Módulo geomático para la integración y el análisis espacio-temporal de casos en brotes epidémicos

Sistema HAP, SAT/ERDNU

Tesis de Maestría, septiembre 2013

Autor: *Juan José Fernández Garcete*
Directora: *Estefanía Aylén De Elía*
Codirectora: *Sofía Lanfri*

Módulo geomático para la integración y el análisis espacio-temporal de casos en brotes epidémicos

Por Juan José Fernández Garcete

Presentado ante la Facultad de Matemática, Astronomía y Física y el Instituto de Altos Estudios Espaciales Mario Gulich como parte de los requerimientos para la obtención del grado de

MAGISTER EN APLICACIONES ESPACIALES DE ALERTA Y RESPUESTA TEMPRANA A EMERGENCIAS

UNIVERSIDAD NACIONAL DE CÓRDOBA

septiembre, 2013

Estefanía Aylén De Elía, Directora del Trabajo de Tesis

Sofía Lanfri, Co-Directora del Trabajo de Tesis

Palabras claves: Módulo geomático, sistemas de información geográfica (SIG), aplicaciones web, geoweb, test de Knox, Knox, análisis espacio-temporal, HAP, SAT/ERDNU, GIAEST.

Resumen

El presente trabajo de tesis desarrolla la implementación de un módulo geomático con herramientas para análisis espacio-temporales de eventos epidémicos, en un entorno web. Se especifican los requerimientos del sistema, se diseña la arquitectura de software para la aplicación, así como también se presentan las interfaces del sistema. El mismo pretende contribuir a los sistemas de vigilancia de la salud y al control de enfermedades transmitidas por vectores, brindando herramientas de análisis espacio-temporales durante el transcurso de brotes epidémicos de este tipo de enfermedades. Está contemplado en el convenio marco entre el Ministerio de Salud de la Nación (MSAL) y la Comisión Nacional de Actividades Espaciales (CONAE), para la generación de sistemas con funcionalidades operativas que puedan hacer uso de herramientas geoespaciales, aplicadas al soporte de planes estratégicos de programas de salud en la Argentina.

Abstract

The present thesis develops a software module, implementing Geomatics tools for spatial and temporal analysis of epidemics events, on a web environment. There are specified the system requirements, the design of the software architecture and the system interfaces. It intends to contribute to the systems of health surveillance and control of vector-borne diseases, providing a space-time analysis tool during the course of the outbreaks. It is considered in the framework agreement between the Ministry of Health of the Nation (MSAL) and the National Commission on Space Activities (CONAE), to the generation of operating systems, using geospatial tools to support strategic planning in health programs of Argentina.

Contenido

Contenido

Lista de figuras

Lista de figuras

Lista de tablas

Lista de tablas

Capítulo 1

Introducción

1.1 Epidemiología

La epidemiología es el estudio de la distribución y los determinantes de la salud relacionados con estados o eventos en poblaciones específicas y la aplicación de este estudio para la prevención y control de problemas de salud (Last, 2001). Los epidemiólogos estudian no sólo la muerte, enfermedad y discapacidad, sino también los estados de salud más positivos y, sobre todo, los medios necesarios para mejorar la salud.

Esta disciplina se origina hace más de 2000 años atrás con Hipócrates, quien observaba que los factores ambientales influían en la aparición de una enfermedad. Pero sin embargo, no fue hasta el siglo *XIX* que se determinó en una mayor medida, la distribución de la enfermedad en determinados grupos de población humana. A partir de ese entonces se marcó no sólo los principios formales de la epidemiología, sino también algunos de los logros más importantes de la misma (Beaglehole and Bonita, 2004).

El descubrimiento de *John Snow*[1], en 1854, demostró que el riesgo de cólera en Londres se relacionó con el consumo de agua suministrada por una empresa en particular; para ello empleó la utilización de un mapa (Figura 1.1) donde relacionaba la agrupación de los casos con los canales hídricos de agua distribuidos en la ciudad. Los estudios epidemiológicos de *Snow* fueron un aspecto de una amplia serie de investigaciones que examinó la relación de los

[1] **John Snow** (1813 - 1858) Médico inglés precursor de la epidemiología, quien sentó las bases teórico-metodológicas de la misma.

Figura 1.1: Mapa de casos de cólera en el centro de Londres, septiembre 1854 (Snow, 1855).

procesos físicos, químicos, biológicos, sociológicos y políticos (Johansen et al., 2003).

La comparación de tasas de enfermedad en subgrupos de la población humana se convirtió en una práctica común a finales del siglo *XIX* y principios del *XX*. Este enfoque fue aplicado inicialmente a la lucha contra las enfermedades transmisibles, pero resultó ser una forma útil de vincular las condiciones ambientales o agentes a enfermedades específicas (Bonita et al., 2006). En la segunda mitad del siglo *XX*, estos métodos fueron aplicados a las enfermedades crónicas no transmisibles, como las enfermedades cardíacas y el cáncer, especialmente en los países de medianos y altos ingresos.

Generalmente varios factores contribuyen a la causalidad de las enfermedades. Algunos factores son esenciales para el desarrollo de una enfermedad y otros incrementan su riesgo de desenvolvimiento. Por esto, nuevos métodos epidemiológicos son necesarios para el análisis de estas relaciones causales (Bonita et al., 2006).

1.2 Brotes epidémicos

Epidemia se define como la manifestación de casos de una enfermedad, en una comunidad o región, con una frecuencia que exceda netamente la incidencia normal prevista. Para evitar el sensacionalismo que genera en la población se la hace referencia con los sinónimos de brote epidémico o brote (Contol and Prevention, 1992).

La incidencia es el número de casos nuevos de una enfermedad específica, diagnosticados o notificados en un lapso de tiempo, dividido el número de personas de una población determinada (Chin, 2001).

El número de casos que indica la existencia de una epidemia varía con el agente infeccioso[1], el tamaño y las características de la población expuesta, su experiencia previa o falta de exposición a la enfermedad, y el sitio y la época del año en que tiene lugar. Por consiguiente, se entiende por epidemicidad a la variable que guarda relación con la frecuencia esperada de la enfermedad en la misma zona, entre la población especificada y en la misma estación del año. La aparición de un solo caso de una enfermedad transmisible que durante un lapso prolongado no había afectado a una población, o que invade por primera vez una región en la que no había sido diagnosticada anteriormente, requiere la notificación inmediata y una investigación epidemiológica. La presentación de dos casos de una enfermedad de esa naturaleza en la que exista una relación de lugar y tiempo constituye una prueba suficiente de transmisión para que se la considere como epidémica (Chin, 2001).

En la década de los años '90 del siglo pasado, ha tenido lugar en el mundo la emergencia o re-emergencia de muchos eventos epidemiológicos. Dentro de los mismos se encuentra el descubrimiento de nuevas enfermedades infecciosas, sus agentes etiológicos y su fisiopatogenia. Del mismo modo resurgieron otras enfermedades que tuvieron determinados niveles de control en el pasado y ahora se muestran con incidencias cada vez más altas convirtiéndose en problemas sanitarios de gran importancia, tanto en los países en vías de desarrollo como en los desarrollados. Son un reflejo de la incesante lucha de los microorganismos por sobrevivir, buscando brechas en las barreras que protegen al ser humano contra la infección. Estas brechas sanitarias, que se han venido agrandando desde hace algunas décadas, pueden obedecer a comportamientos de alto riesgo como: fallas en los sistemas de vigilancia epidemiológica, control insuficiente de la población de insectos portadores de enfermedades, paralización de los sistemas de abastecimientos de agua y saneamiento, acercamiento a la fauna silvestre de los asentamientos humanos por la deforestación, entre otros (Suárez et al., 2000).

[1] **agente infeccioso** Microorganismo (virus, bacteria, hongo, protozoario o helminto) capaz de producir una infección o una enfermedad infecciosa.

A varias de estas enfermedades re-emergentes se las puede incluir dentro de un grupo conocido como enfermedades transmitidas por vectores, denominadas así por tener en común la necesidad de transmitirse por medio de algún organismo vivo portador del agente infeccioso. Las mismas se detallan en la sección a continuación.

1.3 Enfermedades transmitidas por vectores

Una enfermedad transmisible (o infecciosa) es causada por la transmisión de un agente patógeno específico a un huésped susceptible (Bonita et al., 2006). Los agentes infecciosos pueden transmitirse a los seres humanos ya sea: a) directamente, de otros seres humanos o animales infectados, o b) indirectamente, a través de vectores, partículas suspendidas en el aire o vehículos portadores.

Los vectores son insectos u otros animales que portan el agente infeccioso de persona a persona. Los vehículos son objetos contaminados o elementos del medio ambiente (como ropa, cubiertos, agua, comida, sangre, plasma, soluciones parenterales, o instrumentos quirúrgicos). Las enfermedades contagiosas son aquellas que se pueden propagar entre los humanos sin un vehículo o vector intermediario. Literalmente, contagiosa significa "a través del tacto". La malaria es por ejemplo una enfermedad transmisible pero no es contagiosa, mientras que el sarampión y la sífilis son transmisibles y contagiosas. Algunos agentes patógenos causan la enfermedad no sólo a través de la infección, sino también por el efecto tóxico de los compuestos químicos que éstos producen. Por ejemplo, *Staphylococcus aureus* es una bacteria que puede infectar a los humanos directamente, pero el envenenamiento alimenticio por estafilococo es causado por la ingestión de alimentos contaminados con una toxina que produce la bacteria (Bonita et al., 2006).

Así, las enfermedades transmitidas por vectores (ETV) son padecimientos relacionados con el saneamiento del ambiente doméstico y de los espacios cercanos a las comunidades, donde se reproducen o protegen los vectores y facilitan el contacto entre agentes y huéspedes; así mismo, otros procesos se dan por invasión de nichos silvestres o por migración de huéspedes como en la leishmaniasis o la encefalitis equina venezolana (de México, 2001).

La presencia de las ETV obedecen al acercamiento y contacto de vectores que reciben y transmiten agentes patógenos entre los humanos o desde otros animales a los humanos. Se han circunscrito en este concepto de ETV sólo aquellas enfermedades en que intervienen artrópodos, tales como mosquitos (Familia *Culicidae*), moscas (Familia *Simuliidae*, Subfamilia *Phlebotominae*), piojos (Familia *Pediculidae*), chinches (Familia *Reduviidae*), pulgas (Orden *Siphonaptera*) y garrapatas (Familia *Ixodidae*). Los agentes causales son parásitos (Géneros: *Plasmodium*, *Leishmania*, *Onchocerca* y *Trypanosoma*),

arbovirus (Familia *Flaviviridae*) y ricketsias (*Rickettsia rickettsii, R. prowazekii, R. typhi*) (de México, 2001).

La enfermedad de chagas, el paludismo, la leishmaniasis, la hidatidosis, las uncinariasis, la brucelosis, la fiebre hemorrágica argentina, el cólera, el dengue, la esquistosomiasis y otras, son ejemplos conspicuos de patologías regionales de causa infecciosa en donde intervienen vectores. La frecuencia de las enfermedades trasmitidas por vectores (ETV) están estrechamente vinculadas al deterioro y contaminación ambiental, la deforestación o la urbanización desorganizada, entre otras. Las ETV predominan en las zonas rurales, donde la pobreza atenta contra la calidad de la vivienda y se asocia a deficiencias en la provisión de servicios (Tolcachier, 2010).

En Argentina, las patologías de carácter regional de mayor interés para la salud ambiental son: el dengue, el paludismo, la hantavirosis, el chagas, la leishmaniasis, la leptospirosis y la fiebre hemorrágica argentina. Algunos datos de prevalencia[1] general son los publicados por el Boletín Epidemiológico Nacional, y corresponden a tasas de notificaciones. Sin embargo las tasas de prevalencia existentes podrían ser mas altas (Tolcachier, 2010).

1.3.1 Análisis de cluster para la enfermedad del dengue

El dengue es una infección vírica transmitida por la picadura del mosquito *Aedes aegypty*, se presenta en cuatro serologías de virus y sus síntomas aparecen luego de un periodo de incubación tras la picadura. Es una enfermedad similar a la gripe que afecta a lactantes, niños pequeños y adultos. En la actualidad más del 70% de la carga de morbilidad por esta enfermedad se concentra en Asia sur-oriental y en el pacífico occidental. En los últimos años, la incidencia y la gravedad de la enfermedad han aumentado rápidamente en Latinoamérica y el Caribe, así como en regiones de África y el Mediterráneo Oriental. Al aumento mundial del dengue han contribuido la urbanización, los movimientos rápidos de personas y bienes, las condiciones climáticas favorables y la falta de personal capacitado (OPS, 2012).

Un análisis de cluster, en donde previamente se reduce la posibilidad de azar, brinda mayor seguridad en la inferencia de los probables factores externos que causaron dicha agrupación. Esta información es bastante útil en el manejo de epidemias. Con los análisis de cluste se intenta fusionar la distribución espacial de los casos de dengue en los distintos tiempos que fueron apareciendo, de acuerdo a la necesidad de análisis requerido por los usuarios del sistema. El

[1] **Prevalencia** Es la frecuencia de casos existentes en una población definida para un punto dado en el tiempo. Estima la probabilidad que una población esté enferma por alguna enfermedad en el período de tiempo estudiado. Es útil en el estudio de carga de las enfermedades crónicas y su implicancia para los servicios de salud.

análisis de cluster es un método estadístico que clasifica una muestra de entidades (individuos o variables) en un número pequeño de grupos de forma que las observaciones pertenecientes a un grupo sean muy similares entre sí y muy disímiles del resto (Kaufman and Rousseeuw, 1990).

Algunos estudios han utilizado análisis de cluster o conglomerados para identificar espacial y temporalmente áreas de alto riesgo de transmisión de enfermedades en diversos lugares del mundo. Como antecedente podemos observar los análisis estadísticos basados en los métodos de Knox aplicados a la epidemiología en el brote de dengue desarrollado en Tartagal en el año 2004, región del noroeste argentino (Rotela et al., 2007).

1.4 Epidemiología en la Argentina

Es común utilizar jurisdicciones políticas nacionales para el estudio de las características epidemiológicas de un país para analizar los indicadores epidemiológicos de la población, pero esto no es posible replicarlo exactamente en Argentina. Cada uno de sus 24 estados, las 23 provincias y la Ciudad Autónoma de Buenos Aires (CABA), representa un país con población, geografía, economía y cultura muy diferentes al resto de sus estados hermanos. Por ello se opta por utilizar la tradicional división en regiones que determina grupos de provincias que comparten espacios vecinos, quedando delimitadas así cinco regiones como se muestra en el mapa (Figura 1.2). Esta forma de agrupación de las provincias argentinas obedece a una escala geográfica, pero en general coincide con características socio-económicas y ambientales similares en cada lugar (Barragán et al., 2007).

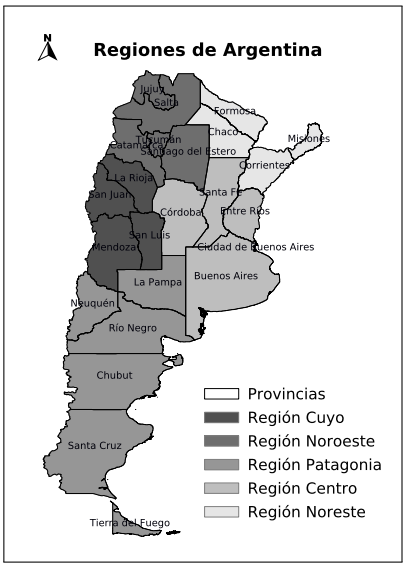


Figura 1.2: Regiones argentinas.

La forma más adecuada de analizar la situación de salud de una población es a través de los indicadores epidemiológicos. En general los indicadores de natalidad o de mortalidad (general, infantil y materna) y la esperanza de vida (Tabla 1.1, Tabla 1.2), todos ellos expresados en la pirámide poblacional, suelen dar una impresión aproximada del perfil epidemiológico de una población. Sin embargo la descripción de otros indicadores como los de morbilidad (Tabla 1.3,

Tabla 1.4) y de servicios es esencial para definir la situación de salud (Barragán et al., 2007).

Tabla 1.1: Indicadores básicos generales de Argentina

Indicador	Valor
Tasa bruta de natalidad[a]	18,7
Tasa bruta de mortalidad general[a]	7,9

[a] Año 2010 (por 1.000 habitantes). MS Serie 5 Nro 54/11.
- Fuente (MSAL and OPS-OMS, 2012)

Tabla 1.2: Indicadores básicos argentinos por regiones

Región	Natalidad[a]	Mortalidad[a]	EVN[b]
Centro	17,8	8,7	74,6
Cuyo	18,9	6,7	73,8
Noroeste	20,1	6,0	72,3
Noreste	21,1	6,2	71,4
Patagonia	20,6	5,8	73,9

[a] Tasa bruta, año 2010 (por 1.000 habitantes). MS Serie 5 Nro 54/11.
[b] Esperanza de vida al nacer en 2000-2001 (en años), ambos sexos. La esperanza de vida al nacer estima el número de años que en promedio puede esperar vivir un nacido vivo en una población y año determinado, si se mantuvieran constantes durante su vida las tasas de mortalidad específicas por edad que prevalecían al nacer.
- Fuente (MSAL and OPS-OMS, 2012)

1.4.1 Vigilancia epidemiológica

Un enfoque diferente a la mirada tradicional de la salud fue planteado por Marc Lalonde. Este se basa en el concepto del *campo de la salud* en el que se desglosan los determinantes de la salud en cuatro elementos: la biología humana, el ambiente, los estilos de vida y la organización de los servicios de salud (Lalonde, 1981). Encarar los problemas de salud desde este punto de vista facilita su comprensión y permite un análisis discriminado en el que cada factor tiene una significancia relativa. Integrados en el problema de salud, a los determinantes se les adjudica un valor relativo proporcional, de manera que se observa cómo contribuyen cada uno en el problema.

Debido a que el proceso de salud-enfermedad es dinámico y cambiante, los sistemas de salud se vieron ante la necesidad de desarrollar mecanismos sistemáticos de recolección de información. Con este objetivo surge el concepto de *vigilancia epidemiológica*, que luego se la ha denominado *vigilancia en salud pública* con el objeto de separarla de la esfera de la epidemiología para darle un

Tabla 1.3: Indicadores argentinos de morbilidad por regiones año 2010 (A)

Región	Leptospirosis	Hantavirus	Chagas[a]	SIDA	VIH
Centro	254	36	–	1.101	3.135
Cuyo	–	–	–	92	368
Noroeste	–	47	3	264	563
Noreste	19	2	2	68	280
Patagonia	4	10	–	102	349

[a] Chagas agudo vectorial.
[b] La información es provista por el Sistema Nacional de Vigilancia de la Salud, excepto SIDA, VIH y Tuberculosis, es parcial y está sujeta a modificaciones.
- Fuente (MSAL and OPS-OMS, 2012)

Tabla 1.4: Indicadores argentinos de morbilidad por regiones año 2010 (B)

Región	Tuberculosis	Meningitis[a]	Tétanos	Sífilis Congénita
Centro	7.377	7	7	441
Cuyo	286	–	–	49
Noroeste	946	1	2	69
Noreste	1.105	–	–	120
Patagonia	370	–	–	6

[a] Meningitis TBC en menores de 5 años.
[b] La información es provista por el Sistema Nacional de Vigilancia de la Salud, excepto SIDA, VIH y Tuberculosis, es parcial y está sujeta a modificaciones.
- Fuente (MSAL and OPS-OMS, 2012)

estatus independiente (Thacker and Berkelman, 1988). La vigilancia se refiere a la observación y recolección sistemática y permanente de datos de la ocurrencia y distribución de los sucesos de salud-enfermedad para su oportuno análisis en cuanto a determinantes, tendencias y otras informaciones útiles para su aplicación práctica en el campo de la salud pública. No es solamente la recolección de datos que surgen espontáneamente del campo, sino un sistematizado servicio de observación y comparación que activamente busca información sobre el dinámico campo de la salud. Se trata de información sobre enfermedades transmisibles y no transmisibles pero también sobre determinantes de la enfermedad y sistemas de salud (Lemus et al., 1996). Provee información y conocimientos para la detección oportuna de amenazas a la salud poblacional que surge de la identificación de: cambios en la ocurrencia de las enfermedades (fundamentalmente con situaciones de epidemias), conformación de conglomerados (grupos de riesgo aún sin situación epidémica) y cambios en las características del proceso (presentación, gravedad) (Denver, 1991).

La ley argentina 12.317 (1936) declaró la notificación obligatoria de casos comprobados o sospechosos de enfermedades contagiosas o transmisibles.

La nómina fue ampliada por normas complementarias hasta que en 1960 se promulgó la actualmente vigente ley 15.465, reglamentada por sucesivos decretos y normas. La normativa de 1994 (Res. MSAS 394/94) actualizó los procedimientos e instancias de notificación que se ordenan en el *sistema nacional de vigilancia epidemiológica* (SINAVE), el cual dio un nuevo impulso a la vigilancia epidemiológica reordenando y actualizando las enfermedades a notificar dentro del marco de la ley 15.465.

La introducción del concepto de riesgo dio pie a mecanismos descentralizados de procesamiento de la notificación de enfermedades facilitados por la computación. Los sistemas de notificación se fundan en el diagnóstico médico, al igual que la definición de muerte, presentando cuatro limitaciones a considerar:

(a) La oportunidad y precisión del diagnóstico.

(b) La notificación no tiene la misma fuerza que la definición de la causa de la muerte, requerida en los certificados de defunción. Ésta es su segunda limitación: el número de casos notificados siempre es menor al número real de casos incidentes y diagnosticados, con variación según el estigma social de la enfermedad y su gravedad. Así es frecuente que se notifiquen menos casos que los incidentes de *enfermedades de transmisión sexual* (ETS) como la sífilis y la blenorragia, por un estigma social, en vez que se notifiquen más de SIDA, por su gravedad. No obstante esta limitación, atendiendo a la inercia del hábito profesional en este aspecto, las comparaciones en el tiempo y en el espacio son posibles. Debe tenerse en cuenta, sin embargo, cualquier circunstancia que haga más notificable una enfermedad en un momento dado.

(c) Dada una masa de notificaciones el tercer factor limitante es su procesamiento no sólo correcto sino oportuno, su disponibilidad y publicidad.

(d) Por último, el mayor riesgo del diagnóstico epidemiológico, es que no se utilice para tomar decisiones (Barragán et al., 2007).

1.4.2 Sistema Nacional de Vigilancia Epidemiológica

El Sistema Nacional de Vigilancia Epidemiológica (SINAVE) es un programa nacional dependiente del Ministerio de Salud de la Nación, desarrollado para realizar las tareas de vigilancia epidemiológica a nivel nacional. Este programa es el responsable de la vigilancia epidemiológica y ha realizado su tarea con irregular efectividad. El Proyecto VIGI+A, auspiciado por Ministerio de Salud de la Nación, el Banco Mundial, el Programa de Naciones Unidas para el Desarrollo y la Organización Panamericana de la Salud, se ha implementado para fortalecer las acciones del SINAVE (Barragán et al., 2007).

Este sistema de vigilancia (SINAVE) está orientado a la detección de las modificaciones que se produzcan en el campo de la salud, abarcando datos

sobre ocurrencia y gravedad de las enfermedades pero también sobre factores ambientales, sistemas de salud y otros factores que influyen en la manifestación poblacional del fenómeno de enfermedad.

El SINAVE recibe información de todo el país y la organiza según dos categorías, un subsistema general y un subsistema específico. El *subsistema general* resume los datos globales y brutos de todas las enfermedades de notificación obligatoria en forma semanal. El *subsistema específico* corresponde al registro de cierto número de entidades particulares que se informan a través de un formulario específico según la enfermedad que se trate. Esta notificación diferencial se fundamenta en la potencial gravedad y difusión de las entidades. Además algunas de ellas no están contempladas en la Ley 15.465, escrita en 1960 ya que o no se conocían o no eran un problema de salud de relevancia. Las entidades de notificación específica son: "parálisis fláccida aguda, sarampión, meningitis, hantavirosis, cólera, dengue, tétanos neonatal, rabia, hepatitis, paludismo o malaria, brotes de enfermedades transmitidas por alimentos, infecciones hospitalarias, enfermedades respiratorias a través de unidades centinelas, diarreas virales y centros de referencia para enfermedades diarreicas". También se reciben los datos sobre fiebre hemorrágica argentina, enfermedad de chagas, tuberculosis, lepra, e infección por VIH-SIDA (de la Nación, 2007). La forma de notificación de estas entidades específicas varía según el caso.

Las actividades de la vigilancia epidemiológica son: recolección y notificación de casos; consolidación, procesamiento y análisis de los datos; formulación de recomendaciones; difusión de la información; supervisión; y evaluación del sistema. Estas actividades se desarrollan en forma diferente según el nivel jurisdiccional. Las jurisdicciones del SINAVE se organizan según tres niveles de acción: el nivel local; el nivel provincial y el nivel nacional (Barragán et al., 2007).

Nivel local

Se refiere a todo profesional de la salud que se encuentre en contacto directo con la población. Son los sujetos generadores del dato pero que además realizan la descripción epidemiológica inicial y una primera evaluación analítica. Son los encargados de detectar oportunamente cualquier eventual daño para la salud de alcance poblacional y de iniciar las primeras acciones de control para el caso específico. Estas acciones se efectúan sobre el caso y los contactos. Informan a través del formulario denominado "Informe epidemiológico semanal" y fichas específicas al nivel inmediato superior (nivel provincial).

