. Spring, Web Services, Inversion of Control, el MVC, el GWT, los protocolos del OGC WMS y WFS, GeoServer, Geomajas, y Maven son las elecciones de tecnologías adoptadas en la implementación, lo que involucró, además de su estudio, el desechar otras opciones posibles luego de un criterioso análisis tecnológico. En este punto la modularidad lograda permite, entre otras cosas, reveer las elecciones tecnológicas de implementación en una unidad. Así, por ejemplo, la modularidad alcanzada en el subsistema VZ permite usar como framework principal en la construcción de la unidad GIS Web Viewer a otras tecnologías como OpenGeo, OpenTools u OpenMaps en lugar de Geomajas. E incluso da la posibilidad de usar programación pura sin un framework que soporte su desarrollo. Cabe destacar que, en general, proyectos Open Source también presentan cuestiones desfavorables al desarrollo a tener en cuenta a la hora de su utilización que quizás no estén presentes de manera tan marcada en herramientas o desarrollos propietarios. Una pobre documentación puede dificultar el tiempo de desarrollo complicando además la capacidad de reconocimiento de bugs presentes en alto porcentaje en tecnologías OSS. Preguntas como, la funcionalidad que quiero lograr no funciona porque estoy usando mal las cosas? es un bug de la tecnología no detectado?, es un bug detectado pero no informado? surgieron a lo largo de este desarrollo en numerosas oportunidades y demoraron el desarrollo e implementación. Otro punto que influye de manera directa en el desarrollo es la dificultad de integración de herramientas quizás agravado por la falta de documentación clave, o la dificultad de encontrar la misma. Sin embargo, como lo demuestra el presente trabajo, es posible desarrollar íntegramente una compleja aplicación operativa con herramientas Open Source innovadoras de alta calidad y bajo coste respondiendo a estándares de primer nivel como los de la Agencia Espacial Europea en el ámbito de la salud. Finalmente y después de una compleja interacción interdisciplinaria se logró una herramienta de fácil uso para el usuario y de fácil implementación en ambientes operativos que se corresponde a los objetivos y requerimientos inicialmente establecidos por el Ministerio de Salud de la Nación de Argentina, fácilmente escalable y mantenible que ayudará a la prevención y el control estratégico del Dengue en la República Argentina.

Referencias

Bejarano, J. (1979). Estudio sobre la fiebre amarilla selvática en la república argentina. *Subsecretaria de Medicina Sanitaria 38 pp*. 1

Booch, G., Rober, A. M., michael, W. E., and et al (2008). Object-oriented analysis and design with applications. *Peking: Post and Telecom Press 3rd ed*. 4

Carcavallo, R. and Martinez (1968). Entomoepidemiología de la república argentina. *Investigaciones Científicas de las Fuerzas Armadas Argentinas, Argentina, p 105-144*. 1

De La Torre, C. (2010). Cuss for sac-d architecture design document. *Instituto Gulich, CONAE*. 13, 39

Delgado, L. and Ramos, S. (2007). The remote sensing perspective to focus landscape ecology, the anthropogenic action and the malaria disease. *Space Technology for E-health Space technology-based tele-health project initiatives in Latin America and the Caribean*. 1

Dominguez, C. and Lagos, S. (2001). Presencia de aedes aegypti, diptera: Culicidae, en la provincia de mendoza, argentina. *Rev Soc Entomol Argent 60*. 1

Estrada, F. and Craig, G. (1995). Biología, relaciones con enfermedades y control de aedes albopictus. Technical report, Washington. OPS. 1

European Space Agency, . (2011). Bssc guides and reports. "http://www.esa.int/TEC/Software_engineering_and_standardisation/TECT5CUXBQE2.html". ESA. 5, 13

Gamma, E., Helm, R., ralph Johnson, and Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley. 20, 21, 23

Geomajas (2011). Web site geomajas. "http://www.geomajas.org/overview/about-geomajas". 28, 32

Geosparc nv, . (2010-2011). Geomajas user guide for developers. "http://files.geomajas.org/maven/trunk/geomajas/docbook-devuserguide/html/master.html". 44

Google (2011). Google web toolkit web site. "http://code.google.com/intl/es/webtoolkit/doc/1.6/overview.html". 29

Hales, S., De Wet, N., Maindonald, J., and Woodward, A. (2002). Potential effect of population and climate changes on global distribution of dengue fever: an empirical model. *Lancet*. 1

