Implementación de una Herramienta Web Orientada al Soporte de Estudios Ambientales

por Edinson Andrés Solarte Casanova

UNIVERSIDAD NACIONAL DE CÓRDOBA

Facultad de Matemática, Astronomía, Física y Computación

Instituto de Altos Estudios Espaciales Mario Gulich.

Córdoba, Argentina

2018

Implementación de una Herramienta Web Orientada al Soporte de Estudios Ambientales

por Edinson Andrés Solarte Casanova

Ingeniero Topográfico

Presentado como parte de los requerimientos para la obtención del título de:

MAGÍSTER EN APLICACIONES ESPACIALES DE ALERTA Y RESPUESTA TEMPRANA A EMERGENCIAS

Director:

Ph.D., David Gorla

UNIVERSIDAD NACIONAL DE CÓRDOBA

Facultad de Matemática, Astronomía, Física y Computación

Instituto de Altos Estudios Espaciales Mario Gulich

Córdoba, Argentina

2018

Esta obra está bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 4.0 Internacional.

Agradecimientos

A mi familia que a pesar de la distancia siempre encuentran la forma de apoyarme y así no sentirme solo en este camino.

A CONAE, el Instituto de Altos Estudios Espaciales “Mario Gulich” y la Universidad Nacional de Córdoba por la oportunidad brindada.

A mi Director David Gorla por su apoyo durante el proceso.

A Pablo Zader quien tuvo la disposición de ayudarme durante el desarrollo de esta tesis.

A Diana Fonnegra por estar ahí siempre para darme su apoyo incondicional.

A mis amigos de la maestría, en especial a Cachi, Juan Pablo y Gianni que han estado y espero que sigan estando en mi camino.

A Felipe, Elena, Pamela y Diana por acogerme y hacerme sentir como parte de su grupo.

A Gabriel que más que un tutor y un docente ha sido un amigo.

A Marcelo que como docente me brindó su confianza y me encaminó en este proyecto.

Y a todos los compañeros y docentes del instituto que de una u otra forma estuvieron dispuestos para contribuir en la culminación de este proceso.

Mil gracias!

Resumen

Los Sistemas de Información Geográfica Web (WebGIS) hacen uso de la capacidad de almacenar y compartir datos por medio de protocolos establecidos por el Open Geospatial Consortium (OGC) para suministrar capas, a diferentes tipos de usuarios, con un procesamiento previo que facilite su utilización y promueva el desarrollo de estudios en diversos campos. Sin embargo, existen diversos formatos y resoluciones en los que esta información es puesta a disposición, lo que supone un inconveniente para aquellas personas que no son especialistas en el procesamiento digital de imágenes. El presente trabajo tiene como objetivo la implementación de un aplicativo web que contenga una base de datos de tipo vectorial y ráster de libre acceso para usuarios interesados en la realización de estudios ambientales en Suramérica. El aplicativo incorpora complementos desarrollados mediante lenguajes de programación como JavaScript y Python, así como *software* SIG de escritorio como GRASS GIS 7.2 para que el usuario tenga la capacidad de obtener información por punto de los datos almacenados, realizar descargas, recortes a las capas ráster, ecuaciones simples entre ellas, regresiones lineales entre dos capas ráster y perfiles temporales de las capas ráster contenidas en servidor para un punto en particular, generando además un archivo en formato txt con los valores de las series. Esto facilita que en algunos casos, las personas puedan manipular la información contenida sin necesidad de realizar descargas y un posterior procesamiento en un *software* SIG de escritorio.

Palabras clave: Datos Geográficos, WebGIS, Estudios Ambientales, Aplicativo Cliente, Servidor de Mapas.

Contenido

Lista de Figuras

Lista de Tablas

Acrónimos

API Application Programming Interface

CGIAR CSI Consortium for Spatial Information

CNPq Consejo Nacional de Desarrollo Científico y Tecnológico

CVC Corporación Autónoma Regional del Valle del Cauca

EOSDIS Earth Observing System Data and Information System

GML Geography Markup Language

GPM Global Precipitation Measurement

KML Keyhole Markup Language

LST Land Surface Temperature

MDE Modelo Digital de Elevación

MODIS Moderate Resolution Imaging Spectroradiometer

NASA National Aeronautics and Space Administration

NDDI Normalized Difference Drought Index

NDVI Normalized Difference Vegetation Index

NDWI Normalized Difference Water Index

OSGeo Open Source Geospatial Foundation

OGC Open Geospatial Consortium

SIG Sistema de Información Geográfica

SMOS Soil Moisture Ocean Salinity

SRTM Shuttle Radar Topography Mission

TOPP The Open Planning Project

WCS Web Coverage Service

WFS Web Feature Service

WKT Well Known Text

WMS Web Map Service

WPS Web Processing Service

Capítulo 1

Introducción

La actividad económica y el bienestar humano se encuentran sustentados gracias al medioambiente y su deterioro implica una reducción en la calidad de vida de los seres humanos (Skidmore, 2003). Su importancia aumenta cada día, lo cual se fomenta dado el desgaste que se genera sobre los recursos, provocando alteraciones en el ambiente que afectan a todos los seres vivos (Cabra Forero *et al.*, 2015).

Existen tres componentes necesarios para un manejo satisfactorio del ambiente y de los recursos naturales: políticas, participación e información. Estos tres componentes interactúan entre sí y son especialmente críticos en países menos desarrollados donde la infraestructura es frecuentemente rudimentaria. El equilibrio entre estos tres componentes y su influencia en el manejo dependerá del problema en gestión, así como la infraestructura y las tradiciones sociales, económicas y culturales del país (Skidmore, 2003).

Los avances en la tecnología y la necesidad de hacer compatible la información obtenida de diferentes fuentes, además de la aparición de la Arquitectura Orientada a Servicios (SOA por sus siglas en inglés), hicieron que los SIG fueran cambiando su concepto de aplicaciones independientes y evolucionaran a Infraestructuras de Datos Espaciales (IDE) (Bocher & Neteler, 2010), las cuales están siendo ampliamente utilizadas en las ciencias ambientales para compartir, descubrir, visualizar y recuperar datos geoespaciales a través de protocolos de servicios web del Open Geospatial Consortium (OGC) (Giuliani *et al.*, 2011).

Sumado a esto, el rápido desarrollo de la tecnología de red ha dado lugar a una explosión de datos e información (Li *et al.*, 2010). Un ejemplo de ello es el Earth Observing System Data and Information System (EOSDIS) de la National Aeronautics and Space Administration (NASA), que genera más de 3 terabytes (TB) de datos diarios interdisciplinarios de las ciencias del sistama Tierra (NASA, 2007a).

Por otra parte, existen diversos tipos de formatos en los cuales las agencias disponen los datos al usuario, lo cual, supone un inconveniente que limita su uso por parte de usuarios que no son especialistas en procesamiento digital de imágenes y sistemas de información geográfica.

Con base en lo anterior, el objetivo del trabajo se enfoca en un aplicativo accesible desde la web que tenga la capacidad de contener una base de datos de acceso libre orientada a estudios ambientales para Suramérica (e.g., Ceccarelli *et al.*, 2015; Gorla, 2002), que tendrá a disposición productos/información en formato vectorial y ráster a 1km de resolución espacial y proyección cartográfica (lat/long), donde además se permitirá realizar operaciones entre las capas ráster, generar gráficos, estadísticos y extraer los resultados obtenidos.

1.1. Estructura de la tesis

La tesis está estructurada de la siguiente manera:

En el primer capítulo se da una introducción al contenido del trabajo y se presentan los objetivos del mismo.

En el segundo capítulo se presentan los conceptos que se emplearán a lo largo de la tesis, así como algunas generalidades de los sistemas de información geográfica.

En el tercer capítulo se presentan algunos sistemas de información geográfica web que se encuentran a disposición actualmente y que contienen información relacionada a estudios ambientales.