Nivel provincial

En general las provincias constituyen una dirección provincial de epidemiología dependiente del Ministerio de Salud provincial. Dentro de la órbita

de esta dirección se consolida el nivel de acción provincial en cuanto a vigilancia epidemiológica. Sus acciones son a nivel individual y poblacional, coordinando las actividades de vigilancia, capacitando a los recursos humanos, realizando las investigaciones epidemiológicas que correspondan y procesando y analizando la información llegada del nivel local. En el análisis se integran todos los datos del caso: datos clínicos, de laboratorio y del brote. Este nivel debe dar la alerta y participar en la planificación para el control en casos de problemas con riesgo elevado ya sea por posibilidad de diseminación o por gravedad del proceso. Este nivel informa a la jurisdicción siguiente: el nivel nacional.

Nivel nacional

Se constituye en la Dirección de Epidemiología del Ministerio de Salud de la Nación. Sus acciones no se desarrollan sobre los individuos sino sobre la población. Sus principales actividades son de normatización; asesoramiento y cooperación científico-técnica, y consolidación de la información. El análisis de la información se realiza para el país como síntesis. Publica semanalmente el "Boletín Epidemiológico Nacional" para proveer información a las provincias como también informa a los organismos internacionales. Sus intervenciones son a nivel político-institucional a través de la legislación sanitaria.

1.5 Herramientas de análisis espacial

Una de las principales aplicaciones de la epidemiología es facilitar la identificación de áreas geográficas y grupos de población que presentan mayor riesgo de enfermar o de morir prematuramente y que por tanto requieren de mayor atención ya sea preventiva, curativa o de promoción de la salud. El reconocimiento de grupos de riesgo supone a su vez la selección de intervenciones sociales y sanitarias para disminuir o eliminar los factores específicos de riesgo. Esto implica una reorganización de los servicios de salud para dar respuesta a esas necesidades insatisfechas (Castillo-Salgado, 1993).

Actualmente, la limitación de recursos y el proceso de descentralización de los servicios de salud que ocurren en la mayoría de los países, exigen que los programas de salud sean más efectivos y eficientes en su toma de decisiones. Para ello, los programas de salud requieren de un sistema de información ágil que les permita identificar áreas y/o poblaciones con mayores necesidades insatisfechas de salud, de manera que les permita focalizar hacia esos grupos prioritarios sus intervenciones (WHO-PAHO/AIS, 2003).

El uso de mapas, particularmente si son digitales, es una herramienta útil para hacer más efectiva la toma de decisiones. Se ha estimado que cerca del

80% de las necesidades de información de quienes toman decisiones y definen políticas en los gobiernos locales, están relacionadas con una ubicación geográfica (Williams, 1987). Es en este contexto que los sistemas de información geográfica pueden ser considerados como una de las tecnologías existentes para facilitar los procesos de información y de toma de decisiones en los servicios de salud.

1.5.1 Sistemas de Información Geográfica

La información geográfica es un conocimiento adquirido a través del procesamiento de datos geográficamente referenciados. En un principio eran denominados como Sistemas de Información Geográfica pero actualmente por el alcance que tienen, hay autores que lo definen en un concepto más global como Servicios de Información Geográfica. Los servicios de información geográfica son (1) la funcionalidad proporcionada por una entidad de software a través de sus interfaces definidas como conjuntos nombrados de operaciones y, (2) la provisión de información generalizada de datos geoespaciales. El desarrollo de los servicios de información geográfica está estrechamente relacionado con objetos distribuidos de tecnología e internet (Shekhar and Xiong, 2008).

Esta ampliación del concepto de no limitarlo sólo a los sistemas, representa un paso importante para facilitar el acceso a la información geográfica y la tecnología de geo-procesamiento. Desde las perspectivas de desarrollo de software, los servicios de información geográfica representan un dominio vertical de servicios de información, donde muchos componentes de los servicios de Tecnologías de Información (TI) están combinados para formar aplicaciones de información. Los componentes de los Servicios de Información Geográfica (SIG) son autónomos, auto-descriptivos, y objetos reutilizables de software que pueden ser publicados, localizados, e invocados en espacios de direcciones múltiples y ambientes distribuidos (Shekhar and Xiong, 2008).

Los SIG permiten producir distintos tipos de mapas que dan lugar al análisis de la información representada en el mismo. Uno de ellos son los mapas de referencia, en donde se muestran los límites de ciertas áreas y se localizan diferentes objetos dentro de cada una, etiquetando usualmente cada objeto. Un ejemplo de este tipo de mapa son los mapas de rutas con varios tipos de carretera, fronteras municipales, distancias, poblados.

Además, se pueden obtener mapas temáticos que son aquellos en los que las áreas de un mapa se colorean o marcan de acuerdo a alguna clave, de manera que la naturaleza del color o marca reflejen la intensidad de alguna variable que se mapea. Entre este tipo de mapas se incluyen, entre otros: de área, que muestran un fenómeno de acuerdo a un territorio; de símbolos, que muestran objetos dispersos que están relacionados a puntos en el mapa; de

isolíneas, que muestran un fenómeno que tiene cambios muy uniformes en una difusión ininterrumpida; de densidad de puntos, los que muestran la ocurrencia de un fenómeno que se distribuye de manera no uniforme; de cartodiagrama, que muestran unidades territoriales con diagramas de magnitud de un fenómeno (Figura 1.3). Finalmente, algunos SIG tienen la capacidad de procesar imágenes, como en el caso de las fotografías aéreas o las imágenes de satélite, lo que implica que se pueden cubrir de manera continua y sistemática grandes extensiones geográficas con diferentes tipos de información, tales como precipitación, nubosidad, cobertura vegetal, tipo de suelos, erosión, etc (WHO-PAHO/AIS, 2003).

En la medida que fueron estableciéndose las tecnologías y convenciones para la distribución de los datos en internet, los SIG también hicieron su importante aparición. De esta manera se crea la posibilidad de compartir más fácilmente y difundir datos de este tipo, reduciendo así la cantidad de colección y creación de datos redundantes. La posibilidad de acceder a través de internet a Servicios de Información Geográfica también amplía el alcance de los mismos a las organizaciones y gobiernos que antes no tenían la capacidad, los recursos y/o conjuntos de habilidades para implementar capacidades SIG completamente. Por último, con el desarrollo de los SIG en Internet, se están desarrollando aplicaciones que se dirigen a los usuarios de SIG menos sofisticados con la función de ampliar la conciencia, proporcionar aplicaciones prácticas que se atraen en masa y proporcionar beneficios útiles para las actividades humanas cotidianas (Peng and Tsou, 2003).

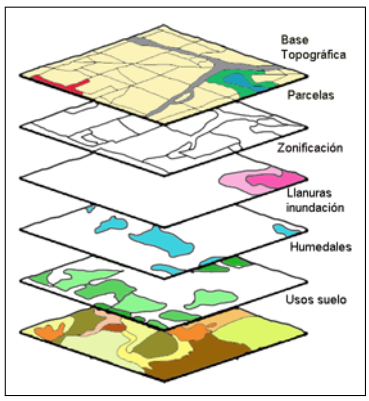


Figura 1.3: Capas de sistemas de información geográfica.

1.5.2 SIG en Salud Pública

Si bien el desarrollo de estos sistemas ha tenido sus raíces en otras áreas tales como el mercadeo, el transporte, la seguridad pública y, desde luego, en el monitoreo de fenómenos geológicos y climáticos de la tierra, los SIG pueden aplicarse en epidemiología para diferentes aspectos, la mayoría de ellos conectados entre sí. Entre algunos de los usos más comunes se tienen: la determinación de la situación de salud en un área, la generación y análisis de hipótesis de investigación, la identificación de grupos de alto riesgo a la salud, la planificación y programación de actividades, y el monitoreo y la evaluación de intervenciones.

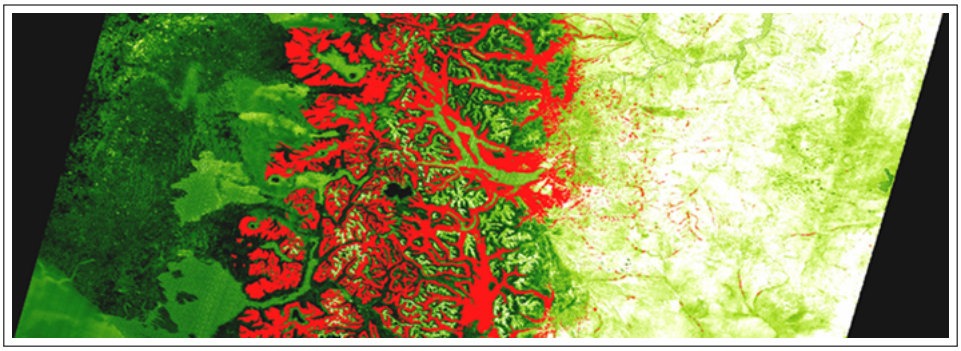


Figura 1.4: Mapa predictivo de distribución del roedor *Oligoryzomys longicaudatus* en la región norte de la Patagonia. En rojo se representan las zonas de presencia del reservorio superpuestas a una imagen de NDVI (*Normalized Difference Vegetation Index*) de SAC-C (resolución espacial: 175 m) en escala del blanco al verde (Porcasi et al., 2005).

Los SIG pueden utilizarse para determinar patrones o diferencias de situación de salud ante perspectivas de agregación particulares, que van desde el nivel continental, pasando por el regional, nacional, departamental o distrital, hasta un nivel local (WHO-PAHO/AIS, 2003).

Representan una poderosa herramienta que apoya el análisis de la situación de salud, la investigación operacional y la vigilancia para la prevención y el control de problemas de salud. Así mismo, estos sistemas proveen el apoyo analítico para la planificación, programación y evaluación de actividades e intervenciones del sector salud. Por ello, los SIG pueden considerarse parte de los sistemas de apoyo a decisiones para quienes formulan y siguen políticas en salud, permitiendo además un fortalecimiento en la capacidad de gestión de los servicios de salud (WHO-PAHO/AIS, 2003).

En el campo del uso de imágenes de satélite en epidemiología, conocido también como epidemiología panorámica o del paisaje (del inglés *landscape epidemiology*), encontramos que pueden utilizarse dentro de un enfoque holístico. En el cual se toma en cuenta las relaciones e interacciones de los diferentes elementos de un ecosistema, bajo la premisa que la dinámica biológica, tanto de los reservorios como de la población de vectores, está afectada por elementos del paisaje como la temperatura y la vegetación (Figura 1.4). Podemos entender a la epidemiología panorámica como una derivación de las aplicaciones de datos de sensores remotos, donde el objeto (el vector o el reservorio de una enfermedad) no puede ser detectado directamente en la imagen de satélite, pero que permite la caracterización de áreas eco-geográficas donde éste se puede desarrollar (Scavuzzo et al., 2006).

Es un campo relativamente nuevo en el área de las ciencias, pero con importantes trabajos de investigación que se desarrollaron a partir de la década pasada. Podemos citar algunos de los estudios que se fueron desarrollando en

el área como: correlaciones ambientales y patrones espaciales de infestación por el vector de la enfermedad de Chagas, el *Triatoma infestans*, en una zona rural de la provincia de Córdoba, Argentina (McGwire et al., 2006); detección de posibles vínculos entre signos climáticos y epidemias, estudio de variabilidad espacial-temporal con índices de vegetación y precipitaciones en zonas australes de sudamérica (Tourre et al., 2008); mapeos de ambientes susceptibles para la distribución del virus de la encefalitis de San Luis, basado en un modelo de árbol de decisión con datos ambientales obtenidos por sensores remotos (Rotela et al., 2011); y más recientemente el desarrollo de operativos sistemas de estratificación de riesgo para el dengue en Argentina, basado también en tecnologías geoespaciales (Porcasi et al., 2012)

1.5.3 SIG utilizados en epidemiología

Las aplicaciones de software en SIG utilizadas en el área de la salud se iniciaron con versiones de escritorio y fueron ampliándose hasta nuestros días donde se puede aprovechar todo el potencial que ofrecen los sistemas en redes con apoyo sustentable de bases de datos relacionales.

La Organización Panamericana de la Salud ha promovido desde sus inicios el uso de estas herramientas para un mejor apoyo en el análisis espacial de problemáticas de salud. Existen software de escritorio de distribución gratuita como el HealthMapper de la Organización Mundial de la Salud (WHO), el EpiMap del Centro de Control y Prevención de Enfermedades de Estados Unidos (CDC), o el SatScan del Instituto Nacional de Cáncer del mismo país (NCI); así también los organismos privados por su parte fueron lanzando software comerciales como el BoundarySeer, el SpaceStat y el ClusterSeer a través de las empresas BioMedware/TerraSeer, o el EpiAnalist y el ResearchAnalist de la empresa Laboratorios de Investigación en Salud Pública (PHRL) (Figura 1.5).

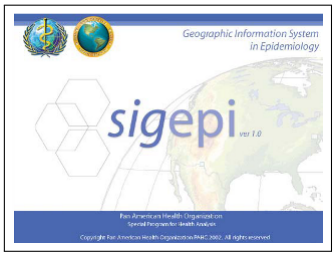


Figura 1.5: Software SIG para aplicaciones en epidemiología (WHO-PAHO/AIS, 2003).

Estos software, en su mayoría, complementan paquetes estadísticos con énfasis en metodologías orientadas a la epidemiología. Con ello se busca sacar mejor provecho al análisis de la información epidemiológica, tratando de contribuir desde ese enfoque a la toma de decisiones.

1.6 Análisis Espacio-Temporal

Principalmente en la epidemiología se analizan las variables de incidencia y prevalencia en pequeñas áreas geográficas homogéneas, especialmente en términos de intervalos de tiempos, dejando a veces de lado los eventos que ocurren a distancias cercanas con áreas un poco heterogéneas para evitar sesgos. Viendo desde el punto de vista de donde se originó el evento. No obstante, en algunas ocasiones y para ciertos análisis de investigativos, el sesgo del factor aleatorio no es tomado en cuenta. Lo importante es tratar de identificar todos aquellos factores que de alguna manera ejercen alguna influencia en la ocurrencia de los eventos epidemiológicos.

En la epidemiología, tanto el lugar como el momento, aportan considerablemente en los análisis que describen los factores por los cuales están influenciados los eventos. Si tenemos en cuenta todos los pares posibles de eventos ocurridos, podemos especificar a cada par, tanto en intervalos de tiempo como en intervalos de distancia. Siendo esto así, podemos probar una interacción entre estas dos variables.

Knox (1964) propuso un método de análisis bidimensional entre las distancias espaciales y temporales de eventos epidémicos. El procedimiento del test Knox consiste en calcular las distancias espaciales y temporales entre todos los pares de casos. Además, a partir de los valores críticos establecidos en el procedimiento, define dos variables dicotómicas que expresan si, un par de casos, están o no próximos en el espacio y en el tiempo. Para calcular la distancia espacial entre los casos $(x_i, y_i\ )$ y $(\ x_j, y_j\ )$ se utiliza la distancia Euclídea. Por su parte, la distancia temporal es la diferencia, en valor absoluto, entre las fechas de ocurrencia de cada caso.

El test Knox es una simple comparación de la relación entre los casos en términos de distancia y tiempo (Knox, 1964), que busca la detección de conglomerados en tiempo y en espacio. Cada par individual de casos es comparado en términos de intervalos de distancia y en término de intervalos de tiempo. La distancia entre puntos es dividida en dos grupos: cercanos y no cercanos. El intervalo de tiempo también es dividido en dos grupos: cercanos y no cercanos. Se forma una tabla de contingencia de $2x2$ mediante la clasificación de pares de casos como: cercanos en espacio y tiempo, cercanos en espacio, cercanos en tiempo, no cercanos ni en tiempo ni en espacio, de acuerdo a predeterminadas *distancias críticas* relacionadas al fenómeno estudiado.

Knox (1964) definió el estadístico X como el número de pares observados cercanos tanto en espacio como en tiempo. Se usan los totales marginales para calcular el número esperado de pares cercanos y compara este número con una distribución Poisson (Barton and David, 1966; Knox, 1964). Este test ha sido criticado debido a la subjetividad y arbitrariedad en la selección de las

distancias críticas.

Baker (1996); Kulldorff and Hjalmars (1999); Mantel (1967) han propuesto diferentes test para analizar la interacción espacial y temporal. Barton and David (1966) proponen la utilización del método de Monte Carlo para aumentar la cantidad de escenarios con eventos producidos aleatoriamente, para así compararlos con el escenario real.

Desde la creación del método, se encontraron decenas de estudios diferentes que utilizan el método de Knox. Muchos de ellos relacionados a la leucemia y otros tipos de cáncer. A pesar de las críticas el test de Knox es un método atractivo, demuestra ser simple y directo para calcular la estadística de prueba, además que sólo requiere el conocimiento de los casos sin necesidad de controles posteriores para su ejecución.

En estudios más recientes, fue aplicado también a brotes de dengue en la Guyana Francesa en el año 2001 Tran et al. (2004) y en Tartagal, provincia de Salta en el año 2004 Rotela et al. (2007), arrojando resultados relevantes para el análisis de los factores que pudieron haber intervenido en esos eventos epidémicos.

1.6.1 El test Knox de Tran et al. (2004)

El test de Knox, que quiere hacer referencia este trabajo de tesis para el análisis espacio-temporal, proviene de la investigación realizada por los autores Tran et al. (2004) para un brote epidémico de dengue del año 2001 en una localidad de la Guyana Francesa. Este trabajo toma la idea del test original de Knox para representarla como un análisis de la variación del riesgo relativo, al variar las distancias espaciales y temporales en la ocurrencia de casos de dengue.

En Tran et al. (2004) se evalúa si el número de pares de casos que se encuentra a una distancia espacial y temporal fija es sustancialmente diferente al número de pares de casos esperados a estas distancias por azar (casos aleatoriamente distribuidos). La relación entre el número real de casos que se encuentran a una distancia espacial S, y a una distancia temporal T, y al número de pares de casos encontrados aleatoriamente, puede ser considerado como el riesgo relativo de ocurrencia de otro caso del evento, T días más tarde y a S metros de distancia del primer caso del evento (Tran et al., 2004). El gráfico resultante de este trabajo puede ser observado en la Figura 1.6.

1.7 Sistema HAP (CONAE-MSAL)

En el marco del convenio entre el Ministerio de Salud de la Nación (MSAL) y la Comisión Nacional de Actividades Espaciales (CONAE) se crea el sistema

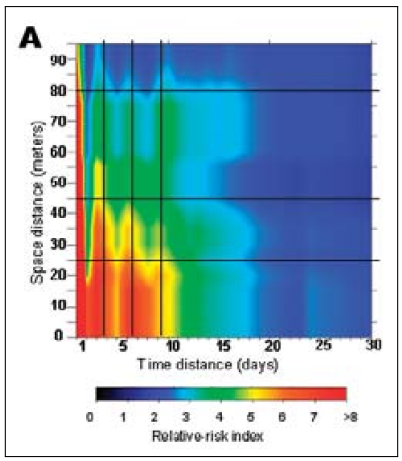


Figura 1.6: Gráfico del test de Knox que muestra el área principal de riesgo para el dengue (a menos de 100 metros y límites de 30 días), proveniente de datos de casos positivos por laboratorio (Tran et al., 2004).

Health Application Project(HAP). El objetivo general del proyecto es desarrollar el sistema operativo para apoyar el plan estratégico del programa de control vectorial a nivel nacional, utilizando las tecnologías geoespaciales y datos de sensores remotos. En ese sentido, el desarrollo de este sistema implica tres paquetes de trabajos principales: (1) Uno que esté relacionado con la plataforma computacional (hardware y software); (2) otra relacionada con el modelado biológico de riesgo basado en diferentes factores a múltiples escalas; y (3) la actividad relacionada con la disponibilidad operativa de los datos, de campo y de sensado remoto, para ejecutar los modelos espaciales estáticos (riesgo) y dinámicos (alerta temprana) (Scavuzzo and Torrusio, 2010).

Dentro de estas generalidades el HAP se constituye para desarrollar herramientas para el análisis de datos espaciales de diferentes fuentes tanto entomológicas, como ambientales. A su vez, debe desarrollar algoritmos específicos para estimación de mapas de riesgo; desarrollar herramientas específicas para evaluar el desempeño del sistema, herramientas de predicción de riesgo temporal y espacial; trabajar a diferentes escalas y posibilitar el ingreso de datos de sitios remotos, entre otros (Scavuzzo and Torrusio, 2010).

El Instituto de Altos Estudios Espaciales "Mario Gulich" perteneciente a CONAE y a la Universidad Nacional de Córdoba, tiene como principal objetivo el generar Sistemas de Alerta Temprana (SAT) en emergencias ambientales usando información espacial proveniente de distintos sensores remotos. Por lo tanto es necesario contar con la correcta infraestructura informática para dar soporte a este tipo de sistema. Dentro de los pilares del HAP se desarrolla el módulo informático 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 reutilizable para SATs de cualquier tipo gracias al uso de un correcto diseño y estructuras de código abierto (Peralta, 2011).

Mientras que la incorporación de software de código abierto en el desarrollo es un punto clave en este proyecto y requiere un complejo análisis de los mismos, los patrones de diseño proveen una manera eficiente de crear software orientado a objetos mas flexible, elegante y reutilizable (Booch et al., 2008). Así mismo, se debe tener en cuenta que el crecimiento exponencial de las aplicaciones web ha dado lugar a un complejo conjunto de las mismas, donde el mantenimiento, la flexibilidad y la expansibilidad se ha vuelto extremadamente difícil, acentuando más la necesidad de una correcta elección en el diseño (Shuang and Cheng, 2009).

Dentro de la escala urbana del HAP, el módulo "estratificación del riesgo de dengue urbano" (ERDU), presenta una estratificación del riesgo de dengue a partir de imágenes satelitales, datos históricos de circulación viral, índices aédicos (IirAa, ovitrampas, descacharrado.) y cartografía de base de formato vectorial, donde se estiman anomalías y variaciones espaciales de variables de relevancia para la estratificación de la enfermedad dentro de cada localidad en un periodo previo a un posible brote epidémico (Peralta, 2011).

El dengue por sus propias características, es una enfermedad bastante dinámica donde intervienen múltiples factores interdisciplinarios. A pesar de los esfuerzos desarrollados para su monitoreo y prevención, siguen apareciendo brotes de dengue año tras año. En este marco, surge la necesidad de incorporar al HAP, un módulo de análisis de agrupaciones de casos de dengue para un mejor manejo, control y prevención de la expansión espacio-temporal de la enfermedad, a medida que se va desarrollando en una localidad determinada. En particular en el HAP fue implentado el análisis de cluster por el método de Knox.

1.8 Motivación

La identificación, administración y disponibilidad operativa de los datos de campo pertinentes a casos de enfermedades es una aplicación de alto valor agregado que facilita el estudio epidemiológico en la vigilancia sanitaria. Complementar los sistemas de alerta temprana con herramientas que faciliten el acceso a datos puntuales georreferenciados y el estudio sobre su comportamiento espacial es la motivación principal de nuestro trabajo.

Así también seguir estructurando los sistemas de alerta temprana con aplicaciones que faciliten el estudio epidemiológico en la vigilancia sanitaria, especialmente en la disponibilidad operativa de los datos de campo, donde se posibilite y administre el ingreso de datos georreferenciados pertinentes a casos

de enfermedades de importancia desde sitios remotos, y ejecutar modelos espaciales dinámicos, fomentan la realización de módulos de sistemas que permitan alcanzar dichas metas y objetivos del HAP.

En este contexto se desarrolló el módulo GIAEST (Módulo Geomático de Integración y Análisis Espacio-Temporal para casos de brotes epidémicos). Este complemento se integra al sistema SAT/ERDNU cumpliendo con las interfaces definidas; cubriendo la necesidad de administración de información georreferenciada en red y la implementación de un método de análisis espacio-temporal durante el trascurso de un brote epidémico.

1.9 Objetivos

1.9.1 Objetivo general

Este trabajo tiene por objetivo principal el desarrollo de un módulo geomático de acceso web para la integración y el análisis espacio-temporal de casos en brotes epidémicos, denominado por sus primeras siglas GIAEST.

1.9.2 Objetivos específicos

A lo largo de esta tesis, se abordan los siguientes objetivos específicos:

- Desarrollar una plataforma web con funcionalidades SIG, brindando un ambiente intuitivo para la interacción con el usuario a través de interfaces gráficas que faciliten la visualización de la información espacial y el manejo de datos.
- Definir el diseño del módulo del sistema y sus requerimientos.
- Especificar, construir y documentar la arquitectura, así como las interfaces del módulo del sistema, adaptando esta nueva herramienta a la plataforma del sistema HAP.
- Implementar una de las técnicas de análisis espacial utilizada para detectar agrupaciones espacio-temporales, específicamente el test de knox.
- Implementar cada una de las funcionalidades que cumplan con los requerimientos planteados según el objetivo específico primero. En particular, los siguientes:
  - Administrar los datos con bases de datos relacionales.
  - Manejar y gestionar distintos niveles de usuarios.

Capítulo 2

Requerimientos

La IEEE[1] define un requerimiento como "(1) Una condición de la capacidad necesaria para resolver un problema o alcanzar un objetivo por un usuario; (2) Una condición o una capacidad que debe ser cumplido por un sistema, para satisfacer un contrato, norma, especificación u otro documento establecido formalmente" (IEEE, 1998) Todo modelo de desarrollo precisa que sean especificados sus requerimientos. El objetivo de la actividad de Especificación de Requerimientos del Software (ERS) es describir lo que el software propuesto debe hacer sin describir cómo el software lo hará (Jalote, 2008).

2.1 Especificación de Requerimientos del Sistema (ERS)

La ERS nos permite acotar la brecha de comunicación entre los clientes y los desarrolladores de software, para que así tengan una visión compartida del software que se está construyendo. Los beneficios de confeccionar una buena ERS nos permiten:

- Establecer las bases para un acuerdo entre el cliente y el proveedor de lo que el producto de software ejecutará.
- Proporcionar una referencia para la validación del producto final.