Jennifer, S. and Droll, D. (2001). A timeline for dengue in the americas to december 31, 2000, and noted first occurences. *Pan American Health Organization. Division of Disease Prevention and Control.* 1

Johnson, R. (2005). Introduction to the spring framework. TheServerSide.com. 21, 22

Johnson, R., Hoeller, J., Arendsen, A., Sampaleanu, C., Harrop, R., Risberg, T., Davison, D., Kopylenko, D., Pollack, M., Templier, T., Vervaet, E., Tung, P., Hale, B., Colyer, A., Lewis, J., Leau, C., and Evans, R. (2011). The spring framework, reference documentation. "http://static.springsource.org/spring/docs/2.0.x/reference/". v, 22, 23, 25

Kennedy, M. (2010). Evaluating open source software. *Defence AT and L.* 20

Kouri, G. (1998). El dengue, situación actual en las américas. *Instituto de Medicina Tropical Pedro Kouria - La Habana, Cuba.* 1

Micieli, M. V. and Campos, R. E. (July 2003). Oviposition activity and seasonal pattern of a population of aedes (stegomyia) aegypti (l.) (diptera, culicidae) in subtropical argentina. *Mem Inst Oswaldo Cruz, Río de Janeiro.* 1

Ministerio de Salud de la Nación, . (2009). Situación del dengue en la argentina. primer semestre de 2009. *Boletín Epidemiológico Periódico. Dirección de Epidemiología del Ministerio de Salud de la Nación.* 3

Ministerio de Salud de la Nación, . (2010a). Informe de vigilancia de dengue. *Parte De Prensa Nº23.* 3

Ministerio de Salud de la Nación, . (2010b). Informe de vigilancia de dengue. *Área de Vigilancia - Dirección De Epidemiología. Dirección De Enfermedades Transmitidas por Vectores.* 3

Morasca, S., Taibi, D., and Tosi, D. (2011). Towards certifying the testing process of open source software: new challenges or old methologies? *Proceeding FLOSS 0́9 Proceedings of the 2009 ICSE Workshop on Emerging Trends in FreeLibreOpen Source Software Research and Development*. 20

Oglietti, M. (2002). Domain independent planning for space: Building a bridge from both shores. *International Workshop on Planning and Scheduling for Space*. 13

Porcasi, X. (2009). *El Porceso de re-infestacion por T. infestans evaluado por sensores remotos*. PhD thesis, UNC. 3

Porcasi, X. (2011). Requeriment baseline document. *Instituto Gulich, CONAE.* 7, 8, 12

Porcasi, X. and Lanfri, S. (2011). Atbd. *Instituto Gulich, CONAE.* 58

Rossi, G., Lestani, E., and D'Oria, J. (2006). Nuevos registros y distribucion de mosquitos de la argentina. diptera: Culicidae. *Rev Soc Entomol Argent 65*. 1

Scavuzzo, C. M., Lamfri, M. A., Rotela, C., Porcasi, X., and Estallo, E. (2000). An overview of remote sensing and geodesy for epidemiology and public health applications. *Advances in Parasitology, 47*. 3

Scavuzzo, C. M., Lamfri, M. A., Rotela, C., Porcasi, X., and Estallo, E. (2006). Satellite image applied to epidemiology, the experience of the gulich institute in argentina. *E-Health. The International Trade Event and Conference fo eHealth, Telemedicine and Health ICT*. 3

Seijo, A. (2009). Dengue 2009: cronología de una epidemia.e. *Arch Argent Pediatr 2009*. 3

Shuang, L. and Chen, P. (2009). Developing java ee applicatrions based on utilizing design patterns. *WASE International Conference on Information Engineering*. 4, 20

Tauil, P. (2001). Urbanization and dengue ecology. *Cad Saude Publica*. 1

The Apache Software Foundation, . (2011). Maven web site. "http://maven.apache.org". Apache Maven Project. 31

Torkar, R., Minoves, P., and Garrigós, J. (2011). Adopting free,libre,open source software practices, techniques and methods for industrial use. *Journal of the Association for Information Systems JAIS*. 20

Ullah, K. and Khan, S. A. (2011). A review of issue analysis in open source software development. *Journal of Theoretical and Applied Technology*. 20, 31, 32

W3C Working Group, . (2004). Web services architecture. "http://www.w3.org/TR/ws-arch/". v, 23, 25

WHO (2009). World health organization. dengue: guidelines for diagnosis, treatment, prevention and control. *Geneva Congress*. 1, 3