El cuarto capítulo presenta la implementación del sistema. Aquí se describe la configuración inicial del aplicativo cliente y las funcionalidades desarrolladas como valor agregado al aplicativo, mostrando ejemplos de sus posibles usos.

En el quinto capítulo se confrontan los resultados de procesos obtenidos mediante el aplicativo implementado contra otras herramientas de uso libre.

En el capítulo 6 se muestran las conclusiones, así como posibles trabajos futuros que contribuirán en el mejoramiento del aplicativo.

1.2. Objetivos

1.2.1. Objetivo General

Implementar un aplicativo web que contenga una base de datos de tipo vectorial y ráster de libre acceso para usuarios interesados en la realización de estudios ambientales en Suramérica, que permita extraer la información contenida, o bien efectuar consultas básicas de subproductos comunes a este tipo de estudios.

1.2.2. Objetivos Específicos

- Procesar la información ráster suministrada.
- Estructurar una base de datos que contenga información socioeconómica en formato vectorial de los países de Suramérica.
- Estructurar una base de datos que contenga información climática y ambiental en formato ráster para los países de Suramérica.
- Implementar consultas que permitan realizar operaciones básicas para datos ráster y vectoriales.
- Implementar consultas que retornen estadísticos de las series de datos ráster contenidas en las bases de datos.

Capítulo 2

Sistemas de Información Geográfica

2.1. Conceptos Generales

A continuación se presentan algunas definiciones que serán utilizadas a lo largo del documento para la descripción de las diferentes tecnologías utilizadas y su implementación.

- **Entidades Geográficas:** Son representaciones de los elementos ubicados en la superficie terrestre (e.g., ríos, vegetación) o divisiones de terreno (e.g., divisiones políticas y parcelas de terreno). Comúnmente son representados como puntos, líneas o polígonos (ESRI, 2012).

- **Protocolo:** Es un conjunto de reglas usadas por las computadoras para comunicarse unas con otras a través de la red (e.g., HTTP, FTP, TCP/IP, entre otros) (Aguirre Patiño *et al.*, 2011).

- **Open Source Geospatial Foundation (OSGeo):** Es una institución sin ánimo de lucro cuya misión es la de promover la adopción de tecnología geoespacial abierta (OSGeo, 2016).

- **OGC:** Sociedad internacional compuesta por más de 522 empresas, agencias gubernamentales y universidades, que participan en un proceso de consenso para desarrollar estándares de interfaz disponibles públicamente. Estos estándares permiten a los desarrolladores de tecnología, hacer la información espacial y los servicios complejos, accesibles y útiles con todo tipo de aplicaciones (OGC, 2016a).

- **Framework:** Estructura *software*, la cual tiene componentes que se pueden personalizar e intercambiar con el fin de desarrollar una aplicación, es decir, es una base en la que es posible añadir o quitar complementos para construir una herramienta personalizada. Su objetivo principal es promover la reutilización de código para la aceleración del desarrollo de nuevas aplicaciones.
- **Cliente:** Es el componente en una arquitectura cliente-servidor que se encarga de enviar peticiones, recibir la información espacial y representarla para que el usuario pueda visualizarla. En otras palabras, es el intermediario entre el usuario y los datos ofrecidos por el servidor (Olaya, 2011).
- **Servidor:** Es el componente en una arquitectura cliente-servidor que contiene los datos espaciales, procesa las solicitudes del cliente y da respuesta a ellas. La solicitud de información por parte del cliente al servidor se lleva a cabo mediante métodos de consulta HTTP GET o HTTP POST, los cuales reservan una serie de caracteres para su construcción (ver Tabla 2.1).

Caracter	Uso reservado
?	Separador que indica el inicio de la cadena de consulta
&	Separador entre parámetros en la cadena de consulta
=	Separador entre el nombre y el valor del parámetro
,	Separador entre valores individuales en una lista de parámetros (como BBOX, LAYERS, y ESTILOS) en la solicitud
+	representación taquigráfica de un caracter de espacio

Tabla 2.1: Caracteres reservados en una cadena de consulta.

Fuente: de La Beaujardiere (2006)

2.2. ¿Qué es un SIG?

Un SIG (o GIS por sus siglas en inglés), es una herramienta para la captura, almacenamiento, consulta, manipulación y representación de información espacial. Permite visualizar

diferentes tipos de datos sobre un mapa, facilitando de esta manera tanto su análisis como la comprensión de patrones y relaciones (MapServer, 2016).

Un SIG utiliza generalmente dos tipos de estructuras de información, ráster y vectorial. Los datos ráster consisten en un arreglo matricial de filas y columnas codificados de acuerdo a una serie de valores y cuyo nivel de detalle depende del tamaño de las celdas (Korte, 2001). Los datos vectoriales ayudan a representar objetos espaciales del mundo real por medio de puntos líneas (o polilíneas) y/o polígonos, los cuales tienen asociados atributos que describen cada elemento espacial. Un SIG está compuesto por cinco elementos básicos que se interrelacionan entre sí (Figura 2.1):

- ***Personas:*** Como personas nos referimos tanto a los usuarios del sistema, como a aquellos encargados de su diseño y mantenimiento.
- ***Datos:*** Hace referencia a los archivos que contienen la información geográfica que se almacenará en el servidor.
- ***Hardware:*** Es la infraestructura requerida para el correcto funcionamiento del sistema.
- ***Software:*** Programas que se necesitan para la configuración del sistema, almacenamiento y tratamiento de datos, así como una interfaz que permita su análisis y visualización por parte de los usuarios.
- ***Métodos:*** Son modelos y prácticas operativas de cada sistema, es decir, las reglas de actividad bien diseñadas que conllevan a una correcta implementación (Carmona & Monsalve, 2004).

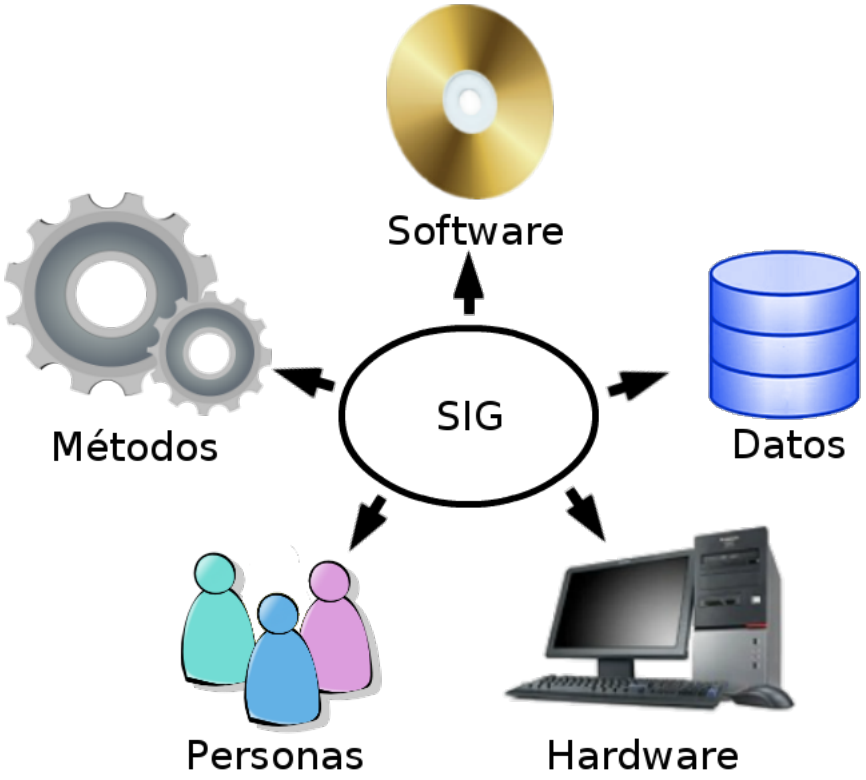


Figura 2.1: Componentes de un SIG.