[1] **IEEE** *Institute of Electrical and Electronics Engineers*. El instituto de ingenieros eléctricos y electrónicos es la asociación profesional más grande del mundo dedicada al avance de la innovación tecnológica y excelencia en beneficio de la humanidad. IEEE y sus miembros inspiran una comunidad global a través de sus publicaciones altamente citadas, conferencias, estándares tecnológicos, y actividades profesionales y educativas.

- Reducir los costos de desarrollo.
- Obtener un software de alta calidad.

Para obtener estos requerimientos se utiliza un proceso que consta normalmente de tres partes que son: "(1) el análisis de requerimientos o análisis del problema, (2) la especificación de los requerimientos, y (3) la validación de requerimientos". Cabe señalar que el proceso de requerimientos no es una secuencia lineal de estas tres actividades ya que hay un solapamiento y una retroalimentación considerables entre estas actividades (Figura 2.1).

2.1.1 Proceso de formulación de requerimientos

Durante el proceso de *análisis del problema* se intenta entender como se podría modelar el comportamiento del sistema, su limitaciones, sus entradas y sus salidas, entre otros. El objetivo básico de esta actividad es obtener un conocimiento profundo de lo que el software tiene que proporcionar. Los clientes y usuarios finales deben explicar al analista sobre sus trabajos, su entorno y sus necesidades en la manera de como lo perciben. Los documentos que describen el trabajo o la organización pueden ser provistos, además de salidas de los métodos existentes para la realización de las tareas.

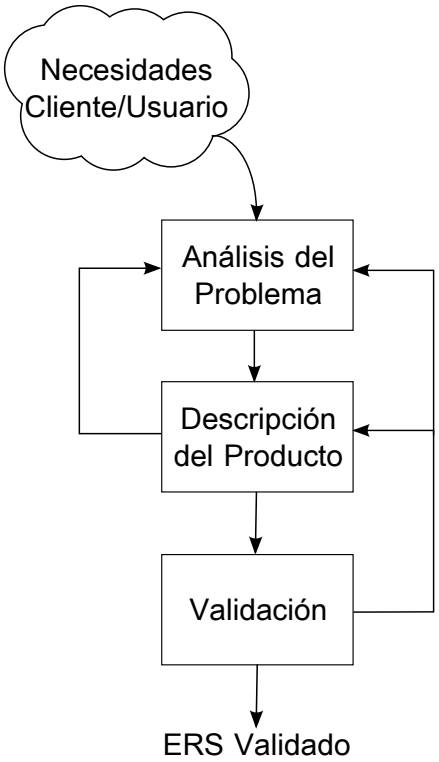


Figura 2.1: Esquema del proceso de formulación de requerimientos (Jalote, 2008).

En la siguiente fase se realiza la *especificación de los requerimientos*. El conocimiento obtenido por el análisis del problema constituye la base de la especificación de requerimientos, en la que el objetivo está en especificar claramente en un documento cuestiones como la representación, la especificación de lenguajes y herramientas.

Los *requerimientos de validación* se centran en garantizar que aquello que se ha especificado en la ERS son de hecho todos los requisitos del software,

además de asegurarse de que la ERS es de buena calidad. El proceso termina con los requisitos de la producción de la ERS validados (Jalote, 2008).

2.1.2 Características deseables de una ERS

Para satisfacer adecuadamente los objetivos básicos, una ERS debería tener ciertas propiedades y contener diferentes tipos de requerimientos. Algunas de las características deseables de una especificación de requerimientos de software son (IEEE, 1998):

1. Correcta
2. Completa
3. No ambigua
4. Verificable
5. Consistente
6. Ordenada por importancia y/o estabilidad

Una ERS es *correcta* si todos los requerimientos incluidos en la ERS representan algo necesario en el sistema final. Es *completo* si el software hace todo lo que debe hacer, y sus respuestas a todas las clases de datos de entrada se especifican en la ERS. Es *no ambigua* si y sólo si, cada requerimiento declarado tiene una y sólo una interpretación. Los requerimientos son a menudo escritos en lenguaje natural, que es inherentemente ambiguo. Si los requerimientos se especifican en un lenguaje natural, el que trascribe la ERS debe tener especial cuidado para asegurar que no haya ambigüedades.

Una ERS es *verificable* si y sólo si, cada requisito declarado es comprobable. Un requerimiento es comprobable si existe algún proceso de costo-beneficio que pueda comprobar si el software final cumple con ese requerimiento. Es *consistente* si no hay ningún requisito que esté en conflicto con otro. La terminología puede causar inconsistencias, por ejemplo, diferentes requisitos pueden utilizar diferentes términos para referirse al mismo objeto. Esto ocurre si la ERS contiene dos o más requisitos cuyas características, lógica o temporal, no pueden ser satisfechas juntas por cualquier sistema de software.

En general, todos los requerimientos de software no tienen la misma importancia. Algunos son críticos, otros son importantes, pero no críticos, y hay algunos que son deseables pero no son muy importantes. Una ERS está clasificada por *orden de importancia y/o estabilidad* si, para cada requisito se indican la importancia y la estabilidad de la exigencia. Estabilidad de un requisito refleja las posibilidades de cambio que pueda sufrir en el futuro. Se puede reflejar en términos del volumen de cambio esperado. Esta comprensión del valor que cada requisito proporciona, es esencial para el desarrollo iterativo (Jalote, 2008).

2.1.3 Componentes de una ERS

La integridad de las especificaciones es difícil de conseguir y más difícil aún verificar. Tener pautas sobre qué cosas diferentes una ERS deberá especificar, ayudará a identificar los requisitos completamente. Las cuestiones básicas que una ERS debe abordar son:

- Funcionalidad
- Desempeño
- Restricciones de diseño impuestas en una implementación
- Interfaces externas

Los requerimientos *funcionales* especifican el comportamiento esperado del sistema, donde las salidas deben ser producidas a partir de las entradas dadas. Ellos describen la relación entre la entrada y salida del sistema. Para cada requisito funcional, se debe especificar una descripción detallada de todas las entradas de datos y su fuente, las unidades de medida, y el rango de entradas válidas. El requerimiento funcional debe indicar claramente lo que el sistema debe hacer si se producen situaciones anormales. En concreto, se debe especificar el comportamiento del sistema para las entradas no válidas y salidas no válidas.

La parte del *desempeño* de los requerimientos de una ERS, especifica las restricciones de funcionamiento en el sistema de software. Todos los requisitos relativos a las características de funcionamiento del sistema, deben estar claramente especificados. Hay dos tipos de requisitos de desempeño: estáticos y dinámicos. Los estáticos son aquellos que no imponen restricción en las características de ejecución del sistema. Los de desempeño dinámicos en cambio, especifican restricciones en el comportamiento de ejecución del sistema, típicamente incluyen el tiempo de respuesta y limitaciones de rendimiento en el sistema. Todos estos requisitos deben expresarse en términos mensurables.

Hay una serie de factores en el entorno del cliente, que pueden restringir las opciones de un diseñador conduciendo a *restricciones de diseño*. Tales factores incluyen normas que se deben seguir, los límites de los recursos, el ambiente operativo, los requerimientos de seguridad y fiabilidad, y políticas que puedan tener un impacto en el diseño del sistema. Una ERS debería identificar y especificar todas las restricciones. Algunos ejemplos de estos son:

Cumplimiento de normas: Especifica los requisitos de las normas que el sistema debe seguir. Las normas pueden incluir el formato de informe y los procedimientos de contabilidad. Puede haber requisitos de auditoría que pueden requerir el registro de las operaciones.

Limitaciones de hardware: El software puede tener que operar en algún hardware existente o predeterminado, lo que impone restricciones en el diseño. Las limitaciones de hardware pueden incluir el tipo de máquinas que se utilizan, el sistema operativo disponible en el sistema, los idiomas compatibles, y los límites en el almacenamiento primario y secundario.

La fiabilidad y la tolerancia a fallos: Los requisitos de tolerancia a fallos puede colocar un obstáculo importante en la forma en que el sistema debe ser diseñado, ya que hacen del sistema más complejo y costoso. Los requisitos de recuperación son a menudo una parte integral aquí, detallando lo que el sistema debe hacer si se produce algún fallo para garantizar determinadas propiedades.

Seguridad: Los requisitos de seguridad son cada vez más importantes. Estos requisitos imponen restricciones sobre el uso de ciertos comandos, el control de acceso a los datos, el proporcionar diferentes tipos de requisitos de acceso para diferentes personas, el requerir el uso de contraseñas y técnicas de criptografía, y el mantener un registro de las actividades en el sistema. También pueden exigir una evaluación adecuada de las amenazas de seguridad, técnicas de programación adecuadas y el uso de herramientas para detectar defectos como desborde del búfer.

En la parte de la especificación de *interfaz externa*, todas las interacciones del software con la gente, el hardware, y los demás software deben estar claramente especificados. Para la interfaz de usuario, se deben especificar las características de cada interfaz del software. La interfaz de usuario es cada vez más importante y se debe dar una atención adecuada. Un manual preliminar debe realizarse con todos los comandos de usuario, formatos de pantalla, una explicación de cómo el sistema aparecerá al usuario, la retroalimentación y los mensajes de error. Al igual que otras especificaciones, estos requisitos deben ser precisos y verificables.

Para los requisitos de interfaz de hardware, la ERS debe especificar las características lógicas de cada interfaz entre el producto y los componentes de hardware. Si el software está apto para ejecutarse en el hardware existente o en el hardware predeterminado, se deben especificar todas las características del hardware, incluyendo restricciones de memoria. Además, se debe proporcionar el uso actual y las características de carga del hardware.

El requerimiento de interfaz debe especificar la interfaz que utilizará el sistema con otros software o los software que utilizarán el sistema. Esto incluye la interfaz con el sistema operativo y otras aplicaciones. Es necesario especificar el contenido del mensaje y el formato de cada interfaz.

2.2 Requerimientos del módulo desarrollado

Luego de haber visto estas generalidades que debería tener la documentación de una especificación de requerimientos de un sistema, se menciona que en un principio se definieron los requerimientos técnicos que necesitaba el sistema principal a partir de unos requisitos globales presentados por el MSAL en el convenio marco. Dichos requerimientos tienen su origen en el documeno *Requirements Baseline Document* (RBD), elaborado por los autores Porcasi, X. e Intrioni, M.V. para el sistema integrado de estratificación de riesgo de circulación viral de dengue a nivel nacional, cumpliendo las normas establecidas para el efecto (Porcasi and Intrioni, 2011; Porcasi et al., 2012).

Los requerimientos utilizados por el sistema principal fueron adoptados, de modo que el módulo geomático para la integración y el análisis espacio-temporal de casos en brotes epidémicos (GIAEST), reutiliza los mismos recursos y requerimientos de dicho sistema.

El Ministerio de Salud de la Nación (MSAL) precisa de un sistema que le permita administrar la información epidemiológica en conjunto con la geográfica a distintos niveles organizativos y en los diferentes puntos del país donde ocurren los eventos epidemiológicos. De esta manera los distintos niveles organizacionales, tanto en los locales y operativos donde suceden los eventos y se toman las primeras acciones, como también en los superiores donde se analiza la información para la toma de decisiones, se pueda obtener en el menor tiempo posible información precisa sobre el evento que se está desarrollando e ir desencadenando las respectivas acciones de control. De esta manera, se pretende obtener un análisis espacial y temporal del desarrollo del evento.

2.2.1 Funcionales

El sistema GIAEST deberá cumplir los siguientes requerimientos. Optimizando su uso y proporcionando facilidades a los usuarios.

Gestión de información. El sistema deberá permitir el ingreso, edición y eliminación de datos relacionados a casos de enfermedades.

Control de campos y validaciones. El ingreso de información al sistema debe presentarse de manera práctica, proveyendo el adecuado manejo de los datos, y evitando la posibilidad de error involuntario en el proceso de carga de la información.

- **Id.** El campo deberá ser numérico entero y generado automáticamente por el sistema al ingresar un nuevo registro. Es un campo obligatorio.
- **Nombre.** Deberá ser de tipo de dato cadena o caracter, de una longitud de hasta 30 caracteres. Es un campo optativo.

- **Fecha.** Deberá ser de tipo de dato fecha y restringido hasta la fecha actual en que se ingresa el registro. Deberá permitir el ingreso desde una lista desplegable o manualmente. Es un campo obligatorio.
- **Distrito.** Deberá ser de tipo de dato cadena o caracter, de una longitud de hasta 30 caracteres. Es un campo obligatorio. Será determinado automáticamente por el sistema a partir de la ubicación geográfica del registro ingresado.
- **Coordenada geográfica.** Será de tipo de dato coordenada geográfica y generado automáticamente por el sistema a partir de la ubicación geográfica del registro ingresado. Deberá ser flexible y permitir almacenar los datos en distintos sistemas de coordenadas (implementación controlada en base al motor de base de datos geoespacial). El mismo representará la coordenada del registro. Es un campo obligatorio.

Visualización de datos en formato raster y vector a diversas escalas. El sistema deberá ser capaz de soportar capas de información, raster o vectoriales, de otras variables relevantes para el tipo de estudio epidemiológico, tales como datos ambientales, socio-económicos y de infraestructura.

Consulta o filtro por atributos. El sistema deberá permitir la consulta a los atributos de los registros de las distintas capas almacenadas en el sistema. De esta manera actuará de manera de filtro, destacando en una grilla aquellos registros que cumplan las condiciones de consultas. Las consultas se efectuaran por medio de operadores lógicos desplegables de una lista.

- **Visualización de atributos seleccionados en grilla.** El sistema deberá presentar los datos y atributos asociados a cada caso en una grilla.
- **Visualización de casos por distintos atributos.** El sistema debe permitir visualizar la información filtrada por algún atributo perteneciente a los casos. Como por ejemplo el nombre de la ciudad o localidad donde ocurren, u otro atributo relevante para estudios epidemiológicos.

Información de atributos. El sistema permitirá la visualización de la información de los atributos, de manera individual, de cada registro almacenado.

Herramientas de navegación. El sistema deberá proveer las herramientas básicas de navegación geográfica en un mapa escalable, tales como el acercamiento, el alejamiento, y el desplazamiento en todas las direcciones.

Búsqueda de localidades. Para un rápido acceso a áreas geográficas específicas, el sistema deberá proveer una función de búsqueda por nombre de país, ciudad o localidad y dirección.

- **Formato de búsqueda.** El formato para ingresar los parámetros de búsqueda debe cumplir el siguiente orden: nombre de calle, nombre de ciudad o localidad, nombre de país. Cada campo debe estar

separado por el delimitador ",". Ejemplo: San Martín 433, Rosario, Argentina.

Ejecutar análisis espacio-temporal. El sistema implementará un análisis espacio-temporal de aglomeración de casos de enfermedades con la información almacenada en la base de datos, de acuerdo a los siguientes parámetros ingresados por el usuario:

- **Programa.** Este parámetro deberá contemplar los análisis para distintos programas de salud del MSAL, en primera instancia para el programa dengue, pero contemplar también a otros programas. Parámetro obligatorio. El parámetro será de tipo de dato cadena o caracter. Deberá poder ser seleccionado de un lista desplegable.
- **Ciudad.** Nombre de la ciudad en la cual se desea realizar el análisis. Parámetro obligatorio. El parámetro será de tipo de dato cadena o caracter. Deberá poder ser seleccionado de un lista desplegable.
- **Fecha desde.** Fecha de inicio del periodo a analizar. Parámetro obligatorio. El parámetro será de tipo de dato fecha. Será restringido, desde el inicio del primer caso registrado en el sistema hasta el día actual, y para este formato de datos. Deberá poder ser seleccionado de un lista desplegable.
- **Fecha hasta.** Fecha de fin del periodo a analizar. Parámetro obligatorio. El parámetro será de tipo de dato fecha. Será restringido, desde la fecha de inicio elegida anteriormente hasta el día actual, y para este formato de datos. Deberá poder ser seleccionado de un lista desplegable.
- **Réplicas.** Deberá permitir el ingreso de la cantidad de escenarios epidémicos aleatorios que debe generar el análisis para compararlo con el escenario epidémico real. Parámetro optativo. El parámetro será de tipo numérico entero. Será restringido solo para este formato de datos. Deberá poder ser seleccionado de un lista desplegable o ingresado manualmente.
- **Bins.** Valor numérico para indicar la resolución que tendrá el gráfico de salida. A mayor cantidad de bins, mayor resolución gráfica del resultado. Parámetro optativo. El parámetro será de tipo numérico entero. Será restringido solo para este formato de datos. Deberá poder ser seleccionado de un lista desplegable o ingresado manualmente.

Delimitación de fechas coherentes. El rango de fechas (parámetros de entrada del test de knox) debe estar delimitado por la fecha del primer caso registrado (límite inferior) y por la fecha del último caso registrado (límite superior) en una ciudad predeterminada.

Cambio de parámetros de los componentes del análisis. El algoritmo de análisis estadístico debe ser sensible al cambio/ajuste de los parámetros. El resultado de salida debe verse influenciado ante los cambios realizados en los datos de entrada.

Resultado del análisis. La ejecución del análisis deberá generar como salida un gráfico de dos dimensiones en la cual se indique la intensidad del riesgo, comparando las frecuencias de casos ocurridos en un lapso de tiempo contra los casos ocurridos en las diferentes distancias espaciales. Un histograma bidimensional del tiempo en relación al espacio.

Salida de información en formato PDF. Las distintas vistas de los mapas que se obtienen en la interfaz, deberán poder ser descargadas de la plataforma en un archivo físico en formato de documentación portable (PDF).

Impresión de mapas. El sistema deberá contemplar la impresión de capas por medio de dispositivos de impresión.

Aviso de ejecución del análisis espacio-temporal. Con un número de repeticiones mayor a 100, el sistema deberá desplegar un aviso en pantalla, informando que el tiempo de ejecución será superior a 10 segundos.

2.2.2 Desempeño

Además de ofrecer una amplia variedad de funciones, el módulo GIAEST deberá ejecutarse con un elevado desempeño, teniendo en cuenta la cantidad de solicitudes a nivel país como así también los tiempos de procesamiento de la información.

Almacenamiento de datos. El sistema contemplará el almacenamiento de datos en una base de datos geográfica.

Almacenamiento extra de capas. El sistema permitirá el almacenamiento extra de capas temporales externas, que el usuario utilizará durante una sesión.

Desempeño del ingreso de información. En el 99% de las operaciones, el ingreso de la información a la base de datos del servidor, deberá realizarse en un tiempo menor a 1 segundo en situaciones óptimas.

2.2.3 Restricciones de diseño

Dentro los niveles de dependencias del MSAL encontramos al local, provincial y nacional, como se mencionó en el capítulo anterior. El MSAL precisa replicar

estos niveles dentro del sistema para mantener las metodologías de trabajo ya establecidas. Así el sistema deberá permitir la existencia de diferentes perfiles de usuario con sus respectivos permisos para la visualización de la información y para el análisis espacio-temporal de los mismos.

Perfil visitante. El perfil de visitante sólo tiene acceso a la visualización de ciertos mapas generales y a algunas pocas funcionalidades de la interfaz gráfica.

Perfil local. El usuario con un perfil local, tiene acceso a las capas propias, además de las herramientas de la interfaz gráfica, dentro de los límites administrativos del organismo de salud. El mismo es el responsable del ingreso (Alta), eliminación (Baja) y modificación de la información correspondiente. Está más avocado a los organismos locales donde se presenta el evento. Tiene acceso a la base de datos de los programas de salud que corresponden a su límite administrativo.

Perfil provincial. El perfil provincial corresponde a los organismos con equipos de capacidad de análisis y organismos con poder de toma de decisiones. Tendrá acceso a la información provista por el usuario con perfil local y podrá realizar el análisis espacio-temporal del límite administrativo que le corresponde.

Perfil nacional. El perfil nacional corresponde más directamente al propio MSAL en su carácter de organismo rector y normativo. Donde obtiene información del usuario con perfil provincial pero a una escala mayor donde se observa el contexto general de todo el territorio del país.

Utilizar los datos de MAPEAR. El sistema deberá visualizar como capas de información geográfica los datos provistos por el proyecto de mapas electrónicos argentinos (MAPEAR).

Una especificación más detallada de las funcionalidades a las que pueden acceder los perfiles, se encuentra descripta en el apéndice B.1.

Capítulo 3

Arquitectura

Cualquier sistema complejo se compone de subsistemas que interactúan bajo el control del diseño general, de tal manera que el mismo proporciona el comportamiento esperado. Al diseñar tal sistema, por lo tanto, el enfoque lógico es identificar los subsistemas que deben componerlo, las interfaces de estos, y las reglas para la interacción entre ellos. Esto es lo que la arquitectura de software tiene como objetivo llevar a cabo.

La arquitectura de software es un área relativamente reciente. Como los sistemas de software se convierten en sistemas cada vez más distribuidos y más complejos, la arquitectura se convierte en un paso importante en su construcción.

Debido a una amplia gama de opciones disponibles actualmente de cómo un sistema puede estar configurado y conectado, un cuidadoso diseño de la arquitectura se vuelve muy importante. Es durante el diseño de la arquitectura donde se escogen opciones como el uso de algún tipo de software de enlace (*middleware*), algún tipo de base de datos interno del sistema (*back-end*), algún tipo de servidor, o algún tipo de componente de seguridad. La arquitectura es también el primer lugar donde las propiedades como la fiabilidad y el rendimiento pueden ser evaluados para el sistema, propiedades de la arquitectura que están siendo cada vez más importantes (Jalote, 2008).

En términos generales, la arquitectura de un sistema proporciona una visión de muy alto nivel de las partes del sistema y cómo éstas se relacionan para formar el sistema completo. Es decir, divide la arquitectura del sistema en partes lógicas de tal manera que cada parte pueda ser comprendida de forma independiente y luego describe el sistema en términos de estas partes y la relación

entre ellas.

Debido a la posibilidad de tener varias estructuras, una de las definiciones más aceptadas de arquitectura de software es la estructura o estructuras del sistema, que comprenden elementos de software, las propiedades exteriormente visibles de esos elementos, y las relaciones entre ellos (Bass et al., 2003).

Algunos usos importantes de las descripciones de la arquitectura de software son (Bass et al., 2003; Clements et al., 2010; IEEE, 2000):

Comprensión y comunicación. Una descripción de la arquitectura es principalmente para comunicar la arquitectura a sus diversas partes interesadas, que incluyen los usuarios que utilizarán el sistema, los clientes que encargaron el sistema, los constructores que construirán el sistema, y, por supuesto, los arquitectos. A través de esta descripción, los interesados obtienen una comprensión de algunas propiedades macro del sistema y la forma en que el sistema pretende cumplir con los requisitos funcionales y de calidad. Como la descripción proporciona un lenguaje común entre las partes interesadas, también se convierte en el vehículo para la negociación y el acuerdo entre las partes interesadas, que pueden tener objetivos contradictorios.

Reutilización. El mundo de la ingeniería de software, durante mucho tiempo, ha estado trabajando hacia una disciplina donde el software puede ser ensamblado a partir de piezas que son desarrolladas por personas diferentes y están disponibles para que otros utilicen. Si se quiere construir un producto de software en el que los componentes existentes se puedan reutilizar, entonces la arquitectura se convierte en el punto clave en el que se decide, a máximo nivel, que reutilizar. La arquitectura debe elegirse de tal manera que los componentes que deben ser reutilizados puedan ajustarse adecuadamente y en conjunto con componentes que pudieran ser desarrollados. La arquitectura también facilita la reutilización entre los productos que son similares y la construcción de familias de productos de tal manera que las partes comunes, de estos productos diferentes pero similares, pudieran ser reutilizados. La arquitectura ayuda a especificar lo que es fijo y lo que es variable en estos productos diferentes, y puede ayudar a minimizar el conjunto de elementos variables tales que los diferentes productos puedan compartir partes de software al máximo. Una vez más, es muy difícil de lograr este tipo de reutilización a un nivel de detalle.

Construcción y evolución. Como la arquitectura divide al sistema en partes, eso requiere que los diferentes equipos (o individuos), tengan posibilidad de trabajar por separado en cada una de ellas. Una división adecuada en la arquitectura, puede proporcionar al proyecto versatilidad en el ensamblado del sistema. Como casi por definición las partes especificadas en una

arquitectura son relativamente independientes (la dependencia entre las partes provienen a través de su relación), lo que posibilita su construcción en forma independiente.

Análisis. Es altamente deseable que algunas propiedades importantes sobre el comportamiento del sistema se puedan determinar antes de que el sistema sea construido. Esto permitirá a los diseñadores considerar alternativas y seleccionar la que mejor se adapte a las necesidades. Muchas disciplinas de ingeniería utilizan modelos para analizar el diseño de un producto por su coste, fiabilidad, rendimiento, etc. La arquitectura abre las posibilidades para el software. Es posible a partir de su arquitectura (aunque los métodos todavía no están completamente desarrollados o estandarizados) analizar o predecir las propiedades del sistema que está siendo construido.

3.1 Infraestructura informática

La Comisión Nacional de Actividades Espaciales desarrolla software bajo los estándares de la *European Space Agency* (ESA), quienes definen las prácticas para el desarrollo de software espacial y deben ser aplicadas en este ámbito (Agency, 2011). La especificación de requerimientos, el diseño detallado del sistema, y la definición de sus interfaces son etapas que deben existir en el proceso de desarrollo de un correcto sistema informático.

El presente capítulo tiene como objetivo presentar la estructura adoptada para este sistema, la cual fue planteada pensando principalmente en su reutilización, debido a la necesidad de contar con una herramienta robusta que nos permita reutilizar el sistema para cualquier tipo de SAT.

La estructura del sistema, comprende los elementos del software (subsistemas), sus propiedades externas visibles (interfaces) y las relaciones entre ellos (a través de unidades). La unidad es la mínima porción del sistema que puede ser desarrollado independientemente y es representada en nuestra estructura como un círculo. Mientras que un subsistema agrupa a un conjunto de unidades con un objetivo en común.