2.3. Web Mapping

El avance de la tecnología ha permitido el acceso a la información contenida en un SIG utilizando la arquitectura cliente–servidor, por lo cual se hacen necesarios componentes del lado del servidor para distribuirla, así como componentes por parte del cliente para acceder a ésta. De esta manera, las tecnologías de *Web Mapping* ayudan a incorporar las ideas de los SIG dentro de páginas Web por medio de un navegador Web como aplicación principal (a lo que se denomina como *WebGIS*) (Olaya, 2011).

2.3.1. Formatos para la interoperabilidad

Con el fin de lograr una mayor habilidad para el intercambio de datos entre diferentes sistemas, la OGC define cierto tipo de formatos para codificar la información geoespacial y lograr su correcta transferencia e interpretación por cada *software* certificado por la

OSGeo (Zader, 2016).

2.3.1.1. Geography Markup Language

El Geography Markup Language (GML), es un tipo de gramática XML para expresar características geográficas, sirviendo como lenguaje de modelado para sistemas geográficos, así como formato de intercambio abierto para transacciones geográficas en Internet (OGC, 2016b).

2.3.1.2. GeoJSON

Es un formato para la codificación de datos geográficos en una colección de pares nombre/valor, pudiendo representar una geometría, característica o colección de características (Butler *et al.*, 2008).

2.3.1.3. Keyhole Markup Language

El Keyhole Markup Language (KML), es un lenguaje de especificación basado en XML que se centra en la visualización de datos geográficos. Este lenguaje fue enviado por Google a la OGC con el fin de estandarizarlo y fomentar así una mayor implementación, interoperabilidad y compartir el contenido de información espacial (Butler *et al.*, 2008).

2.3.1.4. Well Known Text

El Well Known Text (WKT), es un lenguaje de fácil interpretación por las máquinas y los usuarios que contribuye a la representación de entidades geográficas, así como la descripción de un sistema de referencia de coordenadas (CRS por sus siglas en inglés) o una operación de coordenadas (Lott, 2015).

2.3.2. Protocolos para la interoperabilidad

La OGC ofrece diversos estándares que definen protocolos de comunicación entre servidores y clientes para la solicitud y transferencia de información. Entre los estándares más utilizados se encuentran el Web Map Service, Web Coverage Service y Web Feature Service, los cuales se detallan a continuación:

2.3.2.1. Web Map Service

El Web Map Service (WMS), es un estándar que provee una interfaz HTTP simple para la solicitud de imágenes de mapas georreferenciados de una o más bases de datos geoespaciales. Una petición WMS define la capa o capas geográficas y el área de interés que se va a procesar. La respuesta a la solicitud es una o más imágenes de mapas georreferenciadas en diferentes formatos (JPEG, PNG, TIFF, entre otros.) que se pueden mostrar en una aplicación de navegador (OGC, 2016d).

Dentro de las principales consultas del estándar WMS podemos encontrar las siguientes:

- **GetCapabilities:**

  Su propósito es generar un documento XML con los metadatos del servicio, lo cual es una descripción de la información contenida en el servidor y los parámetros de solicitud aceptados (ver Tabla 2.2).

Parámetro de solicitud	Obligatorio	Descripción
VERSION = versión	No	Versión del protocolo de consulta
SERVICE = WMS	Si	Tipo de servicio
REQUEST = GetCapabilities	Si	Nombre de la solicitud
FORMAT = Formato de salida	No	Formato de salida de los metadatos del servicio
UPDATESEQUENCE = string	No	Número o string de secuencia para control de cache

Tabla 2.2: Caracteres de la solicitud URL GetCapabilities

Fuente: de La Beaujardiere (2006)

- **GetMap:**

  Esta operación hace posible que el usuario obtenga un mapa. El servidor recibe la solicitud, la cual deberá satisfacer o emitir una excepción al servicio. Los parámetros obligatorios para efectuar la consulta se describen en la tabla 2.3:

Parámetro de solicitud	Descripción
VERSION = versión	Versión del protocolo de consulta
REQUEST = GetMap	Nombre de la solicitud
LAYERS = Lista de layers	Lista separada por comas de una o más capas de salida
STYLES = Lista de estilos	Lista separada por comas del estilo de renderizado de una o más capas
CRS = SRID:Código	Sistema de referencia de coordenadas (e.g., EPSG:4326)
BBOX = minX, minY, maxX, maxY	Coordenadas de las esquinas de la capa (Inferior izquierda, Superior derecha) en las unidades del CRS.
WIDTH = Ancho de salida	Ancho en píxeles de la imagen
HEIGHT = Altura de salida	Altura en píxeles de la imagen
FORMAT = Formato de salida	Formato de salida de la capa

Tabla 2.3: Caracteres de la solicitud URL GetMap

Fuente: de La Beaujardiere (2006)

- **GetFeatureInfo:**

  Mediante este estándar, el usuario puede obtener información de un determinado píxel (i,j) de la imagen. El GetFeatureInfo indica al servidor la capa que se está observando, incluyendo la mayoría de parámetros que se indican en el GetMap. Adicionalmente se requiren ciertos parámetros obligatorios para la consulta (ver Tabla 2.4).

Parámetro de solicitud	Descripción
VERSION = versión	Versión del protocolo de consulta
REQUEST = GetFeatureInfo	Nombre de la solicitud
map request part	copia parcial de los parámetros del mapa del cual se desea la información
QUERY_LAYERS = Lista de layers	Lista separada por comas de una o más capas de salida
INFO_FORMAT = Formato de salida	Formato de salida de la información
I = píxel columna	coordenada i en píxeles del mapa
J = píxel fila	coordenada j en píxeles del mapa

Tabla 2.4: Caracteres de la solicitud URL GetFeatureInfo

Fuente: de La Beaujardiere (2006)

2.3.2.2. Web Coverage Service

El Web Coverage Service (WCS), es un estándar para la recuperación de datos geoespaciales, es decir, proporciona acceso a capas con información geoespacial en formas que son útiles para la representación del lado del cliente. Esta información puede servir como entrada a modelos científicos o ser utilizada por otros clientes mediante una solicitud llevada a cabo generalmente por tres tipos de operaciones que son procesadas por parte del servidor:

- **GetCapabilities:**

  Por medio de esta solicitud, un cliente puede solicitar el documento de metadatos del servicio en formato XML.

- **DescribeCoverage:**

  Al hacer uso de esta operación, un cliente solicita descripciones completas de una o más coberturas proporcionadas por un servidor WCS particular. El servidor responde con un documento XML que describe completamente las coberturas identificadas.

- **GetCoverage:**

  Mediante esta operación, el cliente puede efectuar la solicitud de una cobertura, entonces, el servidor extrae los datos de la misma y los codifica en un formato conocido (e.g., TIFF) que se entrega al usuario (Whiteside & Evans, 2006). Los parámetros requeridos para una consulta de este tipo son los siguientes:

Parámetro de solicitud	Descripción
service = WCS	Identificador del tipo de servicio
request = GetCoverage	Nombre de la solicitud
version	Versión del protocolo de consulta
coverageId	Identificador de la capa requerida
subset	Dimensiones del recorte, se define un subset por dimensión
format = Formato de salida	Formato de salida de la información
crs	Definición del Sistema de Referencia de Coordenadas

Tabla 2.5: Parámetros de la solicitud URL GetCoverage

Fuente: Whiteside & Evans (2006)

2.3.2.3. Web Feature Service

El Web Feature Service (WFS), es un estándar que se utiliza para el acceso y manipulación de entidades geográficas vectoriales codificadas en GML proveniente de múltples fuentes. Existen tres clases de WFS: Básico, Xlink y Transaccional.

El WFS Básico implementa las operaciones GetCapabilities, DescribeFeatureType and GetFeature; el WFS Xlink soporta las operaciones del WFS Básico más GetGmlObject y el WFS transaccional, soporta las operaciones del WFS Básico y opcionalmente las operaciones GetGmlObject y/o LockFeature.

- **GetCapabilities:**

  Al igual que en el WMS, ayuda a que un cliente solicite el documento de metadatos del servicio en formato XML.

- **DescribeFeatureType:**

  Su función es generar una descripción de los esquemas de los tipos de entidad contenidos en una implementación WFS.

- **GetFeature:**

  Al hacer uso de esta operación, el cliente realiza una solicitud que retorna una o más entidades geográficas vectoriales. En la siguiente tabla se describen los parámetros comunes de las solicitudes WFS (OGC, 2016c).

Parámetro de solicitud	Descripción
version	Especifica el número de versión del protocolo
service = WFS	Nombre de la solicitud
request	Indica la operación a realizar (e.g., GetFeature, GetCapabilities.)
namespace	Es un parámetro opcional que permite definir el espacio de trabajo al que se encuentra vinculada determinada capa.

Tabla 2.6: Parámetros de la solicitud URL GetFeature

Fuente: OGC (2016c)

2.4. Clientes y Servidores

En la arquitectura cliente–servidor (Figura 2.2), un número variable de clientes se conectan a un servidor que contiene información geográfica y cada cliente efectúa peticiones al servidor. El servidor, procesa y retorna los datos a través de los protocolos de servicios para que cada cliente almacene tan sólo lo que es de su interés (Olaya, 2011).

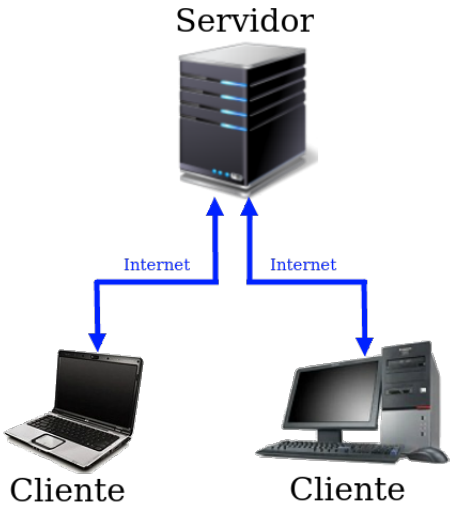


Figura 2.2: Arquitectura Cliente–Servidor.

2.4.1. Clientes

Son los componentes que utilizan los datos que proporciona el servidor mediante los protocolos de servicio. Estos deben tener la capacidad de elaborar solicitudes e interpretar las respuestas por parte del servidor, incluyendo las componentes de representación que han sido creadas con la forma típica de un visor y que varían de un cliente a otro (Olaya, 2011).

2.4.1.1. Lenguajes de Programación y Software SIG

- **JavaScript:**

  Es un lenguaje de programación interpretado, utilizado principalmente en la construcción de páginas web (Flanagan, 2007), siendo su uso posible gracias a la ejecución de un programa capaz de analizarlo y ejecutarlo, como un navegador web (Universidad de Valencia, 2015).

- **Python:**

  Es un lenguaje de programación orientado a objetos de libre distribución propuesto por van Rossum & de Boer (1991), disponible para sistemas operativos Windows, Unix, Linux y Mac OS. Python presenta una sintaxis no muy compleja pero con

estructuras de datos de alto nivel, lo cual lo hace ideal para el desarrollo rápido de aplicaciones en diversas áreas (Van Rossum & Drake Jr, 2004). Cuenta con un intérprete que funciona como una consola Unix. Dicho intérprete puede ser utilizado de forma interactiva, lo cual ayuda a evaluar funciones a medida que se desarrolla el programa (Van Rossum & Drake Jr, 1995).

- **R:**

  Es un lenguaje de programación propuesto por Robert Gentleman y Ross Ihaka en 1993 (Ihaka, 1998). R es ampliamente empleado por la comunidad estadística, que proporciona gráficos de alto nivel, interfaz a otros lenguajes y facilidades de depuración (R Core Team, 2000).

- **GRASS GIS:**

  Es un *software* SIG de código abierto utilizado para el manejo de datos geoespaciales (procesamiento, generación de gráficos, producción de mapas, modelamiento espacial y visualización). GRASS GIS ofrece tanto una interfaz gráfica como una línea de comandos que posibilita la simplificación de operaciones entre los datos (Embrapa, 2017), permitiendo la ejecución de sus comandos mediante *scripts* (archivos que contienen una secuencia de órdenes) desarrollados en lenguajes como Python (Neteler & Mitasova, 2008).

- **QGIS:**

  QGIS es un Sistema de Información Geográfica de código abierto que funciona en la mayoría de plataformas Unix, Windows y Mac. Proporciona un visor de datos SIG que admite diversos formatos ráster y vectoriales, así como una consola Python para el desarrollo de *scripts* (QGIS Development Team, 2018).

2.4.1.2. Librerías para el Desarrollo de Clientes SIG

Existen diferentes librerías que ofrecen cientos de módulos y herramientas para crear desde cero un WebGIS, entre las que se destacan:

- **OpenLayers:**

  Es una librería de código libre escrita en JavaScript cuya función es cargar y desplegar

datos espaciales en la mayoría de navegadores web modernos (OpenLayers, 2008). Permite crear mapas interactivos y proporciona la capacidad de personalizar cada aspecto del mismo con la ventaja que al ser una librería del lado del cliente, no se requiere el uso de ningún *software* especial del lado del servidor (Hazzard, 2011).

- **ExtJS:**

  Es una librería basada en JavaScript que ofrece cientos de componentes de interfaz de usuario de alto rendimiento, que posibilitan el desarrollo de aplicaciones web multiplataforma tanto para dispositivos móviles como para equipos de escritorio (Sencha, 2016). Esta librería cuenta con un tipo de licencia gratuita y uno comercial, la diferencia radica en que si se utiliza la licencia gratuita, se exige la liberación del código fuente de la aplicación desarrollada.

- **GeoExt:**

  Es una librería basada en JavaScript que combina las funcionalidades SIG de OpenLayers con la interfaz de usuario de la librería ExtJS (GeoExt, 2016). Incluye componentes estandarizados para la creación de un WebGIS, con la apariencia y funcionalidad de un SIG de escritorio (Fonts Bartolomé & Pericay, 2012).

2.4.1.3. Clientes Preconstruidos

Además de librerías, existe la posibilidad de partir de clientes de código abierto preconstruidos que utilizan herramientas de uso libre disponibles para su personalización, entre los que se destacan:

- **GeoExplorer:**

  Es una aplicación cliente basada en las librerías GeoExt y OpenLayers, la cual hace uso de los estándares de la OGC para acceder a diferentes fuentes de datos SIG internas o externas. De esta manera brinda al usuario la posibilidad de desplegar y visualizar mapas online, manipulando sus propiedades (orden de visualización, transparencia, entre otros) (Fonts Bartolomé & Pericay, 2012).

- **Opengeo Suite:**

  Es un paquete para Web Mapping compuesto por los principales proyectos de *software*

libre: PostGIS, GeoServer, GeoWebCache, OpenLayers y GeoExt, que incorpora el uso de los estándares de la OGC para la transferencia, interpretación y visualización de la información espacial a través de aplicaciones WebGIS, dispositivos móviles y clientes de escritorio. Entre las principales ventajas que presenta está la fácil instalación para los diferentes sistemas operativos que soporta (Linux, Windows, MAC OS X y Solaris). Asimismo, al integrar los diferentes *software* en un solo paquete, utiliza versiones compatibles entre sí, lo cual facilita la configuración del entorno y el desarrollo de aplicaciones (Romeu Carrasco *et al.*, 2012). Opengeo Suite ofrece el GeoExplorer como aplicación cliente por defecto, además de plantillas con las cuales el desarrollador puede partir de un cliente preconstruido para personalizar una aplicación propia.