3.2 Arquitectura del módulo desarrollado

El módulo GIAEST se inserta en la arquitectura de software del sistema SAT/ERDNU. Este sistema se compone de cinco subsistemas que interactúan bajo el control de diseño del sistema, de tal manera que el sistema proporciona el comportamiento esperado. Los subsistemas que componen al SAT/ERDNU son: subsistema *Data Translation* (DT), subsistema *Spatial Data* (SD), subsistema

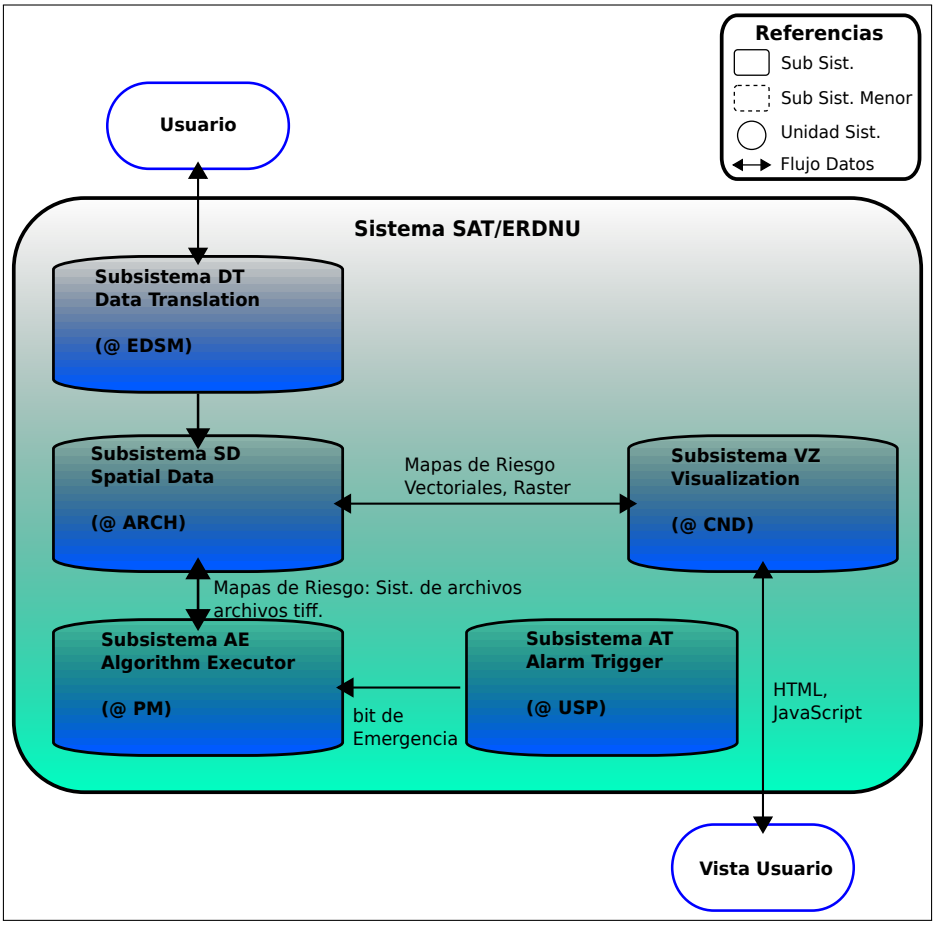


Figura 3.1: Esquema de la arquitectura general del sistema SAT/ERDNU

Algorithm Executor (AE), subsistema *Alarm Trigger* (AT) y subsistema *Visualization* (VZ) (Figura 3.1). De estos subsistemas entraremos a detallar aquellos que reutiliza la arquitectura del módulo desarrollado para adaptarse al sistema SAT/ERNU. Para mayor detalle de los módulos no mencionados en esta sección ver en (Peralta, 2011).

Subsistema SD (Spatial Data)

El subsistema SD, es el que gestiona el repositorio de datos. Es el encargado de almacenar los datos de salida del subsistema AE de manera óptima para luego ser utilizado por el subsistema VZ.

Subsistema AE (Algorithm Executor)

El subsistema AE, es el responsable de ejecutar los algoritmos para producir los mapas y gráficos de análisis espacio-temporal (un algoritmo por tipo de análisis), para el período de tiempo especificado en los parámetros. Los datos de entrada de este subsistema son provistos por los usuarios u

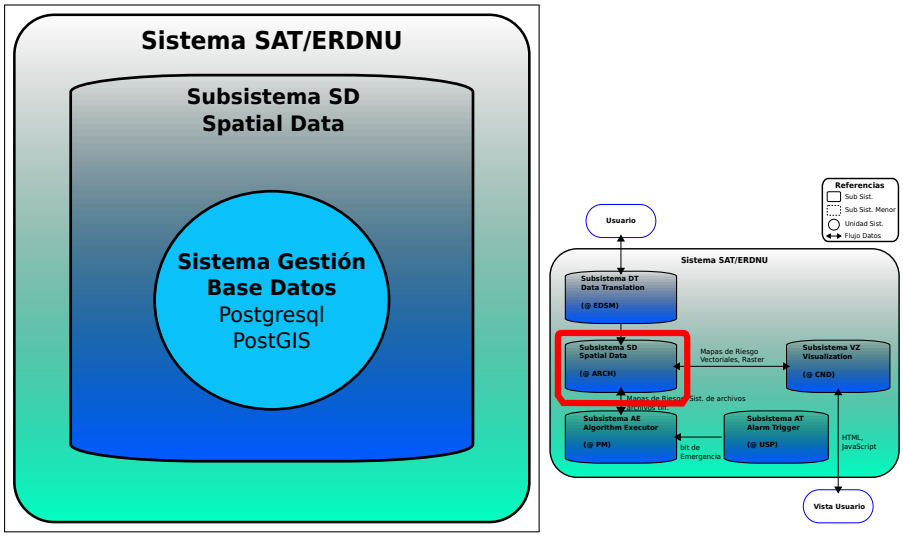


Figura 3.2: Esquema del subsistema SD y su ubicación en el SAT/ERDNU

obtenidos de los demás subsistemas.

Subsistema VZ (Visualization)

El subsistema VZ, es el encargado de mostrar los datos, resultados de análisis y estadísticas de manera amigable al usuario, a través de la web. Este subsistema implementará la descarga de la vista instantánea de los mapas disponibles, a la hora de la visualización. Además, mediante este subsistema, se permitirá la impresión de mapas.

3.2.1 Subsistema Spatial Data

El subsistema SD es el encargado, en primera instancia, de almacenar los datos ingresados por el usuario. Los cuales son necesarios para la ejecución de los algoritmos. Además de las unidades que ya forman parte de la arquitectura del sistema HAP, se agregará una nueva funcionalidad que almacene los resultados de la ejecución de los análisis espacio-temporales en formato PNG para su posterior visualización. Es así que el subsistema SD estará compuesto por una unidad que se encargará de copiar los diferentes archivos en las carpetas de almacenamiento. Dicha unidad manejará conflictos de accesos y atomicidad de operaciones (Figura 3.2).

3.2.2 Subsistema Algorithm Executor

El subsistema AE del SAT/ERDNU fue diseñado de manera robusta para soportar la inclusión de nuevos procesos con sus respectivas unidades, administrados por la unidad del *Algorithm Execution Manager*. Aprovechado está funcionali-

dad los procesos del GIAEST se adhieren perfectamente a esta arquitectura. La Figura 3.3 muestra el esquema de dicho subsistema. Las unidades que conforman el 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, tanto de riesgo como de análisis, producidos por cada algoritmo respectivamente.

Proceso 1, Proceso 2, ..., Proceso 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 es la encargada de la interfaz entre el archivo XML y los datos necesarios para la ejecución de cada algoritmo correspondiente al sistema SAT/ERDNU (Peralta, 2011).

Otra unidad adherida fue la *JDBC4*, la unidad que utiliza la librería de Java para interactuar con el subsistema SD, con el objetivo de obtener y depositar registros en la base de datos para los análisis espacio-temporales.

Así como también la unidad *RParser*, que es la unidad encargada de interactuar con el sistema estadístico R, sirviendo de interfaz para el envío correcto de los parámetros necesarios para la ejecución del algoritmo de análisis (Figura 3.5).

3.2.3 Subsistema Visualization

El subsistema está conformado por las siguientes unidades:

GeoServer. Esta unidad sirve los diferentes datos geográficos en distintos formatos, entre ellos los mapas de riesgo del proyecto HAP generados por los algoritmos o las capas propias del usuario.

GIS Web Viewer. Esta unidad muestra los datos servidos por la unidad GeoServer, de manera amigable interactua con el usuario autorizado. Esta funcionalidad es alcanzada mediante la implementación de un sistema de información geográfica. Además esta interfaz permite la descarga de las capas visualizadas y permite la impresión de mapas. El mismo está representado en la Figura 3.4.

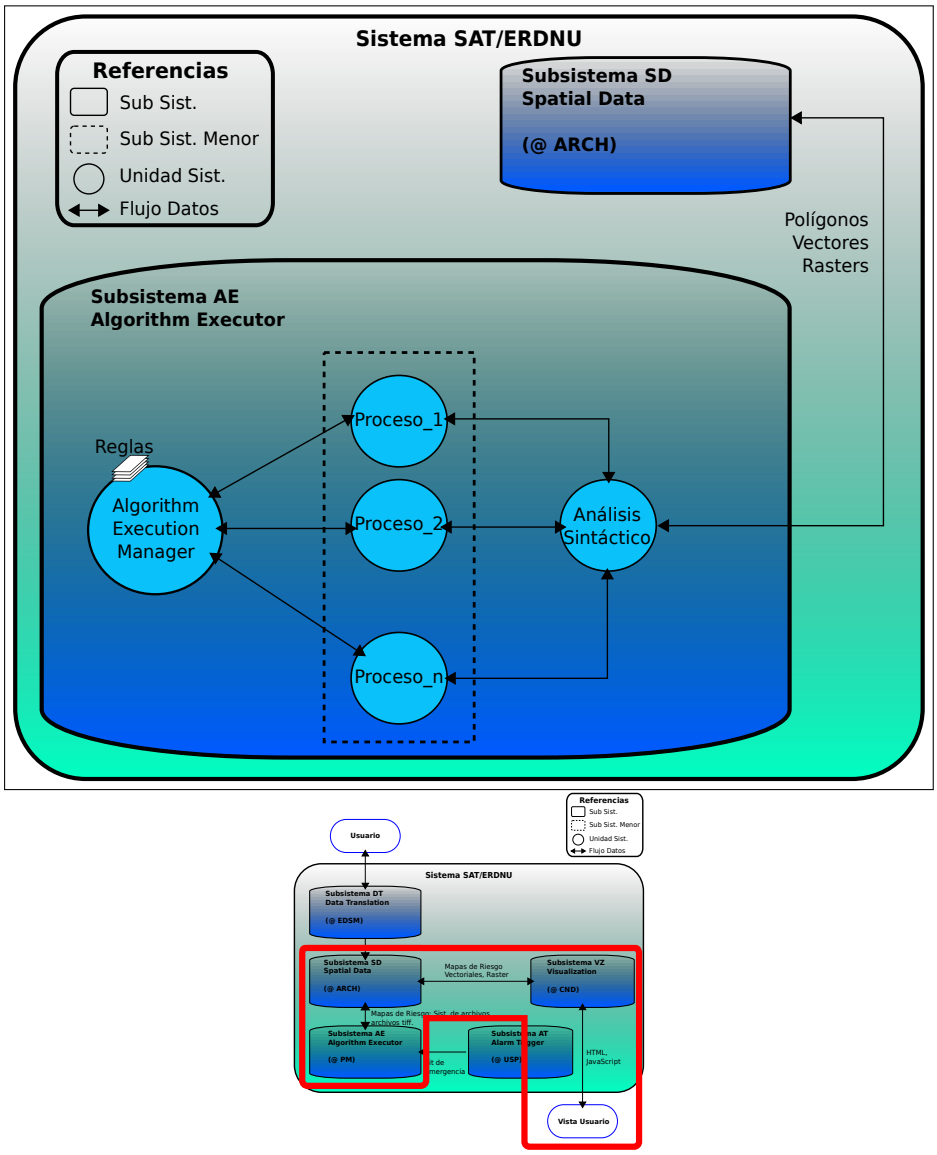


Figura 3.3: Esquema del subsistema AE y su ubicación dentro del SAT/ERDNU

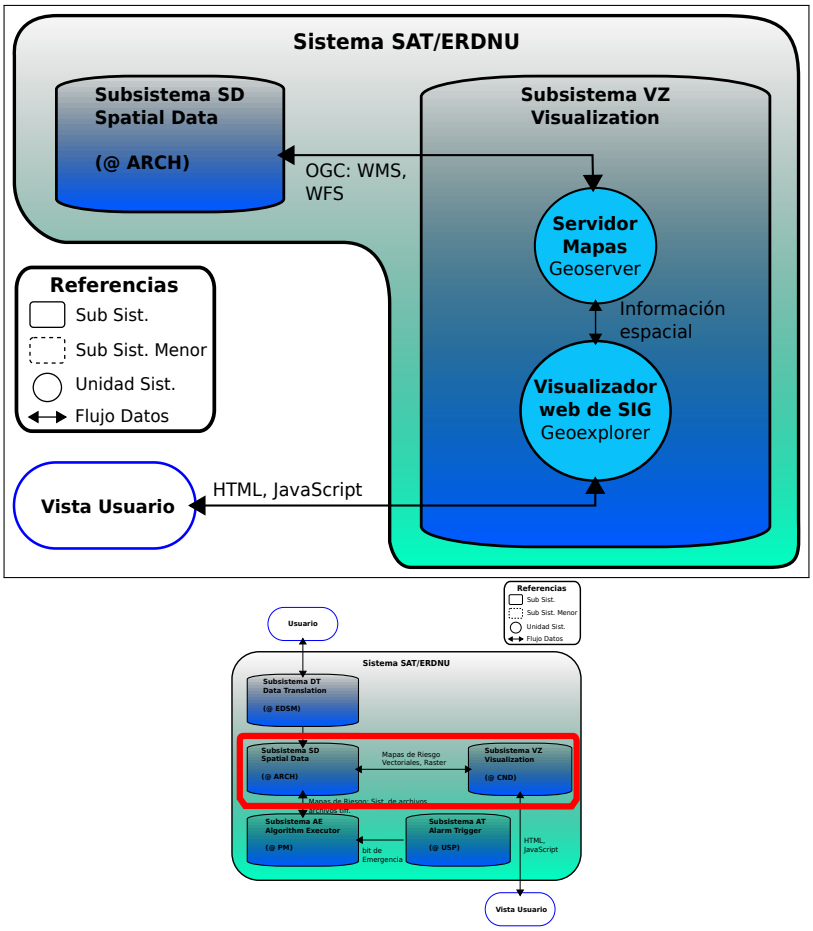


Figura 3.4: Esquema del subsistema VZ y su ubicación dentro del SAT/ERDNU

3.2.4 Módulo GIAEST

El módulo GIAEST es un subsistema menor alojado dentro del subsistema *Algorithm Executor* (AE). Está compuesto por la unidad *JDBC4* y *RParser*.

La unidad *JDBC4* es la que utiliza la librería de Java para interactuar con el subsistema SD, con el objetivo de obtener y depositar registros en la base de datos para el posterior análisis espacio-temporal.

La otra unidad que compone el módulo es *RParser*, encargada de interactuar con el sistema estadístico R, sirviendo de interfaz para el envío correcto de los parámetros necesarios para la ejecución del algoritmo de análisis.

Dentro de los procesos, el *Algorithm Execution Manager* se encargará de priorizar y administrar los pedidos de ejecución de distintos usuarios. Todo el proceso se inicia con la recepción de parámetros por medio del subsistema *Visualization* (VZ) y finaliza con la presentación de los resultados a través del mismo (Figura 3.5).

3.3 Algoritmo del test de Knox

Como se presentó previamente en la sección 1.6.1 de la introducción, este algoritmo permite calcular la intensidad de riesgo relativo que existe que ocurran nuevas aglomeraciones de casos durante un brote epidémico. Lo estima a partir de las aglomeraciones existentes del escenario epidémico real y lo compara con la generación de n escenarios epidémicos aleatorios, evaluando la proximidad de los casos aleatorios en tiempo y en distancia.

Esta teoría es implementada en un algoritmo a través de la unidad *RParser*, en el lenguaje computacional R, debido a la necesidad estadística que requiere. El sistema estadístico R es acoplado al módulo GIAEST por el desempeño que presenta en cálculos estadísticos.

EL algoritmo recibe los parámetros ingresados por los usuarios a través del subsistema VZ, toma los datos del subsistema SD con los parámetros recibidos y ejecuta el cálculo del Test de Knox, en donde se grafica la interacción espacio-temporal resultante indicando el riesgo relativo de que aparezcan nuevas aglomeraciones de casos. La secuencia de comandos que ejecuta el algoritmo en el lenguaje R, se encuentra descripto en el apéndice A.4.

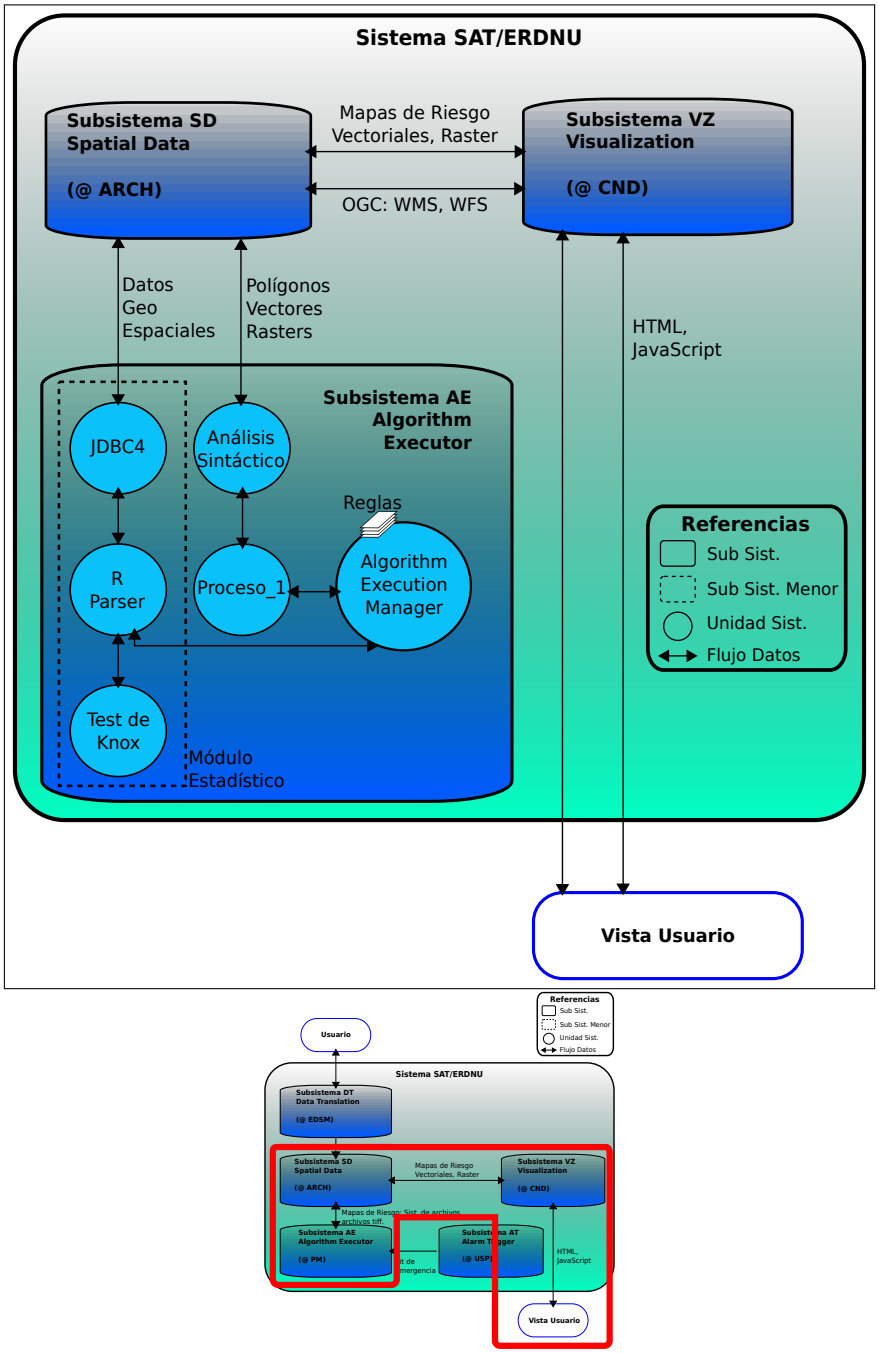


Figura 3.5: Esquema del módulo GIAEST acoplado a la arquitectura del sistema SAT/ERDNU y su ubicación dentro del sistema general

Capítulo 4

Implementación

4.1 Tecnologías utilizadas

Dentro de las tecnologías incorporadas para alcanzar los objetivos, encontramos que el desarrollo orientado a objetos es el de mayor adaptación a proyectos modulares como el HAP. Uno de los mayores beneficios de este tipo de desarrollo es la reusabilidad, lo cual es trivial de alcanzar si se adopta un correcto diseño.

El diseño de software orientados a objetos es complejo, 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 para el problema pero lo suficientemente general para poder reutilizar la solución en problemas y requerimientos futuros, evitando el rediseño o por lo menos reduciéndolo (Gamma et al., 1994).

Teniendo en cuenta la reusabilidad de la solución, se opta por los software de código abierto (OSS, del inglés *Open Source Software*), ya que aseguran un alto porcentaje de éxito en la etapa de adoptar algún patrón de diseño. Así también presentan la gran ventaja de tener un bajo coste, y la colaboración entre desarrolladores y usuarios que van promoviendo ideas que producen soluciones más confiables. El uso de patrones de diseño beneficia en el tiempo la mantenibilidad y reusabilidad del sistema (Morasca et al., 2009).

El software orientado a objetos creció significativamente durante los últimos años, producto del crecimiento en las técnicas de modelado, patrones de diseño y nuevos marcos de trabajo (del inglés, *frameworks*) (Shuang and Cheng, 2009).

Además, desde hace años las aplicaciones comenzaron a manejar con facilidad ambientes multi-usuarios y multi-tareas y con el crecimiento de internet la arquitectura comenzó mayormente a ser cliente-servidor. En la arquitectura cliente-servidor las aplicaciones se reparten entre, los proveedores de recursos o servicios (servidores) y los demandantes (usuarios). En éste tipo de arquitecturas se logra que cada una de las partes que constituían una aplicación pudieran residir en computadoras distintas, aumentando no sólo la capacidad sino también la necesidad de cómputo, permitiendo la evolución hacia las aplicaciones distribuidas.

La arquitectura cliente-servidor es la que está implementada en el sistema SAT/ERDNU, a la cual se inserta el módulo GIAEST. El módulo GIAEST incorpora las tecnologías de código abierto, las técnicas de modelado y los patrones de diseño que fueron implementados. Especialmente las tecnologías en la visualización de mapas, la interfaz de usuario, los protocolos web geoespaciales, y otras tecnologías para la implementación de los análisis espacio-temporales fueron agregadas.

4.2 Tecnologías en la visualización de mapas

A medida que los servicios y aplicaciones en la web fueron evolucionando, también los sistemas de información geográfica fueron adecuándos. Esta evolución permitió el traslado de las funcionalidades SIG, que son brindados en ambientes de escritorio, a la web. El traslado de funcionalidades SIG a la web fue realizado bajo normas y estándares internacionales, los cuales fueron creadas para la distribución en ambientes cliente-servidor. El *Open Geospatial Consortium* se crea como organismo internacional responsable para proporcionar las normas y estándares geográficos en la web.

Así también, organizaciones como Google, Microsoft u OSM han estado aportando una gran contribución, proveyendo bajo diversas licencias sus productos cartográficos Google Maps, Bing Maps y OpenStreetMap respectivamente. Estos productos fueron proporcionando los indicios para montar la visualización de mapas digitales en la web.

Aún así, con las características disponibles que ofrecen estos productos de manera accesible, el usuario SIG requiere de mayores funcionalidades y herramientas para la manipulación de los datos geográficos. Por ello, existe la necesidad de contar con aplicaciones web que brinden las funcionalidades de los software de escritorio a los ambientes web cliente-servidor.

A partir de esta necesidad, surgen organizaciones que incorporan tecnologías web de código abierto y estándares web geográficos para crear aplicaciones de estas características. OpenGeo es una organización que desarrolla

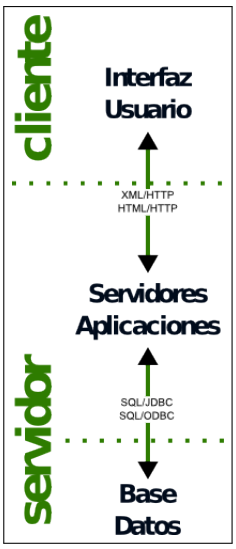


Figura 4.1: Diagrama usual arquitectura cliente/servidor.

soluciones web de código abierto, con tecnologías de visualización para sistemas de información geográfica. Su producto *OpenGeo Suite* se adapta perfectamente a los requerimientos del módulo GIAEST para proveer la solución deseada.

A continuación se detallan los componentes del *OpenGeo Suite* y la facilidad de escalabilidad que brinda por estar desarrollado bajo estándares de código abierto.

4.2.1 El sistema OpenGeo Suite

El diagrama de la arquitectura de una aplicación web genérica usualmente se ve como en la Figura 4.1. Hay una base de datos u otro sistema de almacenamiento de datos en la parte interna, componentes de pequeñas aplicaciones en la capa intermedia, y una capa de interfaz de usuario en la parte externa.

A partir de este diagrama general la organización OpenGeo construye y distribuye la aplicación cliente/servidor de código abierto OpenGeo Suite, como una opción que permite brindar flexibilidad a la incorporación de nuevas funcionalidades demandadas por los no especialistas y agilizar las herramientas de consumo proporcionados por los especialistas. El mismo se implementa con un conjunto de cinco componentes de código abierto, cada uno cumple un papel funcional particular: (1) Almacenamiento: base de datos espaciales PostGIS/-PostgreSQL (2) Servidor de aplicaciones: servidor de mapa/atributos GeoServer (3) Caché de la aplicación: cache de bloques GeoWebCache (4) Marco de interfaz de usuario: GeoExt/ExtJS (5) Componente de mapa de interfaz de usuario: OpenLayers (Figura 4.2).

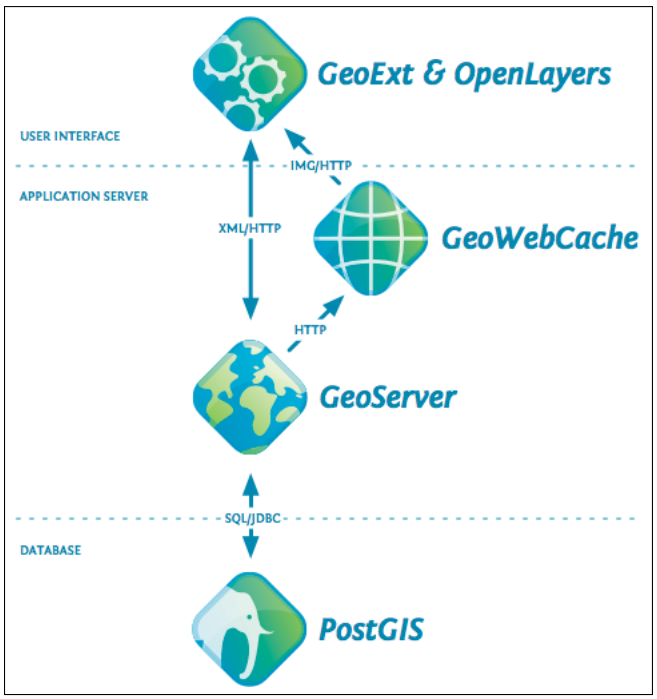


Figura 4.2: Diagrama de arquitectura OpenGeo Suite.

Siguiendo el diagrama usual de arquitectura cliente-servidor, en la capa interna ubicamos el motor de base de datos PostGIS, en la capa intermedia los servidores de aplicaciones Geoserver y GeoWebCache y en la capa externa las librerías OpenLayers y GeoExt.

Los servidores de bases de datos y aplicaciones interactúan a través de SQL[1] (con extensiones espaciales estándares del *Open Geospatial Consortium*). Los servidores de aplicaciones y las capas de la interfaz de usuario interactúan con codificaciones web estándar (XML[2], JSON[3], imágenes) transportados por medio de protocolos HTTP (OpenGeo, 2011).

[1]**SQL** *Structured Query Language*. El lenguaje de consulta estructurado es un lenguaje declarativo de acceso a bases de datos relacionales, que permite especificar diversos tipos de operaciones en ellas. Una de sus características es el manejo del álgebra y el cálculo relacional que permiten efectuar consultas con el fin de recuperar de forma sencilla información de interés de bases de datos, así como hacer cambios en ella.

[2]**XML** *Extensible Markup Language*, XML abreviado, es un lenguaje que describe una clase de objetos de datos llamados documentos XML y describe parcialmente el comportamiento de los programas informáticos que lo procesan (W3C, 2008).

[3]**JSON** *JavaScript Object Notation*. Es un formato de intercambio de datos ligero. De lectura y escritura simples al usuario. Y de análisis y compilación simple para las máquinas. Se basa en un subconjunto del lenguaje de programación JavaScript. JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones estándares de otros lenguajes orientados a objetos.

4.2.2 Almacenamiento

OpenGeo Suite se respalda en la base de datos geoespacial PostGIS/PostgreSQL como fundamento de una arquitectura de código abierto puro. PostgreSQL es un sistema de gestión de bases de datos objeto-relacionales (SGBD) compatible con gran parte del estándar SQL y ofrece muchas características funcionales. Está certificado como compatible con las especificaciones de la OGC de "características simples para SQL".

PostGIS es una extensión espacial de PostgreSQL, y hereda todas las características empresariales de la base de datos subyacente. Además, este tipo de bases de datos como PostGIS agregan tipos, funciones e índices de apoyo a la conservación, gestión y análisis de objetos geoespaciales como: puntos, líneas poligonales, polígonos, multipuntos, multilíneas, multipolígonos y colecciones de la geometría.

Debido a las características de las bases de datos geoespaciales, PostGIS puede almacenar grandes áreas contiguas de datos espaciales y dar lectura/escritura de acceso aleatorio a los datos. Esta es una mejora con respecto a la antigua administración de datos basado en las estructuras de archivos, que eran restringidas por las limitaciones del tamaño del archivo y por la necesidad de bloquear enteramente el archivo durante las operaciones de escritura (OpenGeo, 2011).

Las funciones SQL espaciales disponibles en PostGIS hacen posible el análisis que antes era del dominio de los sistemas de información geográfica de estaciones de trabajo de escritorio:

- Unir dos capas basadas en las reglas de contención espacial.
- Resumir capas basadas en agregados espaciales.
- Crear nuevas capas a través de operaciones espaciales

El OpenGeo Suite soporta además del PostGIS otras bases de datos geoespaciales como Oracle Spatial, DB2 Spatial, ESRI ArcSDE, Microsoft SQL Server Spatial o MySQL Spatial.

Mientras que las bases de datos ofrecen la combinación más fuerte de integridad de los datos, el análisis integrado y soporte para las operaciones de escritura, muchas organizaciones también utilizan otros formatos de almacenamiento para mantener sus datos, tales como los archivos SIG. OpenGeo Suite soporta los siguientes formatos SIG: archivos de capas ESRI (shp), formatos de archivo de imagen, GeoTIFF, ECW, JPEG2000, JPG, PNG, GIF.

4.2.3 Servidores de aplicaciones

La capa intermedia en la arquitectura cliente-servidor es la de servidor de aplicaciones. Es responsable de mediar entre la capa de datos y la capa de interfaz de usuario (UI del inglés *User Interface*). Actúa como una pasarela de protocolo, convirtiendo peticiones web estándar de la interfaz de usuario en las llamadas específicas necesarias para hablar con las bases de datos o de lectura de los archivos SIG. También es un repositorio para la lógica personalizada, como los servicios de procesamiento, y otras rutinas específicas de la aplicación.

OpenGeo Suite soporta el servidor de aplicaciones espaciales GeoServer como el mejor bloque de construcción para los servicios web espaciales, y el servidor de mosaicos GeoWebCache como un acelerador para servicios de mapas personalizados.

Debido a que el OpenGeo Suite está configurado para utilizar protocolos estándar para el acceso y la entrega de datos, es posible sustituir los componentes alternativos, o crear arquitecturas que utilicen múltiples componentes.

La representación de mapas de fuentes de datos PostGIS a una interfaz de WMS se puede hacer con GeoServer, pero también se puede hacer con otros servidores de mapas, tales como Mapserver, Deegree, o las últimas versiones de ArcServer, entre otros. Usando el paquete OpenGeo, las arquitecturas pueden mezclar y combinar diferentes servidores de aplicaciones dependiendo de sus fortalezas correspondientes.

4.2.3.1 GeoServer

GeoServer es un software de enlace (*middleware*) que actúa como servidor de mapas, está escrito en lenguaje Java mediante el cual es posible publicar datos a través de la web. Esta aplicación de software permite una gran interoperatividad utilizando estándares abiertos al presentarse como proyecto de código abierto (OSS). Está construido en gran medida con la librería *Open Sources GeoTools* y se lo instala con un servidor web para aplicaciones Java por defecto. El OGC toma al GeoServer, como una implementación de referencia para servidores de capas, presentando adicionalmente una interfaz gráfica amigable al usuario, mediante la cual se accede a la creación de nuevas capas (OpenGeo, 2011).

GeoServer puede leer de múltiples fuentes de datos, generar múltiples formatos de salida, y comunicarse con múltiples protocolos estándar. Como tal, se ajusta fácilmente en las infraestructuras existentes, proporcionando una ruta de comunicación entre los componentes antiguos y nuevos de software. Los tipos de formatos de las capas a publicar pueden ser PostGIS, archivos de

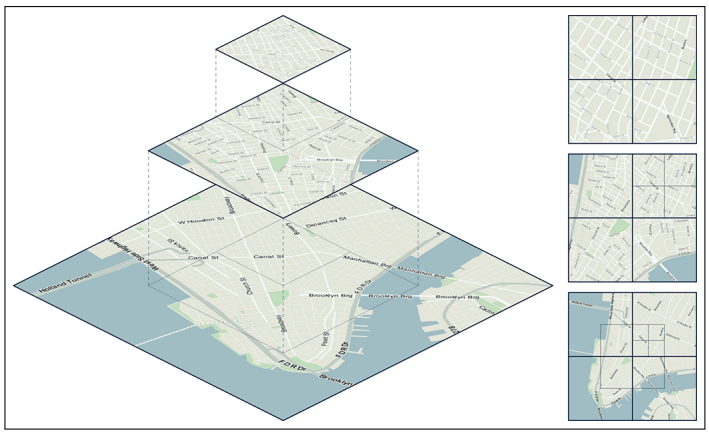


Figura 4.3: Composición de mosaicos de un mapa.

capas ESRI, capas accedidas usando protocolos OGC y capas de formato raster (GeoTiff y Tiff).

4.2.3.2 GeoWebCache

Como GeoServer, GeoWebCache es una pasarela de protocolo en la capa intermedia de la arquitectura. GeoWebCache se encuentra entre los componentes de mapeo de mosaicos (como OpenLayers, Google Maps y Bing Maps) y el motor de capas que presta servicios GeoServer.

Un mapa se compone por medio de mosaicos (Figura 4.3), estos componentes del mosaico generan un gran número de solicitudes paralelas al servidor, los mosaicos siempre tienen los mismos límites, por lo que son los principales candidatos para el almacenamiento en caché de memoria. GeoWebCache recibe solicitudes de mosaicos, comprueba su caché interna para ver si ya tiene una copia de la respuesta, lo devuelve si es así, o los delega al motor de servicio de capas (GeoServer) en caso contrario. Con estos procedimientos se intenta disminuir las demoras de respuesta por parte del servidor (OpenGeo, 2011).

4.3 Interfaz de Usuario

OpenGeo Suite soporta la librería para mapas OpenLayers así como la librería de interfaz de usuario GeoExt, ambas escritas en lenguaje Javascript. Éstas dos librerías son componentes de interfaz de usuario 100% implementadas para el lado cliente. Por su lado, OpenLayers proporciona componentes de mapas e infraestructura asociada al mapa (herramientas de edición de mapas, formatos de objetos espaciales, entre otras), y GeoExt, contribuye al OpenLayers con una

rica estructura de interfaz de usuario al basar sus funcionalidades sobre la librería ExtJS.

Hasta hace poco, las interfaces de usuario altamente interactivas han sido la característica de los software de escritorio, o los sitios web orientados a los complementos (Flash, Java o applets). Con la llegada de los motores de Javascript, de alto rendimiento en los navegadores y complejas librerías de interfaz de usuario como ExtJS, ahora es posible llevar la interactividad de las aplicaciones de escritorio a la web. Las aplicaciones web antiguas requieren de frecuentes actualizaciones de página que lo vuelven más lentas, pero las aplicaciones de construcción Javascript al 100% del lado cliente, no lo necesitan.

La ventaja de las aplicaciones web es la facilidad de implementación y actualización. Actualizar una aplicación de escritorio puede llevar semanas de instalaciones en muchos equipos. Actualizar una aplicación web requiere implementar el código para un solo servidor. Ahora que la calidad de las aplicaciones web es igual a la de las aplicaciones de escritorio, no hay razón para no utilizarla.

4.3.1 Librería OpenLayers

OpenLayers es un componente de mapeo genérico, diseñado para consumir los datos espaciales y los mapas de numerosas fuentes y mostrar los datos en un navegador web. A diferencia de Google Maps o Bing Maps, OpenLayers no está ligado a una fuente de mapa en particular. Puede mostrar mapas de Google, Microsoft o Yahoo!, y mapas personalizados generados por la prestación de motores como GeoServer, MapGuide, MapServer o ArcServer.

Además de la visualización del mapa, OpenLayers ofrece herramientas para trabajar con datos espaciales directamente en el navegador. Componentes de código para leer las características de GeoJSON, GML y formatos de texto. Y herramientas para manipular las funciones de la pantalla: la digitalización, la alteración, y el movimiento de atributos.

4.3.2 Librería GeoExt

ExtJS es una popular librería Javascript del lado del cliente para la creación de interfaces de usuario. Como herramientas de escritorio, incluye componentes pre-construidos para árboles, listas, paneles, tablas, diálogos, etc.

GeoExt agrega extensiones a ExtJS que unen los componentes básicos ExtJS a las características espaciales de OpenLayers. Por ejemplo, un "gestor de selección" GeoExt que se parece a una tabla en la interfaz de usuario está vinculado al conjunto de las entidades seleccionadas en OpenLayers.

El conjunto de componentes GeoExt permite a OpenGeo desarrollar rápidamente aplicaciones espaciales personalizadas, ya que no tenemos que re-escribir constantemente los enlaces entre el componente de mapa y el resto de componentes de la interfaz de usuario.

4.3.3 GeoExplorer

GeoExplorer es una aplicación de código abierto, del lado cliente, que usa las capacidades brindadas por librerías JavaScript, tales como ExtJS, GeoExt u OpenLayers, para implementar un marco de trabajo de visualización SIG en la web. Este marco de trabajo se enfoca en visualizar y administrar capas a través de los protocolos estándares de la OGC, con la habilidad de consultar los datos asociados a las capas y mostrar los resultados.

4.4 Protocolos web geoespaciales

El *Open Geospatial Consortium* (OGC), desde su aparición en 1994, ha especificado más de 30 estándares. Entre los más importantes se puede destacar al servicio de mapas web WMS (por sus siglas del inglés *Web Map Service* y el servicio de entidades vectoriales web WFS (del inglés *Web Feature Service*). Mientras el WFS está relacionado con el acceso directo a los datos: leer, escribir y actualizar las características de los mismos (atributos). El WMS está relacionado con la transformación de los datos en un mapa (imagen). Un atributo es un objeto que abstrae el fenómeno real. Éste objeto tiene un conjunto de propiedades asociadas, cada una de las cuales tiene un nombre, un tipo, y un valor. Un ejemplo de atributo podría ser un camino con un nombre y una ubicación (linea geométrica), límite de velocidad, etc. Típicamente estos atributos son almacenados en una base de datos espacial, un archivo de formato capa (*shapefile*) o cualquier otro formato vectorial.

4.4.1 Web Feature Service (WFS)

El estándar *Web Feature Service* (WFS) representa un cambio en la forma en que la información geográfica es creada, modificada e intercambiada en Internet. En lugar de compartir la información geográfica a nivel de archivos utilizando, por ejemplo el protocolo de transferencia de archivos (FTP), WFS ofrece acceso minucioso y directo a la información geográfica hasta sus características o atributos utilizando protocolos HTTP (Consortium, 2005).

Proporciona operaciones para crear, modificar, eliminar y consultar instancias de una entidad en un almacén de datos subyacente. Las operaciones

WFS se invocan mediante cualquier plataforma de computación distribuida HTTP-transportables para enviar un mensaje de solicitud a un servicio, que a continuación, procesa la solicitud y genera un mensaje de respuesta. Esta especificación define la estructura lógica de los mensajes de solicitud y de respuesta. Sin embargo el contenido y el formato de un mensaje, de petición y respuesta, depende de la operación que se invoque. Se puede pensar un WFS como el código fuente de los mapas enviados, que pueden ser vistos mediante WMS (Consortium, 2008).

4.4.2 Web Map Service (WMS)

Un servicio *Web Map Service* (WMS) produce mapas de datos referenciados espacialmente de forma dinámica a partir de información geográfica. La norma internacional define "mapa" a una representación de la información geográfica como un archivo de imagen digital conveniente para la exhibición en una pantalla de computador. En un mapa no están los datos propiamente. Mapas producidos por WMS se generan normalmente en un formato pictórico como PNG, GIF o JPEG, y ocasionalmente como elementos gráficos basados en formatos de vectores tales como el gráfico vectorial escalable (SVG del inglés *Scalable Vector Graphics*) o el meta archivo gráfico de computador web (WebCGM del inglés *Web Computer Graphics Metafile*).

La norma internacional WMS define tres operaciones: 1) devuelve metadatos a nivel de servicio, 2) devuelve un mapa cuyos parámetros geográficos y dimensionales están bien definidos, y 3) devuelve información sobre las características particulares mostradas en el mapa (operación opcional). Operaciones WMS pueden ser invocadas usando un navegador web estándar mediante la presentación de solicitudes en forma de URI[1]. El contenido de tales URI depende de qué operación se solicita. En particular, al solicitar un mapa, la URI indica qué información es la que se muestra en el mapa, que parte de la Tierra se va a mostrar, el sistema de coordenadas de referencia deseado, el ancho de la imagen de salida y la altura. Cuando dos o más mapas se producen con los mismos parámetros geográficos y tamaño de salida, los resultados pueden ser superpuestos con precisión para producir un mapa compuesto. El uso de formatos de imagen que soportan fondos transparentes (por ejemplo, GIF o PNG) permite que los mapas subyacentes sean visibles. Además, los mapas individuales se pueden solicitar a diferentes servidores. El servicio *Web Map Service* permite así la creación de una red de servidores de mapas distribuidos, de la cual los clientes pueden crear mapas personalizados (Consortium, 2006).

[1] **URI** *Uniform Resource Identifier*. Un identificador uniforme de recursos (URI) proporciona un medio simple y extensible para la identificación de un recurso en internet. Con una sintaxis, una semántica, recomendaciones y requisitos funcionales propios (Berners-Lee et al., 2005).

El WMS se aplica a una instancia que publica su capacidad para producir mapas en vez de su capacidad para acceder a bancos de datos específicos. Un WMS básico clasifica sus activos de información geográfica en "capas" y ofrece un número determinado de "estilos" predefinidos en el que aparecerán las capas.

4.4.3 Web Feature Service - Transactional (WFS-T)

El estándar WFS es un protocolo con bastante potencial para efectuar acciones de cambios en la información original de un objeto, y esto representa un riesgo sino se define un diseño adecuado de seguridad, con políticas que controlen el acceso a la alteración de la información. Debido a esta situación, el protocolo del estándar WFS requiere de una diferenciación para aquellos que no utilizarán todas las funcionalidades del protocolo.

La librería OpenLayers aprovecha este potencial riesgo y lo transforma en una funcionalidad más, brindando una diferenciación en el uso del protocolo WFS por medio de filtros a sus funcionalidades. Crea una clase WFS-T, en términos de programación orientada a objetos, con métodos que permiten efectuar las transacciones de escritura y modificación de la información dependiendo de como se la solicite, de aquí surge el *WFS-Transactional*.

Esta clase es posteriormente utilizada en los servidores de aplicaciones de mapas, como el GeoServer, quienes administran las políticas de seguridad definidas para el acceso a la información. No obstante la administración de la seguridad queda respaldada con los roles de usuarios que proveen los sistemas gerenciadores de bases de datos, como el PostgreSQL/PostGIS u otros.

4.5 Particularidades del módulo GIAEST

El módulo GIAEST se implementa sobre las tecnologías descriptas anteriormente. Haciendo uso correcto del diseño orientado a objetos y reutilizando las tecnologías de código abierto para adherir valor agregado a estos componentes. Si bien toda la arquitectura del OpenGeo Suite da un respaldo importante en los niveles de almacenamiento, servidor de aplicaciones e interfaz de usuario que corresponden a una arquitectura cliente/servidor, se requieren aún de otras tecnologías y aplicaciones que ejecuten los análisis espacio-temporales, y distribuyan los resultados de la información que va siendo almacenada.

Se agregan funcionalidades en todos los niveles para el uso adecuado de la herramienta de análisis. A nivel externo del sistema se agrega una interfaz independiente donde se recepciona y validan los datos que van a ser ingresados como parámetros para el análisis (Ver detalle de validaciones en la sección 2.2.1

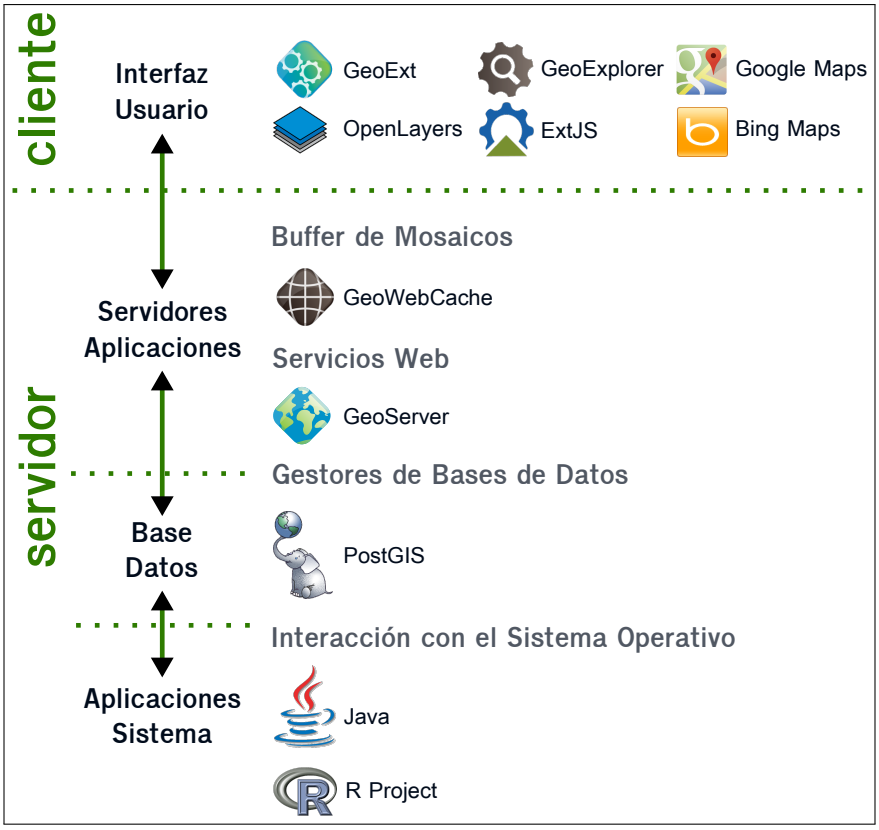


Figura 4.4: Componentes finales de la arquitectura cliente/servidor con el módulo GIAEST.

- Control de campos y validaciones). En el nivel intermedio de la arquitectura del sistema se especifican los roles de usuario para el acceso y uso de la información. A nivel interno del sistema se utiliza el lenguaje de programación Java para realizar las interconexiones entre los demás componentes de almacenamiento, aplicaciones independientes y el nivel externo de interfaz de usuario (Figura 4.4).

4.5.1 Java Servlet

Los *servlets* son la tecnología de la plataforma Java de elección para la ampliación y mejora de los servidores web. Proporcionan un método basado en componentes, independiente de la plataforma para crear aplicaciones basadas en la web, sin las limitaciones de rendimiento de otros programas. A diferencia de los mecanismos de extensión de servidor propietario, los servlets son independientes del servidor y de la plataforma. Esto deja libre el camino para seleccionar una estrategia de variadas opciones para servidores, plataformas y herramientas.

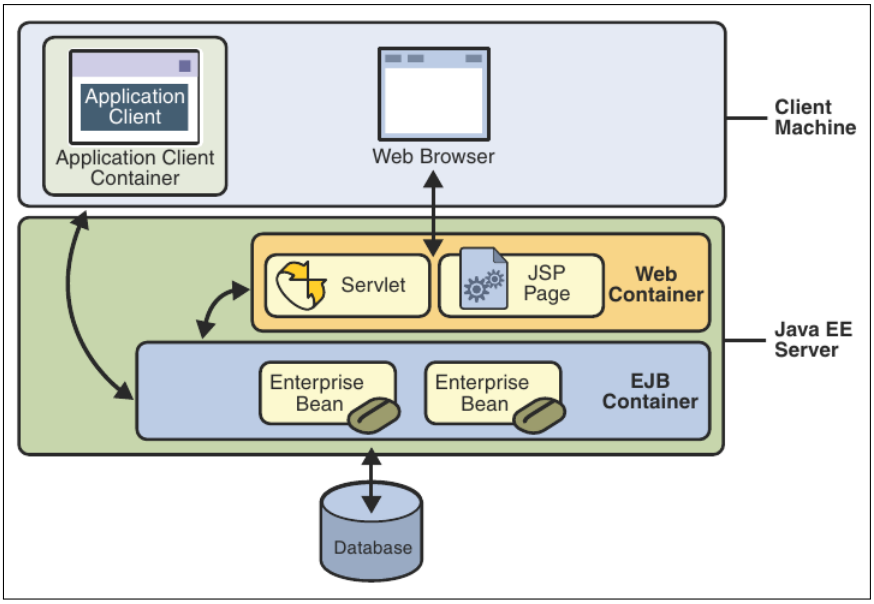


Figura 4.5: Esquema de contenedores y servidores de JavaEE como ejemplo (Oracle, 2010).