- **GXP:**

  Es un framework basado en OpenLayers2 y GeoExt, el cual provee un conjunto de plugins que pueden ser incorporados para personalizar una aplicación de mapas básica (Zader, 2016).

- **ol3:**

  Opengeo Suite ofrece dos tipos de plantillas, para visualización (ol3view) y edición (ol3edit), basadas en OpenLayers3 y Bootstrap que al igual que la plantilla GXP parten de una aplicación básica y una serie de plugins que se adicionan al aplicativo básico para su personalización.

2.4.2. Servidores de Mapas

Existen diferentes tipos de servidores, dentro de los cuales se destacan, por su alto rendimiento y por ser de uso libre, los siguientes:

- **Mapserver:**

  Es un servidor de código abierto con licencia GNU–General Public License escrito en lenguaje C, desarrollado por la Universidad de Minnesota en cooperación con la NASA, que facilita la publicación en la web de datos espaciales y aplicaciones

cartográficas interactivas. Está soportado en diversos tipos de plataformas (e.g., Linux, Windows, Mac OS X, Solaris, entre otros) y reconoce diferentes tipos de lenguajes de programación y ambientes de desarrollo (e.g., PHP, Python, Perl, Ruby, Java, y .NET). Asimismo, el Mapserver implementa numerosos estándares del OGC entre los que se destacan el WMS, WCS y el WFS no transaccional. Soporta más de mil proyecciones y tiene la capacidad de manipular múltiples formatos ráster (algunos de manera directa TIFF/GeoTIFF, NetCDF, MrSID, ECW, y otros formatos utilizando la librería GDAL) y vectoriales (ESRI Shapefiles, Postgis, SpatiaLite, ESRI ArcSDE, Oracle Spatial, MySQL y otros por medio de la librería OGR) (MapServer, 2016).

- **Geoserver:**

  Es un servidor basado en Java, con el cual, los usuarios pueden ver y editar datos geoespaciales. Utiliza estándares abiertos establecidos por el OGC, proporcionando a los desarrolladores una gran flexibilidad para la creación de mapas e intercambio de información (Geoserver, 2014). Se inició en 2001 por la organización sin ánimo de lucro The Open Planning Project (TOPP), con el fin de mejorar la capacidad de compartir datos espaciales, brindando a los ciudadanos herramientas que les permitieran involucrarse en el gobierno y el planeamiento urbano (Geoserver, 2016b). Geoserver es la referencia de la implementación de los estándares WCS y WFS de la OGC y presenta un alto rendimiento para la ejecución del estándar WMS (Geoserver, 2016c). Provee al usuario una interfaz que suministra herramientas para manipular los contenidos del servidor (espacios de trabajo, almacenes de datos, capas y estilos). Asimismo, presenta compatibilidad con diversos formatos de capas ráster (utilizando la extensión ImageIO-Ext de la librería GDAL) y vectoriales (csv, Postgis, ESRI, Capas WFS de otros servidores, entre otros) (Geoserver, 2016a).

Capítulo 3

WebGIS Ambientales

3.1. Importancia de los WebGIS en los Estudios Ambientales

El rápido avance de la tecnología ha traído consigo una explosión de información. Cada año se lanzan nuevos satélites y se generan nuevos tipos de formatos para almacenar de una manera más eficiente el volumen de datos generado. Esto convierte su procesamiento en una carga adicional para los usuarios, pues requiere de gran dedicación y un tiempo extenso por parte de quienes no están familiarizados con su estructura, conllevando a que en muchos casos, grandes cantidades de datos no sean estudiadas (Li *et al.*, 2010).

Aquí radica la importancia de los WebGIS. Estas plataformas ponen a disposición de los usuarios, información espacial preprocesada, en formatos conocidos y compatible entre sí, facilitando su manejo, análisis e interpretación, y contribuyendo de esta manera al avance en campos como la educación y la investigación.

3.2. Ejemplos de WebGIS Ambientales

Son innumerables los ejemplos en los que los WebGIS se han utilizado aportando al estudio del ambiente. Al ser herramientas que proporcionan información espacial (e.g., imágenes de satélite, fotografías aéreas y entidades geográficas), pueden ser utilizadas en una amplia variedad de disciplinas. Por ejemplo, muchos biólogos y ecólogos se interesan

por la investigación de la interacción entre especies (e.g., de Souza *et al.*, 2015) y/o su relación con el medio que las rodea, con el fin de evaluar el impacto que causan los cambios ambientales en su distribución (Fernandez Gómez *et al.*, 2001). El clima también es una variable de gran importancia ambiental a nivel mundial, por esta razón, eventos como inundaciones, sequías y heladas, son estudiados desde diversas disciplinas como la agronomía (e.g., Erena *et al.*, 2009; Gutiérrez Lozano *et al.*, 2011; Ravelo & Pascale, 1997), geografía (e.g., Cantos & Ferrer, 2015), agrimensura (e.g., Orfeo & Ruberto, 2000), entre otras.

Con el paso del tiempo se incorporan nuevas herramientas que simplifican el desarrollo de aplicaciones Web. Estas implementan consultas específicas y ponen a disposición del usuario información que facilita el estudio de fenómenos de acuerdo al objetivo para el cual fueron creadas. A continuación se presentan algunos ejemplos de WebGIS que se encuentran disponibles en la actualidad y que utilizan datos ambientales para diferentes finalidades.

- **WebGIS ISAGRO:**

  Es un sistema de información geográfica satelital dedicada al agro (Figura 3.1). Fue desarrollado gracias al esfuerzo de instituciones de Argentina, Chile, Uruguay y Paraguay, y financiado con aportes públicos regionales del Banco Interamericano de Desarrollo y de instituciones de los cuatro países (ISAGRO, 2017a) [1].

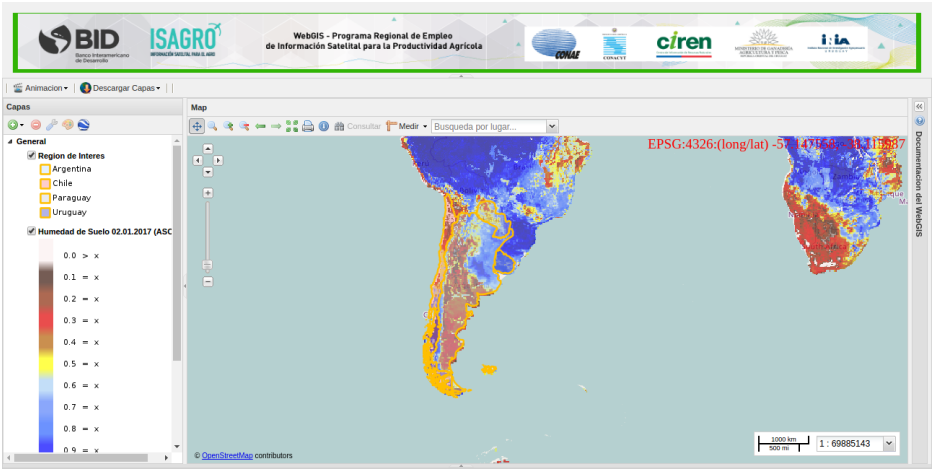


Figura 3.1: WebGIS ISAGRO.