Técnicamente un servlet se define como un componente Java basado en tecnología web, gestionado por un contenedor, que genera contenido dinámico. Al igual que otros componentes basados en la tecnología Java, los servlets son clases Java independientes de la plataforma que se compilan en código de bytes para plataformas neutras que pueden ser cargadas de forma dinámica y se ejecutan en un servidor web habilitado para la tecnología Java. Los contenedores, a veces llamados motores de servlets, son extensiones de servidor web que proporcionan funcionalidad servlet. Los servlets interactúan con clientes web a través de un paradigma de solicitud-respuesta (Figura 4.5) implementada por el contenedor de servlets (Mordani, 2009).

Los servlets tienen acceso a toda la familia de API[1] de Java, incluyendo la API JDBC para acceder a bases de datos empresariales. Los servlets también pueden acceder a una librería de las llamadas HTTP específicas y recibir todos los beneficios del lenguaje Java maduro, como la portabilidad, rendimiento, reutilización y protección contra imprevistos.

El GIAEST hace uso de los servlets para ejecutar las API de Java en el servidor web, recibe los parámetros, los transforma para la interpretación de las tecnologías de la plataforma y permite la interacción con aplicaciones

[1] **API** *Application Programming Interface*. La interfaz de programación de aplicaciones es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta librería para ser utilizada por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas de librerías de un determinado lenguaje de programación.

específicas. Por un lado interactúa con el sistema de almacenamiento y por otro con la aplicación estadística que ejecuta el algoritmo de análisis espacio-temporal, con la respectiva devolución de los resultados.

4.5.2 Librerías Java utilizadas

De entre las librerías de Java que se utilizaron para la construcción de las clases propias que actúan como serlvets, podemos destacar:

servlet 3.0 La librería servlet de javax para administrar las solicitudes y respuestas bajo protocolos HTTP.

gson 2.2 Se refiere a la librería desarrollada por Google para el empaquetado de objetos en formato JSON. Este formato se utiliza para serializar y transmitir datos estructurados sobre una conexión de red, principalmente entre un servidor y una aplicación web (interfaz externa).

JDBC para PostgreSQL El *Java Data Base Connector* (JDBC) es una librería que provee funcionalidades genéricas y específicas para conectarse a un motor de base de datos, y a través de esa conexión ejecutar sentencias SQL. De este modo, se obtiene información de la base de datos, dependiendo de los permisos de acceso que tenga asignado el rol con el cual se conecta el usuario. En este caso particular se utiliza el JDBC para motores de bases de datos PostgreSQL.

R-Caller Es una librería de código abierto desarrollado por el Dr.M.Hakan Satman en el cual se utiliza el núcleo de la aplicación estadística R desde el motor Java, desde allí se envían sentencias en formato interpretable por R para que se ejecuten las peticiones y se devuelvan los resultados.

4.6 Estadísticas con R

R es un lenguaje y entorno para cálculos estadísticos y gráficos. Ofrece una gran variedad de estadísticas (modelos lineales y no lineales, pruebas estadísticas clásicas, análisis de series temporales, clasificación, aglomeración, etc.) y técnicas gráficas, además de ser altamente extensible.

Uno de los puntos fuertes de R es la facilidad con la que se pueden producir gráficos bien diseñados con calidad de publicación, como símbolos matemáticos y fórmulas donde sea necesario.

Por estas razones, y debido a las destacables propiedades numéricas (R Development Core Team, 2013), se utilizó el núcleo principal de R para la ejecución del algoritmo del test de Knox, y sus librerías gráficas para la representación bidimensionales de los resultados.

Capítulo 5

Interfaces del sistema

Este capítulo presenta las interfaces externas desarrolladas para cumplir parte de los requerimientos del módulo GIAEST. Como se ha mencionado, la herramienta es de acceso web, por lo tanto se accede a ella través de un URI, usando cualquier navegador web (Figura 5.1).

5.1 Guía de uso del sistema

Siguiendo el primer paso que fue expuesto anteriormente (donde es ingresada la ruta de direccionamiento al servidor), se mostrará la interfaz principal (Figura 5.2). Seguidamente, como muestra la Figura 5.3, ésta interfaz principal se divide en tres secciones: la barra de herramientas - sección (A), panel de capas - sección (B), y el panel del mapa - sección (C).

La sección (A), contiene los controles principales de la interfaz y brinda distintas funcionalidades para interactuar con las capas de información, tales

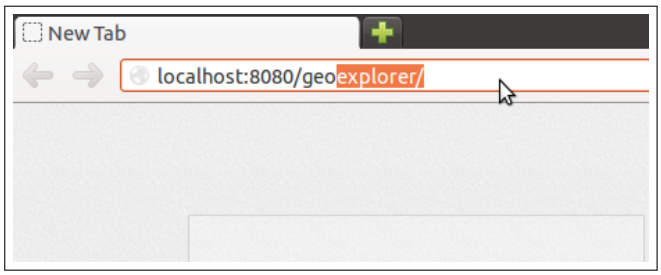


Figura 5.1: Ingreso de la dirección del servidor web SIG en el URI.

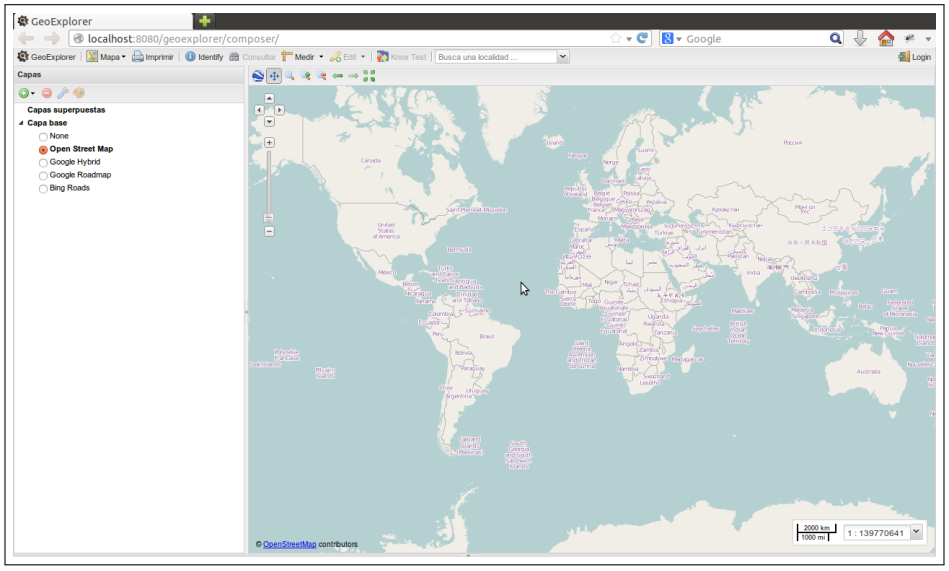


Figura 5.2: Interfaz principal del sistema.

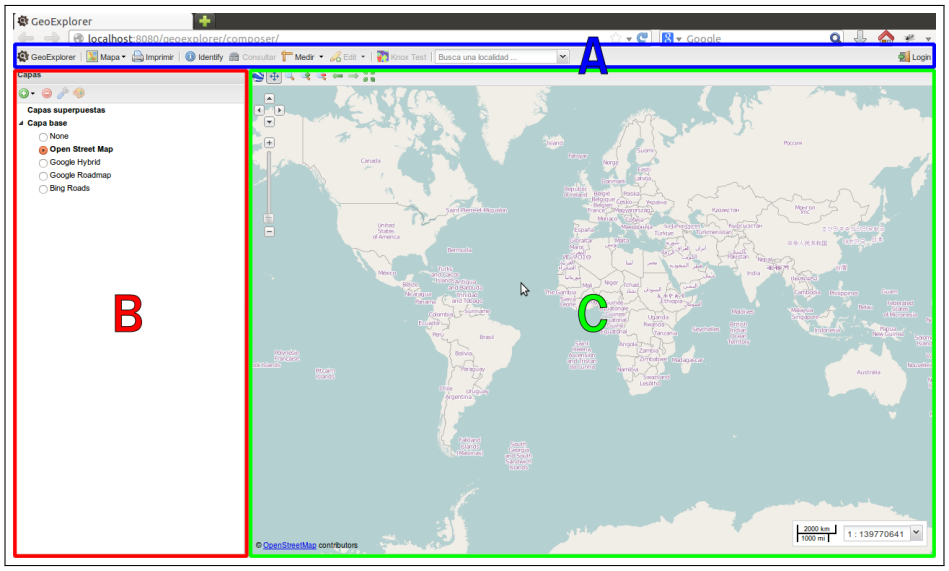


Figura 5.3: Divisoria de la interfaz. (A) Barra de herramientas, (B) Panel de capas, (C) Panel del mapa.

como consultas, zooms, exportación de mapas, entre otros (Figura 5.4).

En la sección (B), principalmente se listan las capas disponibles, con sus respectivas leyendas. Las capas se agrupan en (1) capas superpuestas y (2) caspas base. Las capas superpuestas son las capas propias de cada usuario, son las que están alojadas en el servidor de mapas GeoServer o en algún otro servidor externo; en cambio las capas base son los mapas públicos de Google, BingMaps u OpenStreetMap, que proporcionan el contexto general que necesita el usuario para ubicarse geográficamente con sus capas.

Finalmente en la sección (C), se construye una vista superpuesta de las capas base con las capas propias del usuario, generando el mapa con información que el usuario desee visualizar. Este panel del mapa contiene los controles de navegación dentro del mapa.

5.1.1 Inicio de sesión

Una vez que el usuario visualiza la interfaz principal, debe iniciar sesión presionando el botón de la Figura 5.4i, ubicado en el extremo derecho de la barra de herramientas. De esta manera se habilitarán los demás controles que se encuentran inhabilitados para usuarios anónimos. Posteriormente se despliega una ventana donde deberá ingresar el nombre de usuario y contraseña, como se muestra en la Figura 5.5a.

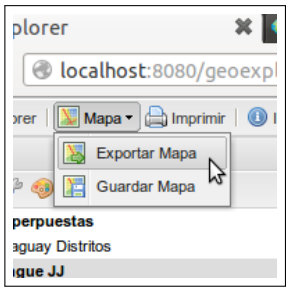
Cuando el usuario inicia sesión de manera exitosa, automáticamente el botón *Login*, cambiará de nombre a *Logout*. Esto indica que al presionarlo nuevamente, se cerrará la sesión correspondiente al usuario conectado (Figura 5.5b). De esta manera se podrá acceder al servidor de mapas.

5.1.2 Manejo de capas

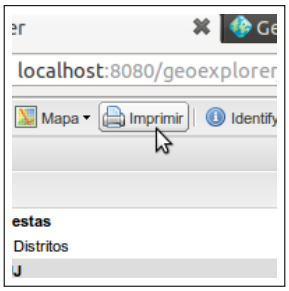
Luego de iniciar sesión, el usuario tiene la posibilidad de ir agregando sus capas al panel de capas superpuestas. Podrá agregar y eliminar capas del panel de capas, modificar el orden de superposición, ocultar capas y realizar modificaciones en el estilo de visualización. Se aclara que al eliminar la capa, sólo es eliminada de la vista de los paneles.

Para agregar una capa al panel, se debe pulsar el botón *añadir capas* que se muestra en la Figura 5.6a, ubicado en el extremo superior izquierdo del panel de capas. Al pulsar el botón, en el mismo panel de capas, aparece una lista desplegable de los servidores de mapas disponibles, de los cuales se podrían obtener las capas, ya sea de un servidor público o privado Figura 5.6b.

Para nuestro caso, el servidor local que contiene las capas propias del usuario es el *GeoServer Local*. Al seleccionar el servidor se mostrarán, como



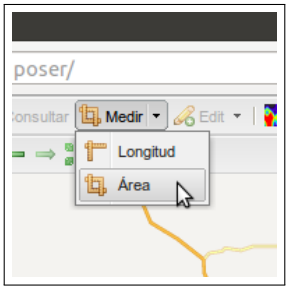
(a) Botón mapa, exportar y guardar mapa.



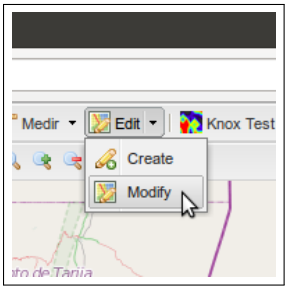
(b) Botón imprimir, imprime capas propias.

(c) Botón identificador, información detallada de un elemento de una capa.

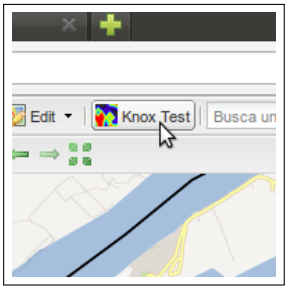
(d) Botón consultar, realiza consultas a atributos de capa.



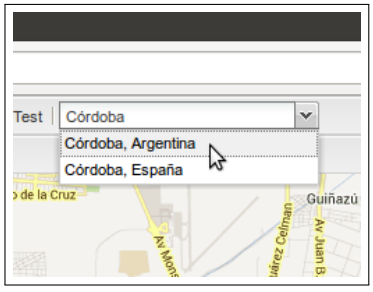
(e) Botón medir, calcula distancias y áreas.



(f) Botón editar, agrega nuevos o modifica elementos de la capa.



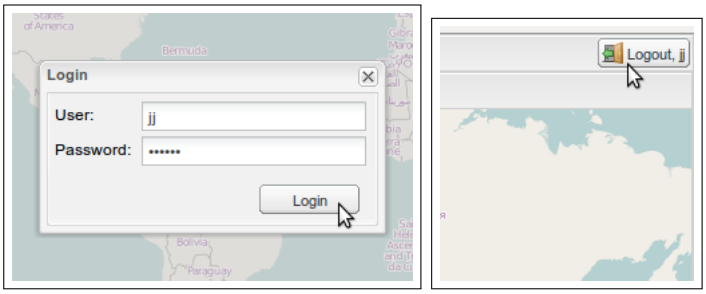
(g) Botón test de Knox, introduce parámetros para análisis espacio-temporal.



(h) Localizador, realiza búsquedas de calles o ciudades.

(i) Botón de inicio de sesión de usuarios.

Figura 5.4: Funcionalidades principales de la barra de herramientas.



(a) Ventana de inicio de sesión. **(b)** Botón de cierre de sesión de usuario.

Figura 5.5: Controles de sesión de usuario.

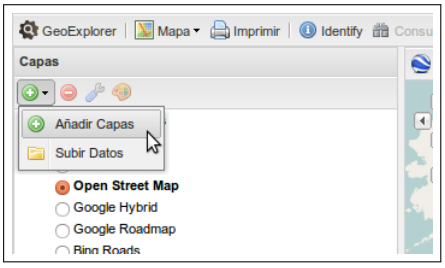
en la Figura 5.6c, las capas que dispone dicho servidor para ofrecer al usuario conectado, de acuerdo a su rol.

De las capas disponibles que muestra el servidor, el usuario elige con que capa trabajar por medio de dos opciones, 1) haciendo doble click sobre el nombre de la capa y luego volviendo al panel de capas presionando el botón *Capas* mostrado en la Figura 5.6d, o 2) haciendo un click sobre el nombre de la capa, añadirlo al panel de capas con el botón *Añadir Capas* y aceptando con el botón *Hecho*, ambos botones se presentan en el extremo inferior derecho del panel de capas.

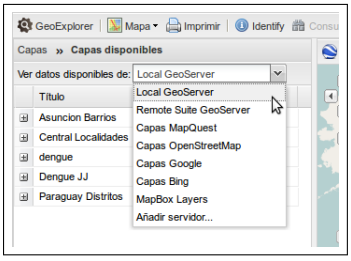
Las capas obtenidas desde el servidor pueden ser de cualquier formato geográfico, ya sean en formatos vectoriales o raster. Habrá una administración de la base de datos que se encargará de proveer las capas y mantenerlas en el servidor. El usuario con los roles adecuados tiene la posibilidad de subir otras capas propias al servidor de mapas GeoServer además de las que ya están provistas, pero se específica que dichos mapas no se almacenarán en el servidor central de datos PostGIS sino solo en el servidor intermedio GeoServer. Este almacenamiento está regido por las políticas que posee la administración de la base de datos central.

Una vez agregadas las capas, el usuario tiene la opción de manipularlas de acuerdo a sus preferencias. Puede mostrarlas u ocultarlas respectivamente con el marcado o el desmarque del pequeño cuadro que se encuentra delante del nombre de la capa como vemos en la Figura 5.7a. Si el usuario desea cambiar el orden de superposición en el que se visualizan las capas, debe seleccionar la capa deseada con un click y sin soltar el click arrastrar la capa, como se muestra en la Figura 5.7b, hasta posicionarla en el orden que la desee.

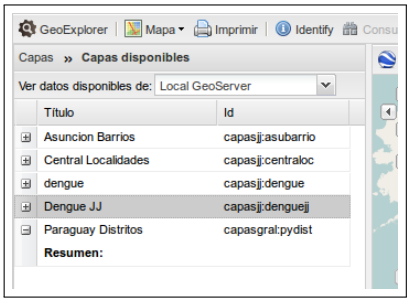
Para cambiar el estilo referente al tamaño, bordes, rellenos o colores como se visualizan las capas en el mapa, el usuario puede acceder a las opciones del estilo a través del botón *estilo* que se encuentra en el extremo superior derecho del panel de capas como se ve en la Figura 5.7c.



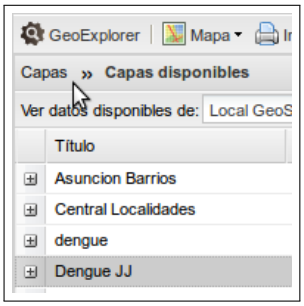
(a) Botón añadir capa.



(b) Menú desplegable de servidores de mapas.

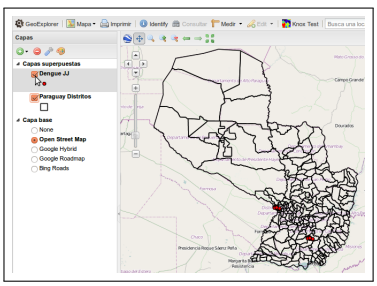


(c) Selección de la capa a visualizar.

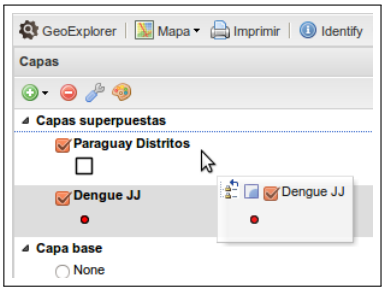


(d) Volver al panel de capas.

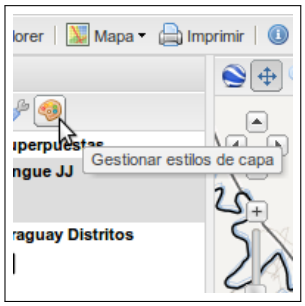
Figura 5.6: Controles para el manejo de capas (1).



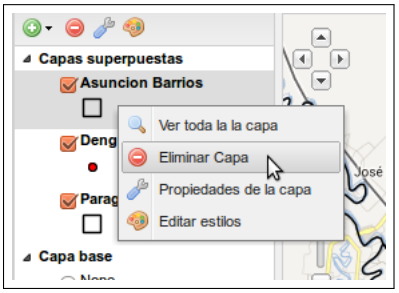
(a) Capa seleccionada del panel de capas.



(b) Cambio en el orden de superposición de capas.



(c) Botón estilo de capa.



(d) Menú contextual de la capa.

Figura 5.7: Controles para el manejo de capas (2).

Así también el usuario puede borrar una capas del panel presionando el botón *eliminar capa*, que se encuentra al lado de botón *añadir capa* del mismo panel. Este botón no elimina físicamente el archivo de la capa en la base de datos, solo lo quita de la visualización en la interfaz. A todas estás opciones, el usuario también puede acceder haciendo click derecho sobre la capa que quiera manipular y obtendrá un menú contextual como muestra la Figura 5.7d con estas opciones.

5.1.3 Manejo de datos

El usuario, dependiendo de rol que posea, tendrá la posibilidad de manipular los datos que contiene cada capa. Podrá añadir, editar y/o eliminar información, como así también efectuar consultas en base a los atributos de la capa.

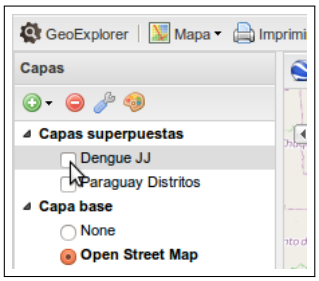
Para añadir nueva información a una capa, el usuario deberá seleccionar previamente de entre las capas superpuestas (Figura 5.8a), la capa que desea editar como muestra la Figura 5.8b. De esta manera activa el botón *editar* del menú de herramientas y selecciona la opción *crear* según la Figura 5.8c.

Seguidamente debe posicionarse en el mapa sobre la ubicación geográfica que corresponda y hacer click. Al pulsar el click se despliega una ventana emergente con un formulario para rellenar con los datos correspondientes según la Figura 5.8d. Una vez completado el formulario debe pulsar el botón *guardar* de la ventana para confirmar la operación.

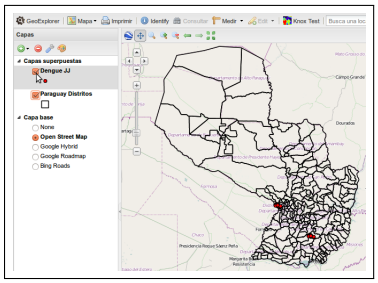
En el caso de la edición de datos existentes se debe proceder del mismo modo que para añadir un nuevo dato, se pulsa el botón *editar* y se elige la opción *modificar*. Luego de seleccionar la opción *modificar* se debe identificar el dato específico que se desee modificar, ya sea por medio de una búsqueda visual o por medio de una consulta a los registros de la capa, la cual se realiza presionando el botón *consultar* de la Figura 5.4d.

Una vez identificado el registro a modificar, se pulsa click sobre el mismo y se despliega la ventana de la Figura 5.9a con el formulario que presenta los datos del registro. Luego para habilitar la edición de los campos, se presiona el botón *editar* de la ventana emergente. Al finalizar la edición debe confirmar la operación pulsando el botón *guardar*.

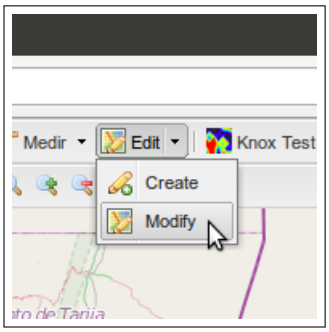
Para la eliminación de información se procede de la misma manera que para la edición de datos existentes. Se identifica el registro que se quiere eliminar, se pulsa click sobre el mismo y se despliega la ventana emergente de edición. En este caso se elige la opción *borrar* como indica la Figura 5.9b. Para confirmar la eliminación del registro debe presionar la opción *Yes*, que se muestra en la figura Figura 5.9c. En caso de querer cancelar la operación, debe pulsar la opción *No*.



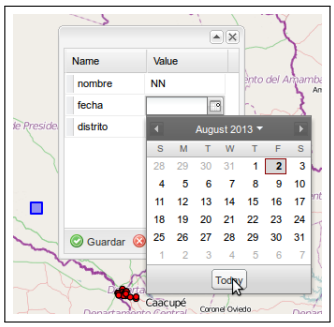
(a) Capas superpuestas en el panel de capas sin seleccionar.



(b) Capa seleccionada del panel de capas.

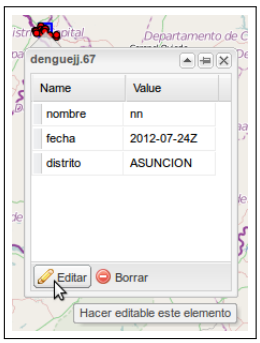


(c) Botones de edición para crear o modificar datos.

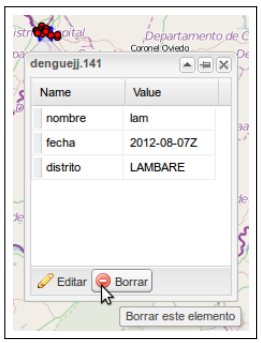


(d) Formulario para ingreso de datos.

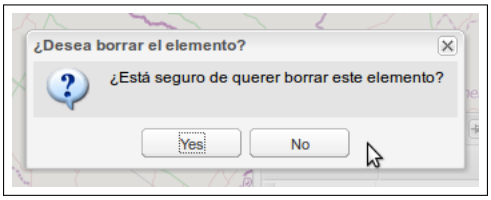
Figura 5.8: Controles del manejo de edición de datos (1).



(a) Ventana emergente para edición.



(b) Ventana emergente para eliminación.



(c) Mensaje emergente de confirmación.

Figura 5.9: Controles del manejo de edición de datos (2).

5.1.4 Ejecución del test de Knox

El test de Knox puede ejecutarse una vez que existan registros de datos en las capas de casos de enfermedades. Se accede por medio del botón *test Knox* que se muestra en la Figura 5.10b, el cual despliega una ventana emergente con un formulario a ser rellenado con los parámetros necesarios para el test, como muestra la Figura 5.10a.

En este formulario primeramente se selecciona el campo "programa", que hace referencia a los programas de salud del MSAL que se requiera analizar, para este caso se utiliza el programa de dengue. El sistema contempla proveer análisis espacio-temporales para otros programas de salud del MSAL como chagas, leishmaniasis u otros.

Seguidamente el sistema realiza una búsqueda interna de entre las ciudades donde se registraron los casos de esta enfermedad y las lista en el campo "ciudad" del formulario.