[1] http://webgis.isagro.org.ar/geoexplorer/composer/

Su principal objetivo es difundir de modo efectivo y amigable información satelital a diversos actores agrícolas para la toma de decisiones en cada uno de los países (ISAGRO, 2017b). El WebGIS ISAGRO utiliza GeoExplorer como aplicación cliente (ver Sección 2.4.1.3), proporciona diferentes capas ráster actualizables diariamente, así como series históricas para los últimos 10 años, las cuales se basan en datos provenientes de sensores como Moderate Resolution Imaging Spectroradiometer (MODIS)[2], Global Precipitation Measurement (GPM)[3], ALOS-PALSAR[4], Soil Moisture Ocean Salinity (SMOS)[5] y Advanced SCATerometer (ASCAT)[6]. Entre las capas y consultas existentes encontramos las siguientes:

- **Capas vectoriales:**
  - Polígonos de cada uno de los cuatro países.
  - Polígonos de huellas de las imágenes Landsat8.
- **Capas Ráster:**
  - Áreas Forestadas
  - Bosque
  - Fuego
  - Heladas
  - Humedad de Suelo
  - Meteorología
  - Vegetación
  - Humedad de la Vegetación
  - Precipitación
  - Sequía

[2]https://modis.gsfc.nasa.gov
[3]https://www.nasa.gov/mission_pages/GPM/main/index.html
[4]https://earth.esa.int/web/guest/missions/3rd-party-missions/historical-missions/alos/instruments/palsar/
[5]https://earth.esa.int/web/guest/missions/esa-operational-eo-missions/smos
[6]https://www.eumetsat.int/website/home/Satellites/CurrentSatellites/Metop/MetopDesign/ASCAT/index.html

- **Consultas:**

  Proporciona a los usuarios la ejecución de diferentes operaciones, como hacer zoom a una capa, ver sus propiedades, medir distancias, ver animaciones para algunos productos, realizar descargas, entre otros. Publica la información disponible a través de los protocolos WMS, WFS y WCS, lo que permite interoperar con otras IDE y cargar capas en un *software* SIG de escritorio. Además cuenta con el servicio Web Processing Service (WPS) para la ejecución de geoprocesos como servicios web (Zader, 2016).

- **GeoCVC:**

  La Corporación Autónoma Regional del Valle del Cauca (CVC), en convenio con la Universidad del Valle, puso en marcha en 2014 el visor avanzado GeoCVC (Figura 3.2). Mediante este WebGIS, diversos tipos de usuarios pueden consultar la información básica de las capas ambientales que genera la corporación, así como diferentes fuentes datos de archivo que se han venido construyendo a lo largo de los años, como fotografías aéreas históricas del departamento (las cuales han sido escaneadas y ortorrectificadas) e imágenes satelitales (las cuales han sido corregidas atmosféricamente y ortorrectificadas).

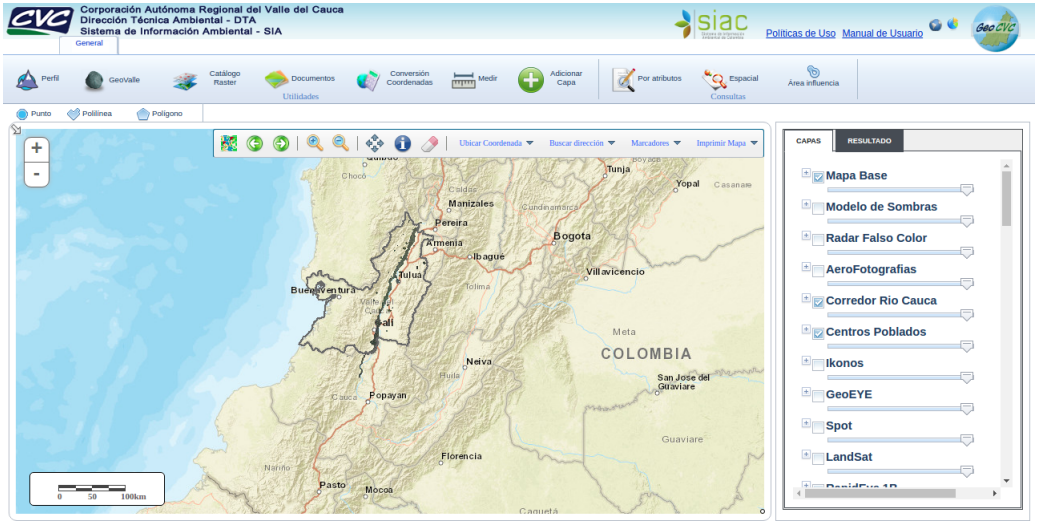


Figura 3.2: SIG Ambiental para el Valle del Cauca, Colombia - GeoCVC.

GeoCVC está basado en la Application Programming Interface (API) de ArcGIS 9.3

y es compatible con HTML5, aunque está abierto a todo público, al tener información protegida por derechos de autor, no todos pueden tener acceso a los datos crudos. Sin embargo, por medio de protocolos WMS, se pueden visualizar sus capas en un SIG de escritorio. De esta manera, se puede decir que el visor está principalmente dirigido a funcionarios de la corporación que requieran imágenes y cartografía base para continuar con el monitoreo de recursos ambientales y la generación de cartografía temática a partir de ésta. El visor avanzado proporciona diversas consultas, además de capas ráster y vectoriales como:

- **Capas vectoriales:**

  Actores sociales, grupos étnicos, estaciones de monitoreo (vertimientos, aire, agua), red hidroclimatológica, aguas subterráneas, humedales, hidrografía, cuencas, división política, incendios forestales, inundaciones, biodiversidad, entre otras.

- **Capas Ráster:**

  Entre estos se destacan archivos en formato tiff provenientes de fotografías aéreas históricas del departamento que han sido escaneadas y ortorrectificadas, además de imágenes de radar aerotransportado, datos LiDAR del corredor del Río Cauca e imágenes satelitales de mediana y alta resolución como Ikonos, GeoEYE, RapidEYE, Spot, Landsat, QuickBird y EROS.

- **Consultas:**

  El visor avanzado implementa el protocolo WMS para desplegar las capas en un SIG de escritorio, además de consultar la información de cada una de ellas, controlar su opacidad, dibujar polígonos, polilíneas, puntos sobre el mapa, medir distancias, ubicar coordenadas, buscar direcciones e imprimir el resultado final en diferentes formatos de papel, entre otras consultas.

- **Base de datos virtual de los ecosistemas costeros en la Cuenca de Campos, Brasil:** El WebGIS **Base de dados virtual dos Ecossistemas Costeiros da Bacia de Campos** (Figura 3.3), fue desarrollado por la empresa brasileña Embrapa, a partir del proyecto "Cambio climático global y el funcionamiento de los ecosistemas costeros en la Cuenca

de Campos: Una perspectiva espacio-temporal", financiado por el Consejo Nacional de Desarrollo Científico y Tecnológico (CNPq) bajo el programa de Investigación Ecológica de Larga Duración. El visor proporciona información acerca del plan de manejo de la cuenca, así como capas derivadas del análisis geoecológico desarrollado durante el proyecto. Su objetivo es apoyar a la población, las instituciones de investigación y otros organismos en la gestión y el desarrollo sostenible de la zona (Embrapa, 2017).