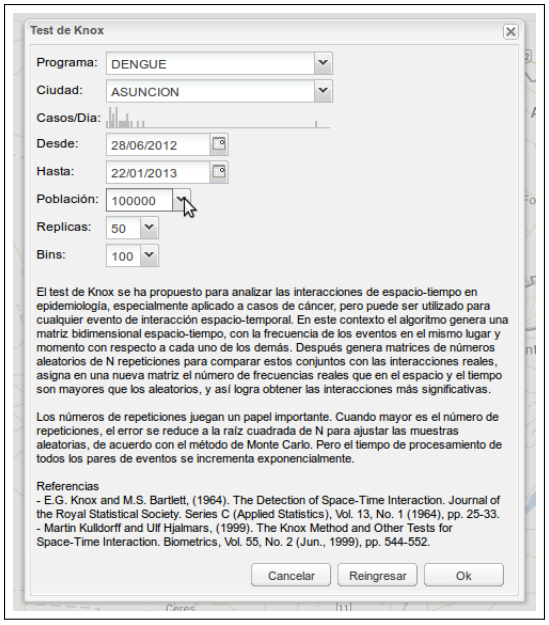
Una vez seleccionada la ciudad para el análisis, se rellenan automáticamente los campos "desde" y "hasta", que corresponden al periodo de fechas para el cual será ejecutado el análisis. De manera predeterminada el sistema completa con la fecha del primer caso registrado en el campo "desde", y la fecha del último caso registrado en el campo "hasta". Si el usuario desea otro periodo de fechas para el análisis, deberá especificarlo.

El campo "réplicas" hace referencia a la cantidad de situaciones epidémicas aleatorias, similares en espacio y tiempo, que generará el algoritmo para evaluarlo con la situación epidémica real ocurrido en el periodo de fechas elegido. Cabe destacar que a medida que aumenta la cantidad de réplicas, aumenta considerablemente el tiempo de cálculo en generar un resultado. Está predeterminada una cantidad intermedia de réplicas que puede ser ajustada.

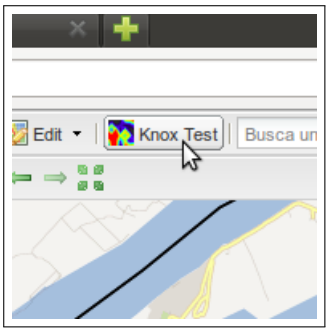
Finalmente el último campo "bins", es un valor numérico que necesita el algoritmo para proporcionar un gráfico de mayor o menor resolución de píxeles de acuerdo a la cantidad de bins ingresada. A mayor cantidad de bins, mayor resolución gráfica del resultado.

Completados todos los parámetros necesarios para efectuar el test presionamos el botón *ok*. Los datos son enviados al servidor para ejecutar el algoritmo. Una barra de procesamiento, como se muestra en la Figura 5.10c, indica al usuario que el proceso se encuentra en ejecución. Al concluir el test se abrirá en otra ventana del navegador, los gráficos resultantes del análisis espacio-temporal del test de knox como se muestra en la Figura 5.10e.

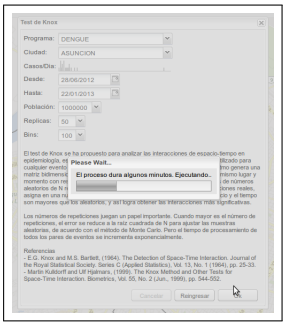
Es importante completar adecuadamente el formulario para el envío de parámetros. En caso de completar incorrectamente algún campo, el sistema devolverá mensajes de error como se indica en la Figura 5.10d, solicitando un



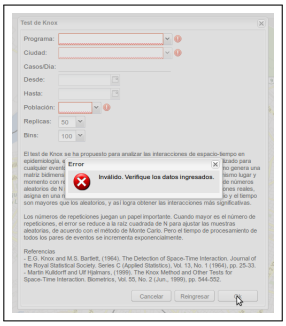
(a) Ventana emergente con parámetros del test.



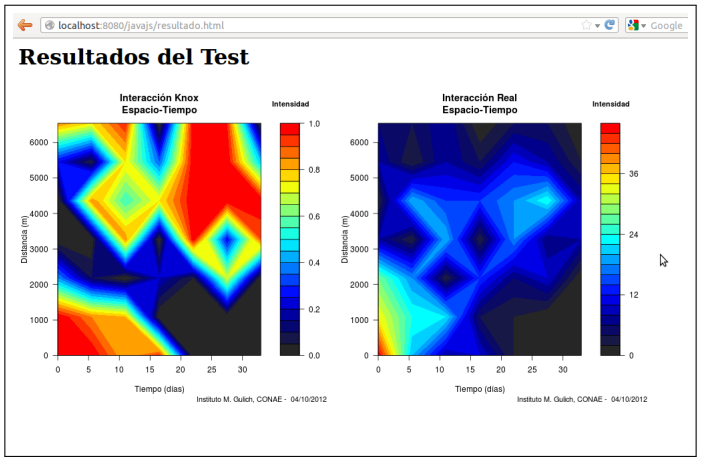
(b) Botón para test de Knox.



(c) Mensaje emergente de confirmación.



(d) Mensaje de error en el llenado de parámetros.



(e) Gráficos resultantes del test de Knox.

Figura 5.10: Controles del test de Knox y resultados.

correcto llenado de los mismos.

5.2 Otras funcionalidades

En esta sección se presentan las demás funcionalidades de la interfaz y se exponen con mayor detalle algunas otras que fueron brevemente mencionadas en la sección anterior. Estas funcionalidades proporcionan mayor versatilidad al sistema.

Búsqueda de ubicaciones geográficas. La interfaz del sistema provee el buscador de ubicaciones geográficas desarrollado por la empresa Google™ para su producto ©Google Maps. Este buscador es bastante robusto para encontrar ubicaciones alrededor del globo terrestre. Permite buscar desde países hasta nombres de calles de una ciudad o localidad.

El usuario debe escribir sus búsquedas con el siguiente orden y separar los campos por ",": nombre de la calle, ciudad o localidad y país, como se muestra en la Figura 5.11. Este funcionalidad se encuentra en el centro del menú de herramientas de la interfaz.

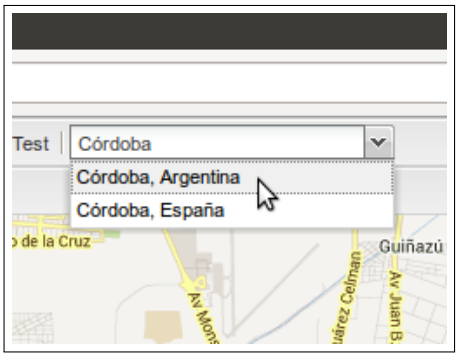


Figura 5.11: Buscador de ubicaciones geográficas

Impresión de mapas. Se provee la funcionalidad de impresión de mapas en formato PDF, a través del botón *imprimir* del menú de herramientas (Figura 5.4b). El mismo toma la vista actual del panel del mapa con las capas propias del usuario y muestra una ventana con la vista capturada de las capas en exposición (Figura 5.12).

El usuario puede ajustar el área de impresión a la escala deseada, el tamaño del papel a imprimir y la resolución con la que se imprimirá el documento. Se aclara que no todos los mapas de base tienen permiso de impresión por políticas de las empresas que las distribuyen.

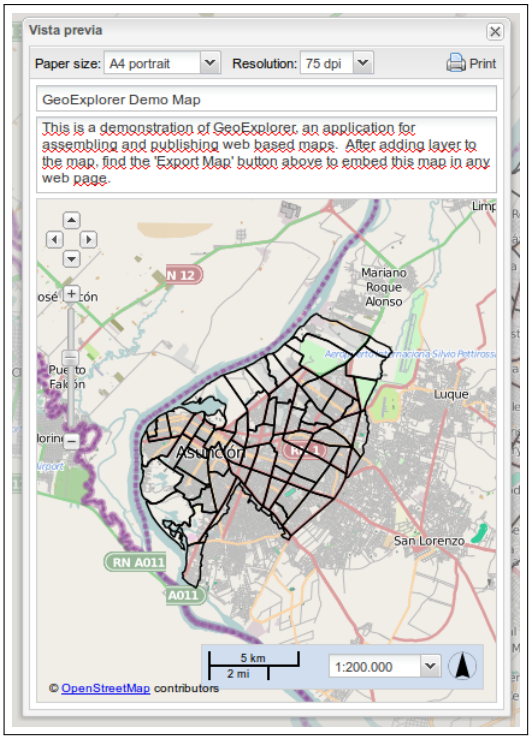


Figura 5.12: Ventana con el cuadro de diálogo de impresión.

Identificador de elementos. La interfaz también provee una funcionalidad para identificar los elementos de las capas, se realiza con el botón *identificador* del menú de herramientas (Figura 5.4c). Al activarlo el usuario debe hacer click en el mapa donde se encuentre el elemento de la capa que quiera identificar. El identificador proporciona la información contenida en los campos de la tabla asociada a la capa, lo muestra en una ventana emergente similar a la ventana de edición de datos (Figura 5.13).

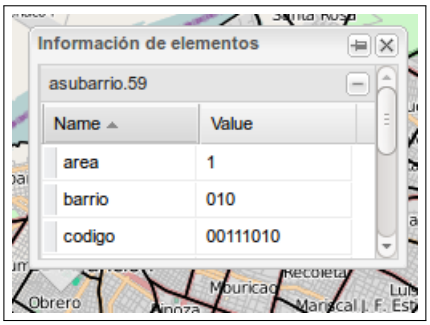


Figura 5.13: Ventana información del elemento seleccionado.

Selección de mapas base. En la sección capas base del panel de capas, se listan los mapas de referencia, de los distintos servidores públicos, que tienen a disposición los usuarios para dar un contexto espacial a sus propias

capas. Al seleccionar alguna de ellas cambia el contexto del mapa de acuerdo al mapa base que se ha seleccionado, como se observa en la Figura 5.14.

Figura 5.14: Selección del mapa base.

Consultar atributos. El usuario tiene la posibilidad de realizar consultas a los atributos que caracterizan los elementos de las capas. Actúa en forma de filtro al consultar por los elementos que reúnan la o las características solicitadas. Se activa por medio del botón *consultar* que se encuentra en el menú de herramientas (Figura 5.4d).

Una vez activado se desprende un panel secundario en la parte inferior de la interfaz como muestra la Figura 5.15, en donde el usuario por medio de consultas lógicas obtiene el conjunto de elementos de la capa seleccionada, que cumplen la condición de la consulta. Estos elementos son visualizados en una grilla en el mismo panel inferior de la consulta.

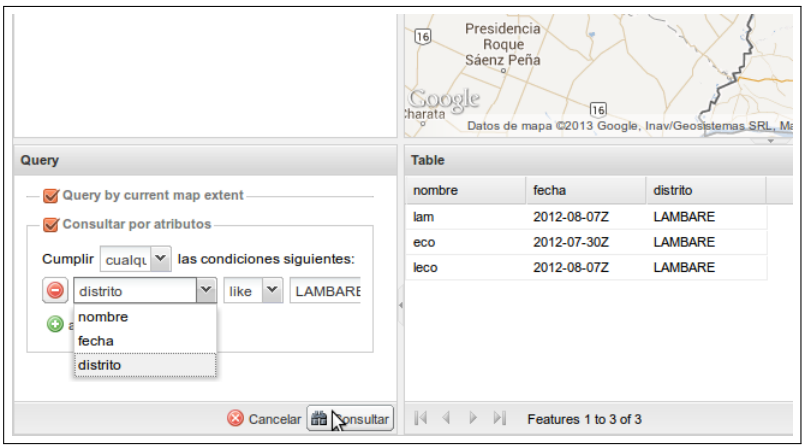


Figura 5.15: Panel inferior con opciones de consultas a los atributos.

Medición de distancias. Otra funcionalidad provista por la interfaz es la herramienta de medición de distancias espaciales. Permite medir distancias de trayectos o de áreas según la opción que se elija. Se activa por medio del botón *medir* del menú de herramientas (Figura 5.4e).

Para la opción de medir trayectos se elige la opción *longitud* del botón *medir* y luego se va trazando el trayecto haciendo clicks en cada cambio de dirección como se muestra en la Figura 5.16. La opción *área* del botón *medir* es la que activa la función para poder dibujar un polígono que contenga la forma del área que se desea medir como se muestra en la Figura 5.17.

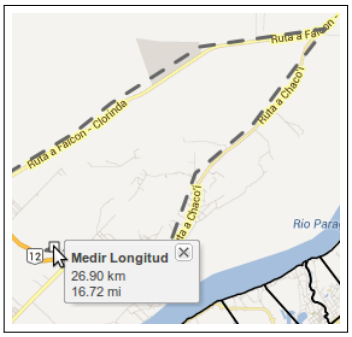


Figura 5.16: Medición de longitud del trayecto en el mapa.

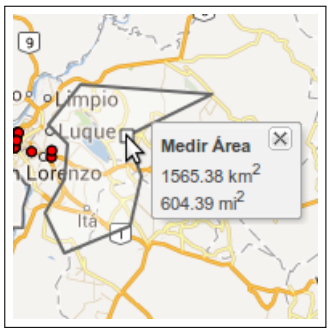


Figura 5.17: Medición de un área en el mapa.

Navegación en el mapa. Dentro del panel del mapa el usuario puede desplazarse por medio de los controles de navegación que aparecen en el mismo panel. Estos controles se encuentran en el extremo superior izquierdo del panel del mapa, por debajo del menú de herramientas como muestra la Figura 5.18. El primer botón es el de *desplazamiento*, al estar activo, el usuario puede desplazarse dentro del mapa de un lugar a otro haciendo click sobre el mapa, y sin soltarlo, dirigirlo en la orientación deseada. El siguiente botón es el *zoom de área*, permite dibujar un rectángulo al cual se ajusta la escala de la vista.

Figura 5.18: Controles de navegación del mapa.

Figura 5.19: Controles del zoom.

El tercer botón *acercar*, disminuye la escala de la vista del mapa para ver en mayor detalle. El cuarto botón *alejar*, aumenta la escala de vista de manera proporcionada para ver información contextual. Estas funciones de desplazamiento y zoom, se encuentran también en los botones de la Figura 5.19.

El quinto botón es el de *vista anterior*, permite volver a la vista anterior inmediata que haya tenido el usuario. De igual modo, el sexto botón *vista siguiente*, devuelve a la última vista disponible. El séptimo y último botón es el de *vista extendida*, que provee una vista panorámica y contextual de todo el globo.

Cambio de estilos. Esta interfaz del sistema permite además realizar cambios en los estilos de visualización de las capas del usuario. El usuario puede diferenciar el estilo con el que se visualizan las capas del mismo formato geográfico, ya sean puntos, líneas o polígonos. Se accede a esta función pulsando click derecho sobre la capa que se desea editar el estilo, y luego se elige la opción *propiedades de la capa* o la opción *editar estilos* (Figura 5.20).

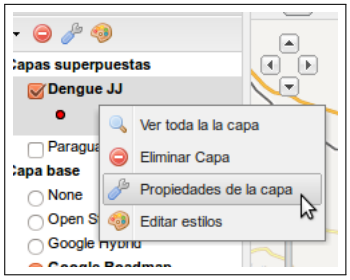


Figura 5.20: Menú contextual propiedades de la capa.

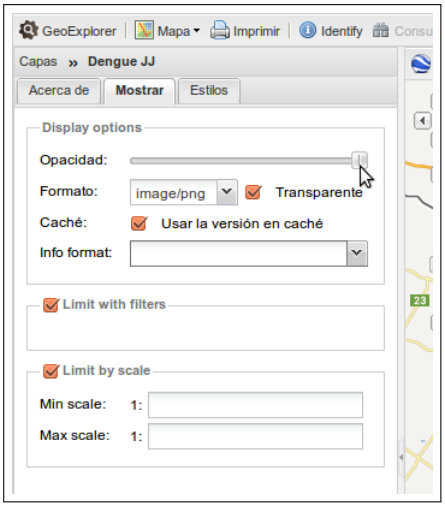


Figura 5.21: Cuadro de diálogo de las propiedades de la capa.

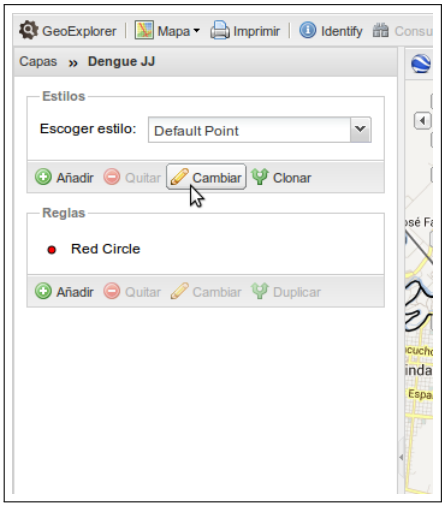


Figura 5.22: Cuadro de diálogo del estilo de la capa.

Al pulsar la opción propiedades de la capa, se abre en el mismo panel de capas el cuadro de diálogo con las propiedades particulares que posee dicha capa como se muestra en la Figura 5.21. Se presenta un cuadro con tres pestañas, una pestaña *acerca de* con información sobre el título, el nombre y una breve descripción sobre la capa; otra pestaña *mostrar*

donde se indican ajustes de opacidad, formato, transparencia, caché de memoria, y limitaciones de filtro o escala; y una tercer pestaña de *estilos* que se muestra en la Figura 5.22.

El cuadro de diálogo del estilo que se muestra en las propiedades de la capa, es el mismo cuadro al que se accede por medio de la opción *editar estilos*, mencionada anteriormente. Desde este cuadro de diálogo el usuario puede diferenciar los colores, tamaños, contornos o rellenos de la imagen que representa la capa.

Añadir servidores públicos de capas. Una funcionalidad más que proporciona la interfaz GeoExplorer, es la de agregar otros servidores externos públicos que proveen capas por medio de protocolos WMS.

Si el usuario conoce los URI de los servidores, lo puede agregar a través de la acción de *añadir capa*. Luego, en lugar de seleccionar el servidor *GeoServer Local* de la Figura 5.6b, se selecciona la opción *añadir servidor* que aparece al final de la lista desplegable de servidores.

A continuación se mostrará en el panel de capas un cuadro de diálogo donde se solicitará la ruta de acceso URL/URI del servidor externo como se muestra en la Figura 5.23.

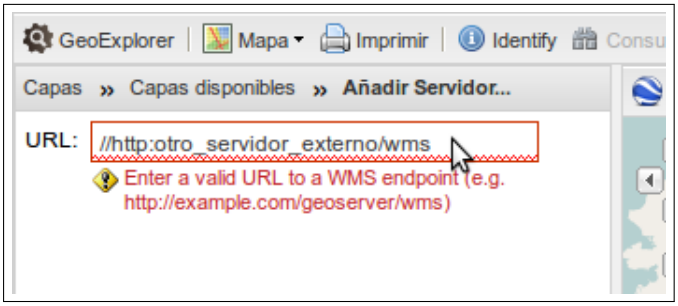


Figura 5.23: Cuadro de diálogo para agregar servidor externo público WMS.

Capítulo 6

Conclusiones

En respuesta a las necesidades de los servicios de salud, y dentro del marco del convenio del Ministerio de Salud de la Nación con la CONAE para generar el sistema HAP, se aprovechó la potencialidad que brindan los sistemas modulares para insertar a la arquitectura existente del sistema SAT/ERDNU, el módulo GIAEST.

Continuar con la construcción de los módulos del *Health Application Project* (HAP), contribuye en gran medida a su propósito principal de brindar sistemas operativos que utilicen herramientas geoespaciales para el soporte de los planes estratégicos de programas nacionales de control vectorial (dengue, leishmaniosis, malaria y chagas) en la Argentina (Scavuzzo and Torrusio, 2010).

Con este trabajo de tesis fue posible desarrollar un módulo geomático de acceso web para la integración y el análisis espacio-temporal de casos de distintas enfermedades en brotes epidémicos, logrando de esta manera cumplir con el objetivo general planteado. Este desarrollo permite el acceso a la información entomológica y virológica desde puntos remotos que cuenten con servicios de internet. La generación de este módulo pretende contribuir a los sistemas de información para vigilancia epidemiológica y servir de herramienta de apoyo para la toma de decisiones en *salas de situación*.

Desde el punto de vista de la vigilancia de la salud, el GIAEST permite la carga simultánea de información precisa en términos geoespaciales, distribuyendo la notificación en tiempos de brotes. Esto posibilita el seguimiento del evento epidémico a medida que va desarrollándose en el transcurso del tiempo, lo cual es un valioso aporte durante los análisis *in situ* que se realicen en las *salas de situación* de los puestos de vigilancia.

Cubriendo con los requerimientos de los programas de salud de contar con sistemas de información ágiles, que permitan identificar áreas y poblaciones con mayores necesidades de salud, se logró desarrollar una plataforma web con funcionalidades para sistemas de información geográfica (SIG). La misma permite independizar la carga de datos realizada por diversos usuarios y las unifica en una base de datos central, con un ambiente intuitivo a través de interfaces gráficas. De esta forma se posibilita focalizar las intervenciones hacia los grupos más prioritarios (WHO-PAHO/AIS, 2003).

Cabe resaltar la importancia para los servicios de salud, de contar con herramientas SIG como la desarrollada, dirigidas a usuarios no especializados, y que proporcionen aplicaciones prácticas en el uso cotidiano y garanticen llegar a resultados eficientes que posteriormente contribuirán a la toma de decisiones. Esto además se presenta como un beneficio para los servicios de salud, por no requerir de mayores costos económicos y de tiempo en la capacitación de sus usuarios.

Se cumplieron tanto los objetivos generales planteados como los objetivos específicos, destacando los siguientes logros:

- Se definió un diseño apropiado para el sistema que estuvo basado tanto en los requerimientos del sistema general, como en los requerimientos particulares respecto al módulo.
- Se especificó, construyó y documentó la arquitectura e interfaces del módulo GIAEST en el desarrollo de este trabajo, con tecnologías de código abierto. Los cuales continúan aportando flexibilidad y reusabilidad al sistema general.
- Mediante la implementación del test de Knox, se logró cumplir con el objetivo específico de detectar agrupaciones espacio-temporales de casos en brotes epidémicos. Por lo tanto, aporta a la identificación de riesgos relativos de nuevos casos, basados éstos en términos de espacio y tiempo, entre otros.

Una vez más se pudo verificar como la versatilidad que proporciona la programación orientada a objetos, sumado a los aportes que brindan los software de código abierto o bajo licencias GNU, posibilitan la articulación de pequeños componentes independientes de software para construir soluciones mucho más completas.

6.1 Aportes y lineamientos futuros

Este trabajo de tesis amplió mis conocimientos técnicos en diseño de software; especialmente sobre aplicaciones de arquitectura cliente-servidor para sistemas

de información geográfica.

Se lega un aporte de índole informático, que contribuye con la comunidad del proyecto de software estadístico R, con la implementación de un paquete que contiene el análisis espacio-temporal del test de Knox propuesto por los autores Tran et al. (2004), para distribuirlo a la comunidad como complemento de software independiente.

Respecto a lineamientos futuros, se observa la oportunidad que brindan la modularidad de software para proporcionar nuevas herramientas de análisis para su uso en los sistemas de salud. Éstos pueden ser implementados sin mayores inconvenientes bajo tecnologías de código abierto y licencias GNU.

Por otro lado, se observa la necesidad de continuar aportando al análisis espacio-temporal pero delimitando geográficamente las agrupaciones de casos. El análisis del test de Knox implementado solo genera un gráfico de frecuencias resultante por las agrupaciones, aunque no la ubicación geográfica de las mismas.

En relación al consumo de recursos y tiempo, cabe aclarar que el análisis espacio-temporal implementado incrementa considerablemente el tiempo de generación de resultados a medida que aumenta la cantidad de casos y de réplicas de eventos aleatorios a comparar. Debido a esto, un lineamiento a futuro es poder implementar otros tipos de test estadísticos que aporten resultados de similar beneficio epidemiologíco con menor costo en términos de recursos y tiempos. Otra opción a considerar para abordar este problema es pensar en una infraestructura de "computadoras de alto desempeño". Esta disciplina, permite el desarrollo de algoritmos de procesamiento paralelo y programas de software que pueden ser divididos en pequeñas piezas, de modo tal que cada pieza puede ser ejecutada simultáneamente por procesadores separados. Estas técnicas aprovechan al máximo los recursos de una computadora, optimizando ampliamente el tiempo de ejecución.

Por último, una interesante línea de desarrollo futura sería generar las interfaces apropiadas para estandarizar las entradas y salidas de nuestro algoritmo estadístico de modo tal de convertir el test de Knox en un servicio de procesamiento geoespacial. Convertir nuestro algoritmo en un servicio implica que esta funcionalidad podrá ser invocada (ejecutada) desde cualquier cliente que soporte el servicio web WPS (Web Processing Service).

Referencias

Referencias

Agency, E. S. (2011). Basics guides and reports. (accessed on July 2012).

Baker, R. (1996). Testing for space-time clusters of unknown size. *Journal of Applied Statistics*, 23:543–554.

Barragán, H. L., Moiso, A., Mestorino, M., and Ojeda, O. A. (2007). Fundamentos de la Salud Pública. Technical report, Universidad Nacional de La Plata.

Barton, D. E. and David, F. N. (1966). The Random Intersection of Two Graphs. *Research Papers in Statistics*, pages 455–59.

Bass, L., Clements, P., and Kazman, R. (2003). *Software Architecture in Practice.* Addison Wesley, 2nd edition.

Beaglehole, R. and Bonita, R. (2004). *Public health at the crossroads: achievements and prospects*. Cambridge University Press.

Berners-Lee, T., Fielding, R., and Masinter, L. (2005). Uniform Resource Identifier (URI): Generic Syntax. Technical report, The Internet Engeneering Task Force (IETF).

Bonita, R., Beaglehole, R., and Kjellström, T. (2006). *Basic Epidemiology*. World Health Organization, 2nd edition.

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

Castillo-Salgado, C. (1993). Estratificación epidemiológica de la malaria en la región de las Américas. *Mem Inst Oswaldo Cruz*.

Chin, J. (2001). El control de las enfermedades transmisibles. Technical Report 581, Organización Panamericana de la Salud.

Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Merson, P., Nord, R., and Stafford, J. (2010). *Documenting Software Architectures: Views and Beyond.* Addison Wesley, 2nd edition.

Consortium, O. G. (2005). OpenGIS Location Services (OpenLS): Core Services, OGC 05-016. Technical report, Open Geospatial Consortium.

Consortium, O. G. (2006). OpenGIS Web Map Server Implementation Specification. Technical Report OGC® 06-042, Open Geospatial Consortium.

Consortium, O. G. (2008). OWS5: OGC Web feature service, core and extensions. Technical report, Open Geospatial Consortium.

Contol, C. f. D. and Prevention (1992). *Principles of Epidemiology*. Public Health Practice Program Office, Atlanta, Georgia.

de la Nación, M. d. S. (2007). *Manual de normas y procedimientos de vigilancia y control de enfermedades de notificación obligatoria. Revisión nacional 2007.*

de México, S. d. S. (2001). *Programa de Acción: Enfermedades Transmitidas por Vectores*. 1ra. edition.

Denver, A. G. E. (1991). *Epidemiología y administración de servicios de salud.* Organización Panamericana de la Salud.

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

IEEE (1998). IEEE recommended practice for software requirements specification. Technical Report 830-1998, IEEE.

IEEE (2000). IEEE recommended practice for architectural description of software-intensive systems. Technical Report 1471-2000, IEEE.

Jalote, P. (2008). *A Concise Introduction to Software Engineering*. Springer-Verlag.

Johansen, P., Brody, H., Rachman, S., and Rip, M. (2003). *Cholera, Cholorform, and the Science of Medicine: a life of John Snow*. Oxford University Press.

Kaufman, L. and Rousseeuw, P. J. (1990). *Finding Groups in Data. An Introduction to Cluster Analysis.* John Wiley & Sons.

Knox, E. G. (1964). The detection of Space-Time interaction. *Journal of the Royal Statistical Society.*, 13(1):25–30.

Kulldorff, M. and Hjalmars, U. (1999). The Knox Method and Other Tests for Space-Time Interaction. *Biometrics*, 55(2):544–552.

Lalonde, M. (1981). A new Perspective on the Health of the Canadians. Technical report, Government of Canada.

Last, J. (2001). *A dictionary of epidemiology.* Oxford University Press, 4th edition.

Lemus, J. D., Tigre, C. H., Ruíz, P. L., and Dachs, N. (1996). *Manual de vigilancia epidemiólogica*.

Mantel, N. (1967). The Detection of Disease Clustering and a Generalized Regression Approach. *Cancer Research*, 27(2):209–220.

McGwire, K., Segura, E., Scavuzzo, M., Gomez, A., and Lanfri, M. (2006). Spatial pattern of reinfestation by Triatoma infestans in Chancaní, Argentina. *Journal of Vector Ecology*, 31(1):17–28.

Morasca, S., Tiaibe, D., and Tosi, D. (2009). Towards certifying the testing process of Open-Source Software: New challenges or old methodologies? *FLOSS '09 Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development*, pages 25–30.

Mordani, R. (2009). Java Servlet Specification, Version 3.0. Technical report, Sun Microsystems, Inc.

MSAL and OPS-OMS (2012). Indicadores Básicos 2012. Technical report.

OpenGeo, O. (2011). The OpenGeo Architecture.

OPS, O. P. d. l. S. (2012). *Salud en las Américas.* Organización Panamericana de la Salud, 15 edition.

Oracle, C. (2010). The Java EE 5 Tutorial, For Sun Java System Application Server 9.1 . Technical report, Oracle Corporation.

Peng, Z. and Tsou, M. (2003). *Internet GIS: distributed geographic information services for the Internet and wireless networks.*, volume 36. John Wiley & Sons, 1th edition.

Peralta, G. (2011). Geomática aplicada a un Sistema de Alerta Temprana. Master's thesis, Instituto de Altos Estudios Espaciales Mario Gulich.

Porcasi, X., Calderón, G., Lanfri, M., Scavuzzo, M., Sabattini, M., and Polop, J. (2005). Predictive distribution maps of rodent reservoir species of zoonoses in South America. *Mastozoología Neotropical*.

Porcasi, X. and Intrioni, M. V. (2011). SAT/ERDNU Requirements Baseline Document (RBD). Technical report, Comisión Nacional de Actividades Espaciales.

Porcasi, X., Rotela, C., Introini, M., Frutos, N., Lanfri, S., Peralta, G., De Elia, E., Lanfri, M. A., and Scavuzzo, C. (2012). An operative dengue risk stratification system in Argentina based on geospatial technology. *Geospatial Health.*, 6(3):31–42.

R Development Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Rotela, C., Fouque, F., Lanfri, M., Sabatier, P., Intrioni, M. V., Zaidenberg, M., and Scavuzzo, M. (2007). Space-time analysis of the dengue spreading dynamics in the 2004 Tartagal outbreak, Northern Argentina. *Elsevier B.V.*, pages 1–13.

Rotela, C., Spinsanti, L., Lanfri, M., Contigiani, M., Almiron, W., and Scavuzzo, M. (2011). Mapping environmental susceptibility to Saint Louis encephalitis virus, based on a decision tree model of remotely sensed data. *Geospatial Health*, 6(1):85–94.

Scavuzzo, 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. In *E-Health. Proceedings of Med-e-Tel.* The International Trade Event and Conference for Health, Telemedicine and Health ICT.

Scavuzzo, M. and Torrusio, S. (2010). Project Management Plan for HAP. Technical report, Comisión Nacional de Actividades Espaciales.

Shekhar, S. and Xiong, H. (2008). *Encyclopedia of GIS.* Springer Science+Buisiness Media, LLC, 233 Spring Street, New York, NY 10013, USA.

Shuang, L. and Cheng, P. (2009). Developing JAVAEE applications based on utilizing design patterns. *WASE International Conference on Information Engineering. ICIE*, pages 398–401.

Snow, J. (1855). On the Mode of Communication of Cholera. *London*, 8.

Suárez, L., Carmen, L., and Berdasquera, C. (2000). Enfermedades Emergentes y Reemergentes: Factores Causales y Vigilancia. *Revista Cubana de Medicina General Integral*, 16(6):593–597.

Thacker, S. and Berkelman, R. (1988). Public health surveillance in the United States. *Epidemiol Rev.*, pages 165–190.

Tolcachier, A. J. (2010). *Salud Ambiental*, chapter 10. IntraMed & Laboratorios Roemmers.

Tourre, Y., Jarlam, L., Lacaux, J., Rotela, C., and Layafe, M. (2008). Spatio-temporal variability of NDVI–precipitation over southernmost South America: possible linkages between climate signals and epidemics. *IOP Science*, pages 98–108.

Tran, A., Deparis, X., Dussart, P., Morvan, J., Rabarison, P., Remy, F., Polidori, L., and Gardon, J. (2004). Dengue Spatial and Temporal Patterns, French Guiana, 2001. *Emerging Infectious Diseases*, 10(4):615–621.

W3C (2008). Extensible Markup Language (XML) 1.0 (Fifth Edition). Technical report, W3C Working Group.

WHO-PAHO/AIS (2003). SIGEPI v1.26 - Manual de Usuario. Technical Report 26, Organización Panamericana de la Salud.

Williams, R. (1987). Selling a geographical information system to government policy makers. *URISA*, pages 150–156.

Apéndice A

Apéndice A

A.1 Fuente del Servlet

A continuación se describe el código fuente desarrollado de la clase $Java$ que cumple la función de servlet residente en el servidor. El mismo desencadena la ejecución del algoritmo del análisis espacio-temporal del test de Knox.

Código clase ServletKnox

```
1  package paq;
2
3
4  import java.io.IOException;
5  import java.io.PrintWriter;
6  import javax.servlet.ServletException;
7  import javax.servlet.http.HttpServlet;
8  import javax.servlet.http.HttpServletRequest;
9  import javax.servlet.http.HttpServletResponse;
10
11 import com.google.gson.JsonObject;
12
13
14
15 @SuppressWarnings("serial")
16 public class ServeltKnox extends HttpServlet {
17
18
19 public ServeltKnox(){
20 super();
21 }
22
```

```
23
24 protected void doGet(HttpServletRequest request, ...
       HttpServletResponse response) throws ServletException, ...
       IOException {
25 doPost(request,response);
26 }
27
28
29 protected void doPost(HttpServletRequest request, ...
       HttpServletResponse response) throws ServletException, ...
       IOException {
30
31
32 PrintWriter out = response.getWriter();
33 response.setContentType("text/html");
34 response.setHeader("Cache-control", "no-cache, no-store");
35 response.setHeader("Pragma", "no-cache");
36 response.setHeader("Expires", "-1");
37 response.setHeader("Access-Control-Allow-Origin", "*");
38 response.setHeader("Access-Control-Allow-Methods", "POST,GET");
39 response.setHeader("Access-Control-Allow-Headers", "Content-...
       Type");
40 response.setHeader("Access-Control-Max-Age", "86400");
41
42 String ciudad = request.getParameter("ciudad");
43 String programa = request.getParameter("programa");
44 String tabla = request.getParameter("tabla");
45 String fechaini = request.getParameter("startdt");
46 String fechafin = request.getParameter("enddt");
47 String rep = request.getParameter("rep");
48
49
50 try {
51         KnoxR.knox(tabla, ciudad, fechaini, fechafin, rep);
52 } catch (Exception e) {
53         // TODO Auto-generated catch block
54         e.printStackTrace();
55 }
56
57 System.out.println(ciudad);
58 System.out.println(programa);
59
60 JsonObject myObj = new JsonObject();
61
62 myObj.addProperty("success", true);
63 myObj.addProperty("message", "La capa -"+programa+"- -"+ciudad+...
       "-" + "- ini:"+fechaini+"- fin:"+fechafin+"- tab:"+tabla+"...
       - rep:"+rep+"- se carg  correctamente.");
64 out.println(myObj.toString());
65 out.close();
66
67 }
68 }
```

A.2 Fuente conexión a R

A continuación se describe el código fuente desarrollado de la clase $Java$ que recepciona los parámetros desde el servlet, realiza la apertura de conexión a la base de datos, y reenvía los parámetros al R para ejecución del algoritmo del test de Knox y su respectiva recepción de resultados.

Código clase KnoxR

```
1  package paq;
2
3  import java.awt.image.BufferedImage;
4  import java.io.File;
5  import java.util.ArrayList;
6  import java.util.Calendar;
7  import javax.imageio.ImageIO;
8  import rcaller.RCaller;
9  import rcaller.RCode;
10
11 public class KnoxR {
12         static String res = "";
13         public static String knox(String tab, String ciudad, ...
               String fdesde, String fhasta, String rep) throws ...
               Exception {
14
15                 DataBase conn = new DataBase();
16                 System.out.println("se conecto");
17                 System.out.println("fechaini: "+fdesde);
18                 System.out.println("fechafin: "+fhasta);
19
20                 //string del SELECT de la tabla
21                 ArrayList<Object> argu = conn.dataBase(tab, ...
                       ciudad, fdesde, fhasta);
22
23                 if (argu.isEmpty()) {
24                         return "La fecha es inexistente para ...
                               los intervalos ingresados";
25                 } else {
26                 //Argumentos para pasarle a la funcion knoxtest...
                       .
27                     argu.add(rep);
28
29                     runKnox(argu);
30                     conn.disconnect();
31                     res=", al culiminar enviaremos los ...
                           resultados a su e-mail.";
32                     return "Ok "+tab+" "+rep+" - "+res;
33                 }
34         }
35
36
37
```

```
38          public static void runKnox(ArrayList<Object> argu) ...
                throws Exception{
39            try {
40              RCaller caller = new RCaller();
41              caller.setRscriptExecutable("/usr/bin/Rscript");
42              caller.cleanRCode();
43
44              RCode code = new RCode();
45              code.clear();
46
47              int f = Integer.parseInt(argu.get(0).toString());
48              int x = argu.subList(1, f+1).toString().length();
49              int y = argu.subList(f+1, f*2+1).toString().length...
                    ();
50              int t = argu.subList(f*2+1,f*3+1).toString().length...
                    ();
51
52              code.R_require("knox");
53
54              code.addRCode("tab.data<-data.frame(xdata=c("+argu....
                    subList(1, f+1).toString().substring(1, x-1)+"...
                    ),"+
55                  "ydata=c("+argu.subList(f+1, f*2+1).toString()....
                        substring(1, y-1)+"),"+
56                  "tdata=c("+argu.subList(f*2+1,f*3+1).toString()...
                        .substring(1, t-1)+"))");
57              code.addRCode("tknox<-knoxtest(tab.data,"+argu.get(...
                    f*3+1)+","+argu.get(f*3+2)+")");
58
59              caller.setRCode(code);
60              caller.runOnly();
61
62              File img = new File("/tmp/knox-acumulado.png");
63              File img1= new File("/tmp/knox-real.png");
64
65              BufferedImage out = ImageIO.read(img);
66              ImageIO.write(out, "png",new File("/ruta/de/...
                    almacenamiento/knox-acumulado.png"));
67
68              out = ImageIO.read(img1);
69              ImageIO.write(out, "png",new File("/ruta/de/...
                    almacenamiento/knox-real.png"));
70
71            } catch (Exception e) {
72              // TODO: handle exception
73              System.out.println(e.toString());
74            }
75          }
76 }
```

A.3 Fuente conexión a base de datos

A continuación se describe el código fuente desarrollado de la clase $Java$ que realiza la apertura y cierre de conexión a la base de datos, realiza la consulta y reenvía el resultado de los datos necesarios para la ejecución del algoritmo del test de Knox.

Código clase DataBase

```
1  package paq;
2
3 import java.sql.Connection;
4 import java.sql.DatabaseMetaData;
5 import java.sql.DriverManager;
6 import java.sql.ResultSet;
7 import java.sql.SQLException;
8 import java.sql.Statement;
9 import java.util.ArrayList;
10
11 public class DataBase {
12   /**
13   * @param args
14   */
15   static Connection db; // A connection to the database
16   static Statement sql; // Our statement to run queries wit
17   static DatabaseMetaData dbmd; // This is basically info the ...
       driver delivers
18   static ResultSet rs;
19   static String database = "dbname";
20   static String username = "admin";
21   static String password = "pass";
22
23
24   public void connect(){
25     try{
26       Class.forName("org.postgresql.Driver");
27       db = DriverManager.getConnection("jdbc:postgresql://...
           localhost:5432/"+database, username, password); //...
           connect to the db
28
29       dbmd = db.getMetaData(); //get MetaData to confirm ...
           connection
30       System.out.println("Connection to "+dbmd.....
           getDatabaseProductName()+" "+dbmd.....
           getDatabaseProductVersion()+" successful.\n");
31
32       sql = db.createStatement(); //create a statement that we ...
           can use later
33
34     }catch(Exception e){
35         e.printStackTrace();
36     }
37   }
```

```
38   public ArrayList<Object> dataBase(String tabla, String ciudad...
       , String desde, String hasta) throws SQLException{
39         System.out.println("antes de conectarse");
40         connect();
41         ArrayList<Object> salida = new ArrayList<Object>();
42         System.out.println("usuario conectado correctamente");
43
44         //hacemos un select de esa tabla
45         String sqlSelect = "SELECT SUBSTR(ASTEXT(the_geom)...
               ,7,6) AS X,"+
46                         " SUBSTR(ASTEXT(the_geom),POSITION(' ' IN...
                              ASTEXT(the_geom))+1,7),"+
47                         " TO_CHAR(fecha,'J') FROM "+tabla+
48                         " WHERE distrito = '"+ciudad.toUpperCase()...
                             +"'"+
49                              " AND (fecha BETWEEN '"+desde+"'::...
                                   DATE AND '"+hasta+"'::DATE);";
50         System.out.println("string cargado "+sqlSelect);
51
52         rs = sql.executeQuery(sqlSelect);
53
54         System.out.println("ejecuto el query");
55
56         if (rs.wasNull()) {
57                 return salida;
58         } else {
59                 ArrayList<String> x = new ArrayList<String>();
60                 ArrayList<String> y = new ArrayList<String>();
61                 ArrayList<String> t = new ArrayList<String>();
62                 int row=0;
63
64                 while (rs.next()){
65                                 x.add(rs.getString(1));
66                                 y.add(rs.getString(2));
67                                 t.add(rs.getString(3));
68                                 row++;
69                 }
70                 salida.add(row);
71                 salida.addAll(x);
72                 salida.addAll(y);
73                 salida.addAll(t);
74
75                 return salida;
76         }
77   }
78
79   public void disconnect(){
80     try{
81         db.close();
82     }catch(Exception e){
83         e.printStackTrace();
84     }
85   }
86 }
```

A.4 Fuente algoritmo Test de Knox

A continuación se describe el código fuente desarrollado en lenguaje R del algoritmo del test de Knox. Recepciona los parámetros desde $Java$, analiza los los datos del escenario epidémico real, genera los escenarios aleatorios para compararlos con el real y genera los gráficos resultantes de los análisis.

Código del algoritmo Test de Knox

```
################## Algoritmo de KNOX 0.1 ######################
#
# Indicar el archivo tab.data, que debe tener solo 3
# columnas, en la 1ra la coordenada X, la 2da la coordenada
# Y y en la ultima la columna del tiempo (dia que sucedio
# el caso).
#
# Configurar ademas las rutas de salidas de los archivos
# PNG donde correspondan.
#
######## Dependencias (Paquetes R requeridos) ##############
#  - "gplots" (Histograma 2D)
#  - "Runiversal" (instalar para correrlo desde Java)
#############################################################



knoxtest <- function (tab.data, rep, nbin=NULL, incid=NULL){

attach(tab.data)


### Variables ###
repli <- rep                  # Nro. de replicas
n     <- dim(tab.data)[1]     # Nro. Total de casos

m_espacio <- matrix(-1,n,n)  # Dimensionar matriz dist. ...
    Espacial
m_tiempo <- m_espacio        # Dimensionar matriz dist. ...
    Temporal


## Calculo dist. Espacial y Temporal, elemento a elemento ##
for(i in 1:n){
  for(j in 1:n){
      m_espacio[i,j] <- round(sqrt((xdata[i]-xdata[j])^2 + (...
          ydata[i]-ydata[j])^2))
      m_tiempo[i,j] <- round(abs(tdata[i]-tdata[j]))
  }
}


```

```
41 ### Los bins para el Hist2D son 1/3 de "n" por default ####
42 bins <- round(n/3)
43 if(length(nbin) > 0) bins <- nbin
44
45 require('gplots')
46
47
48 #######  Calculo Frecuencia 2D Real (Histograma 2D) ########
49 m_hist2d <- hist2d(m_tiempo,m_espacio,nbins=c(bins,bins),show=F...
       ) #Histograma 2D Real
50
51 histreal <- as.matrix(m_hist2d$counts) #tomar datos hist2d Real
52 lc<-length(colnames(histreal))
53 lr<-length(rownames(histreal))
54
55
56 ### Se ajustan los parametros para generar la imagen ###
57 ramp.colors <- colorRampPalette(c(gray(0.15),"#00007F", "blue",...
       "#007FFF", "cyan", "#7FFF7F", "yellow", "#FF7F00", "red"))
58 x <- seq(from=min(m_tiempo),to=max(m_tiempo),length.out=nrow(...
       histreal))  #Fracciona X
59 y <- seq(from=min(m_espacio),to=max(m_espacio),length.out=ncol(...
       histreal))  #Fracciona Y
60
61
62 ### Guardar la imagen ###
63 png(filename='/tmp/knox-tart-real-prueba.png')
64 filled.contour(x, y, histreal, color = ramp.colors,
65                plot.title = title(main = "Interacci n Real\...
                       nEspacio-Tiempo", xlab = "Tiempo (d as)", ...
                       ylab = "Distancia (m)"),
66                plot.axes = { axis(1,yaxp=c(min(x),max(x),5))
67                              axis(2,xaxp=c(min(y),max(y),5))},
68                key.title = title(main=" Intensidad", cex.main...
                       =.9),
69                key.axes = axis(4, seq(min(histreal), max(...
                       histreal),by=round(max(histreal)/4))))
70 mtext(paste("Instituto M. Gulich, CONAE - ",format(Sys.Date(),...
       '%d/%m/%Y')), side = 1, line = 4, adj = 1, cex = .9)
71 dev.off()
72
73
74 ### Generacion Matrices Aleatorias ###
75 histrand   <- array(,c(lr,lc,repli))
76 m_iespacio <- m_espacio
77 m_itiempo  <- m_tiempo
78
79
80 ### Cargar las matrices con valores aleatorios ###
81 for(k in 1:repli){
82   for(i in 1:n){
83     ix <- runif(n)      #random uniform x
84     iy <- runif(n)      #random uniform y
85     it <- runif(n)      #random uniform t
```

```
86
87      itdata <- round(it*(max(tdata)-min(tdata))+min(tdata))...
                #normalizacion del valor aleatorio
88      ixdata <- round(ix*(max(xdata)-min(xdata))+min(xdata))...
                #normalizacion del valor aleatorio
89      iydata <- round(iy*(max(ydata)-min(ydata))+min(ydata))...
                #normalizacion del valor aleatorio
90
91      for(j in 1:n){
92        m_iespacio[i,j] <- round(sqrt((ixdata[i]-ixdata[j])^2 + (...
              iydata[i]-iydata[j])^2))
93        m_itiempo[i,j] <- round(abs(itdata[i]-itdata[j]))
94      }
95    }
96    print(c("Generacion Aleatoria: ",k))
97
98    ####  Frecuencia 2D Aleatoria (Histograma 2D) ####
99    h2drand <- hist2d(m_itiempo,m_iespacio,nbins=c(bins,bins),...
          show=F)
100
101   histrand[,,k] <- as.matrix(h2drand$counts) #tomar datos ...
          aleatorios
102 }
103
104
105 ###### Calculo de KNOX #####
106 m_knox <- array(0,c(lr,lc))
107 acum <- array(0,c(lr,lc))
108
109 for(k in 1:repli){
110   for(i in 1:lr){
111     for(j in 1:lc){
112       if(histreal[i,j] > histrand[i,j,k]){
113         acum[i,j]<-acum[i,j]+1
114       }
115     }
116   }
117   print(c('acumulado ',k))
118   m_knox <- acum/(repli)
119 }
120
121
122 #### Generacion de imagen KNOX ####
123 x <- seq(from=min(m_tiempo),to=max(m_tiempo),length.out=nrow(...
      m_knox))  #Fracciona X
124 y <- seq(from=min(m_espacio),to=max(m_espacio),length.out=ncol(...
      m_knox))  #Fracciona Y
125
126 png(filename='/tmp/knox-acumulado-tart-prueba.png')
127 filled.contour(x, y, m_knox, color = ramp.colors,
128                plot.title = title(main = "Interacci n Knox\...
                       nEspacio-Tiempo", xlab = "Tiempo (d as)", ...
                       ylab = "Distancia (m)"),
129                plot.axes = { axis(1,yaxp=c(min(x),max(x),5))
```

```
130                                 axis(2,xaxp=c(min(y),max(y),5))},
131                 key.title = title(main=" Intensidad", cex.main...
                        =.9),
132                 key.axes = axis(4, seq(min(m_knox), max(m_knox),...
                        by=.2)))
133 mtext(paste("Instituto M. Gulich, CONAE - ",format(Sys.Date(),...
        '%d/%m/%Y')), side = 1, line = 4, adj = 1, cex = .9)
134 dev.off()
135
136 results <- list(space.dist=m_espacio,time.dist=m_tiempo,space....
        time=histreal,knox=m_knox)
137
138 print('Results returns a "list" object containing:')
139 print(summary(results))
140 return(results)
141 }
```

Apéndice B

Apéndice B

B.1 Perfiles de usuarios y funcionalidades del sistema

En la presente tabla se representa con el símbolo "✓", las funcionalidades de la interfaz a las que pueden acceder los distintos perfiles de usuarios. Del mismo modo, el símbolo "–" representa las funcionalidades no accesibles.

		Perfiles			
	Funcionalidades	Visitante	Local	Provincial	Nacional
•	Gestión de información.	–	✓	✓	✓
•	Control de campos y validaciones.	–	✓	✓	✓
•	Visualización de datos en formato raster y vector a diversas escalas.	–	✓	✓	✓
•	Consulta o filtro por atributos.	–	✓	✓	✓
•	Información de atributos.	✓	✓	✓	✓

Tabla B.1: Tabla de perfiles de usuarios y funcionalidades

	Perfiles			
Funcionalidades	Visitante	Local	Provincial	Nacional
• Manipulación de capas.	–	✓	✓	✓
• Manipulación de datos.	–	✓	✓	✓
• Selección de mapas base.	✓	✓	✓	✓
• Navegación en el mapa.	✓	✓	✓	✓
• Medición de distancias.	✓	✓	✓	✓
• Herramientas de navegación.	✓	✓	✓	✓
• Añadir servidores públicos de capas.	✓	✓	✓	✓
• Búsqueda de localidades.	✓	✓	✓	✓
• Cambio de estilos.	–	✓	✓	✓
• Ejecutar análisis espacio- temporal.	–	–	✓	✓
• Delimitación de fechas coherentes	–	–	✓	✓
• Cambio de parámetros de los componentes del análisis.	–	–	✓	✓
• Resultado del análisis.	–	–	✓	✓
• Salida de información en formato PDF.	–	✓	✓	✓
• Impresión de mapas.	✓	✓	✓	✓
• Almacenamiento de datos.	–	✓	✓	✓
• Almacenamiento extra de capas.	–	✓	✓	✓
• Utilizar los datos de MAPEAR.	–	✓	✓	✓

Tabla B.2: Tabla de perfiles de usuarios y funcionalidades