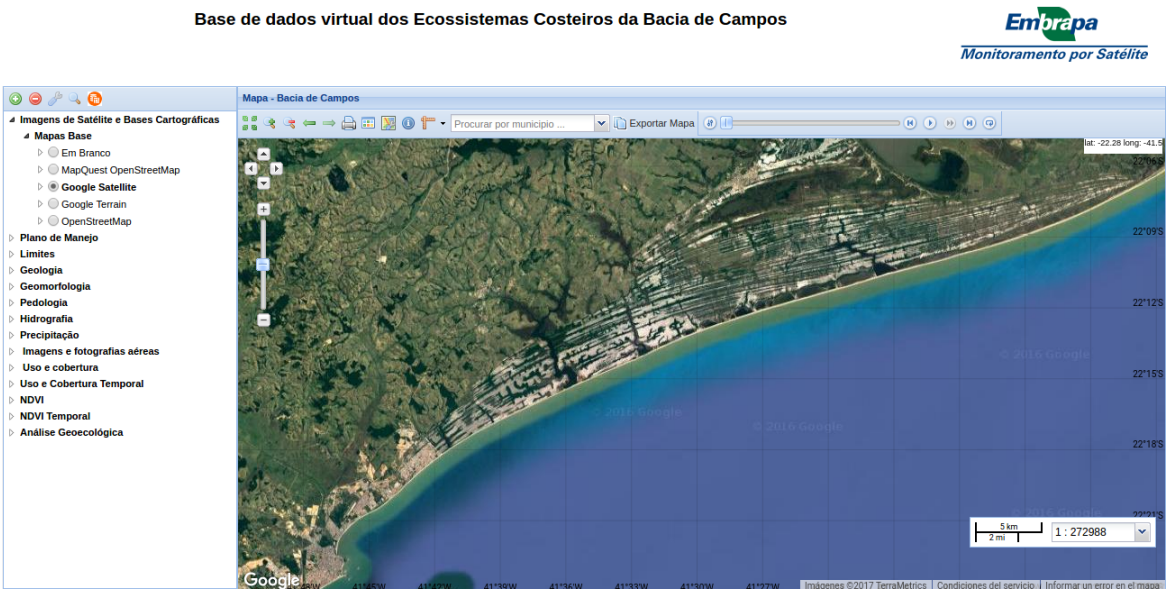


Figura 3.3: Base de datos virtual de los ecosistemas costeros de la Cuenca de Campos, Brasil.

Este WebGIS está desarrollado utilizando como base el framework GXP de la opengeo suite, el cual implementa librerías para web mapping de uso no comercial (GeoExt, OpenLayers y EXTJS) para su personalización. Entre las consultas, capas ráster y vectoriales que contiene están las siguientes:

- **Capas vectoriales:**

  Provee capas acerca de áreas estratégicas contenidas en el plan de manejo de la cuenca, límites, geología, geomorfología, edafología, hidrografía, precipitaciones, usos y coberturas y análisis geoecológicos.

- **Capas Ráster:**

  El visor suministra dos mosaicos de fotografías aéreas ortorrectificadas (1967 y 2007), así como 15 imágenes satelitales Landsat5 TM tomadas entre 1996 y 2011 de las cuales se derivaron productos de vegetación y capas de series temporales que se encuentran disponibles para consulta pero no para su descarga.

- **Consultas:**

  En el WebGIS se pueden utilizar herramientas básicas para manipular y extraer información de cada una de las capas (Zoom, Get Feature Info, Paneo del mapa, Ver Propiedades), imprimirlas como mapa o generar un link para publicar el mapa en otro sitio web. También es posible medir distancias sobre el mapa, realizar búsquedas utilizando el geolocalizador y visualizar la animación de la serie temporal de vegetación de la zona que se obtuvo previamente a partir de las imágenes Landsat5 TM.

- **WebGIS de Calidad de Agua de la República Argentina:**

  Este WebGIS (Figura 3.4), fue desarrollado por la empresa SUR Emprendimientos Tecnológicos para la Secretaría de Ambiente de Argentina, el cual fue pensado para manejar los datos de muestreos de agua que se llevan a cabo a nivel nacional, favoreciendo el conocimiento y la toma de decisiones por parte de la Secretaría en materia de agua segura (SUR Emprendimientos Tecnológicos, 2017b).

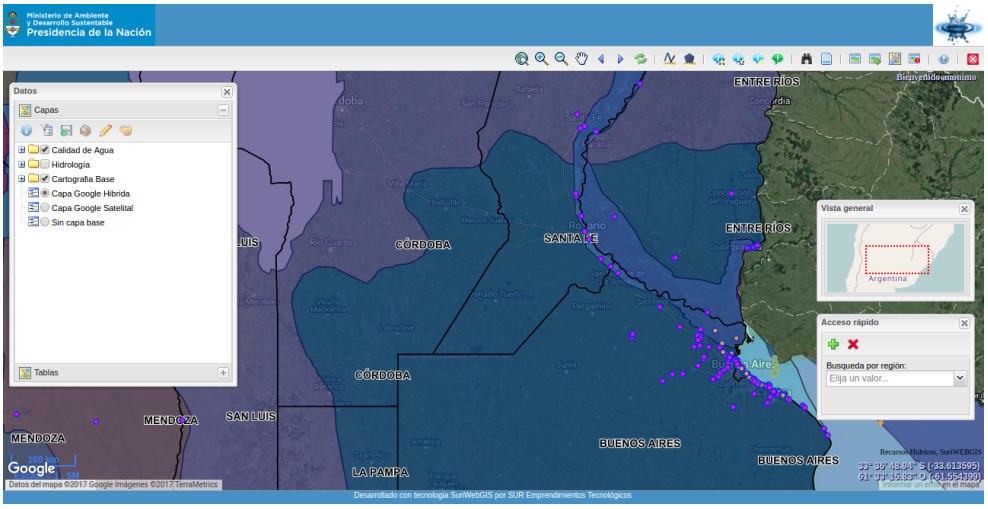


Figura 3.4: WebGIS Calidad de Agua de la República Argentina.

Está basado en SuriWebGIS, el cual es un sistema web de información geográfica construido a partir de librerías para web mapping de libre distribución y donde es posible capturar, almacenar y distribuir datos geoespaciales a partir de estándares WMS, WCS y WFS del OGC (SUR Emprendimientos Tecnológicos, 2017a). Presenta las siguientes capas vectoriales y consultas:

- **Capas vectoriales:**

  El WebGIS de Calidad de Agua de la República Argentina, además de mapas base suministrados por la API de Google, proporciona a los usuarios capas vectoriales referidas a Hidrología, cartografía base (provincias, departamentos, caminos, entre otras), información tipo punto y reportes de las estaciones de monitoreo de la calidad de agua.

- **Consultas:**

  El WebGIS cuenta con herramientas básicas de acercamiento o alejamiento (zoom), paneo, consulta de información de una capa vectorial por polígono, recuadro, punto o elemento, medición de distancias o áreas, realización de búsquedas por nombre de las estaciones de monitoreo, generación de reportes de parámetros críticos por cuenca, evolución de parámetros de calidad por estación o la evolución de un parámetro a lo largo de un perfil.

- **Google Earth Engine:**

  Esta plataforma ofrece a los usuarios acceso a datos globales de acceso gratuito como Landsat, MODIS, Modelo Digital de Elevación (MDE), mapas de coberturas del suelo, cultivos, temperatura superficial, entre otros (Figura 3.5). Además permite a los usuarios realizar el procesamiento de datos haciendo uso de la infraestructura de Google mediante scripts desarrollados por los usuarios en JavaScript o Python, así como herramientas disponibles en la plataforma[7].

[7] https://earthengine.google.com/

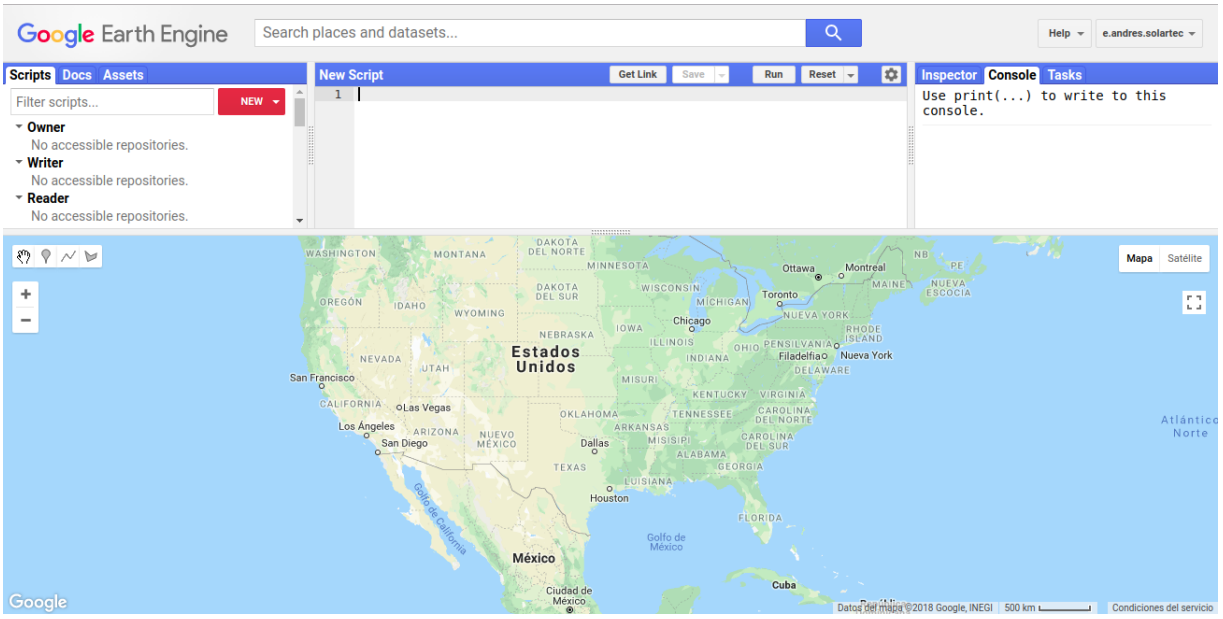


Figura 3.5: Plataforma Google Earth Engine.

Capítulo 4

Implementación del Sistema

4.1. Motivación

Los conjuntos de datos multitemporales y de gran cobertura geográfica son ampliamente utilizados dado a los requisitos crecientes de información precisa y actualizada para el monitoreo de los recursos, lo que resulta en altos requerimientos de cómputo y espacio de almacenamiento para los usuarios (Wang *et al.*, 2018). Sumado a esto, en la actualidad existen diversos tipos de formatos en los que se disponen los datos al usuario, resultando en un verdadero inconveniente que limita su uso por parte de aquellos que no son especialistas en procesamiento digital de imágenes y SIG. Por esta razón, es importante contar con un aplicativo web que ponga a disposición de los usuarios información de tipo ambiental en un solo lugar y compatible entre sí, dado que facilita el desarrollo de este tipo de estudios generando ventajas como:

- El usuario final no debe realizar preprocesamientos como georreferenciación, reproyección o remuestreo de las capas.

- Se disminuirá el tiempo y requerimientos de procesamiento para el usuario final, dado que los datos se encontrarán almacenados en un servidor central y serán compatibles entre ellos, lo que implica un mismo sistema de referencia, formato y resolución espacial.

- El usuario tendrá la posibilidad de desplegar la información y consultarla con

anterioridad a su descarga, facilitando el trabajo de selección de los datos requeridos para el estudio que se quiere llevar a cabo.

- Por otra parte, el WebGIS proporcionará herramientas para procesamiento, consulta y descarga de información ráster que permiten al usuario llevar a cabo algunos procesos de manera remota. Esto facilita el análisis de los datos previos a su descarga y la obtención de subproductos a partir de las capas ráster contenidas en el servidor.

4.2. Requerimientos del Sistema

Los requerimientos son las necesidades que se fijan para la construcción de un sistema. Para el desarrollo de este WebGIS, el director del presente trabajo, el Dr. David Gorla, estableció con anterioridad una serie de requerimientos que garanticen la utilidad del sistema para su uso en estudios de tipo ambiental, los cuales se presentan a continuación:

- El WebGIS debe tener una estructura que permita cargar a la base de datos las capas ambientales que serán puestas a disposición de los usuarios inicialmente.
- Se deben implementar consultas por píxel de las capas ráster.
- Se implementará la consulta de información de líneas, puntos o polígonos de las capas vectoriales.
- Se deberá contar con la opción de descarga de capas ráster y vectoriales contenidas en el servidor.
- El usuario podrá obtener el recorte de capas ráster a partir de selección de un recuadro en el visor.
- Se requiere generar gráficos y estadísticos de las series de datos contenidas en el WebGIS.
- Se podrán efectuar operaciones básicas como sumas, restas, divisiones y multiplicaciones entre capas ráster contenidas en el servidor.

- Debe proveerse un mecanismo para la descarga de la información ráster procesada por el usuario.
- La implementación se debe llevar a cabo utilizando *software* de uso libre.

Los requerimientos fueron discutidos y validados con el Mg. Pablo Zader, codirector de la tesis, quien ha trabajado previamente en el desarrollo de este tipo de proyectos (eg. WebGIS ISAGRO).

4.3. Herramientas Seleccionadas

4.3.1. Servidor de Mapas

Si bien MapServer proporciona un mejor desempeño y uso de memoria al estar escrito en lenguaje C, se optó por utilizar GeoServer. Este servidor de mapas ofrece una interfaz web que permite manipular fácilmente la base de datos del sistema adicionando conjuntos de mapas o creando estilos que permitan una mejor interpretación de lo que se muestra en el visor, lo que requiere un menor conocimiento en programación que otros servidores como MapServer. Además, al ser escrito en Java asegura una portabilidad entre diferentes sistemas operativos, facilitando la migración de información contenida en la base de datos.

4.3.2. Cliente

La Opengeo Suite integra los principales proyectos de *software* libre del lado cliente y servidor, incluido el visor de mapas escogido. Se optó entonces por utilizar esta plataforma y así obtener un conjunto de elementos configurados y compatibles entre sí para el desarrollo de visores web. Se hizo entonces uso del framework GPX que la plataforma proporciona y que permite incorporar plugins de librerías como OpenLayers2 (orientada al manejo de datos espaciales) y GeoExt (combina las funcionalidades de OpenLayers y ExtJS).

4.3.3. Lenguajes de Programación

- ***JavaScript:*** los complementos a desarrollar deben ser escritos en JavaScript e incorporados al aplicativo cliente, dado que este es el lenguaje base del framework

elegido. Asimismo, librerías como ExtJS y OpenLayers también se basan en este lenguaje, lo que facilita su uso para la personalización del aplicativo cliente.

- ***Python:*** Se eligió Python para el desarrollo de funcionalidades desde el lado del servidor dado que es un lenguaje de proposito general, ampliamente utilizado y que dispone de librerías/modulos para trabajar con datos geoespaciales. Python es compatible con programas como GRASS GIS y QGIS, así como diferentes lenguajes de programación como R, lo que facilita la integración de tecnologías a aplicar.

4.3.4. Software

Tanto QGIS como GRASS GIS son programas de uso libre que ofrecen herramientas para el procesamiento de datos ráster y vectoriales. Sin embargo se decició utilizar GRASS GIS desde el lado del servidor, dado que para el uso de la plataforma Opengeo Suite no es recomendable la instalación de QGIS en la misma máquina, ya que puede ocasionar conflictos con las librerías de GDAL que ambos utilizan (Boundless Spatial Inc., 2016a). Además, la mayoría de funciones de GRASS GIS pueden ser importadas y ejecutadas desde Python, lo que lo convierte en una herramienta interesante para el desarrollo de funcionalidades.

4.4. Configuración del Sistema

Teniendo en cuenta que el principal requerimiento planteado fue el uso de *software* libre, se propuso la arquitectura del sistema siguiendo el esquema que muestra la figura 4.1. En el mismo, el aplicativo cliente es el que genera las peticiones al servidor de aplicaciones, pudiendo seguir uno de tres caminos posibles:

- Acceder directamente al GeoServer, en los casos en que se consulta la información disponible mediante protocolos de transferencia de datos como WCS, WMS ó WFS.
- Ejecutar un procesamiento interno que utiliza Python para acceder a la base de datos y extraer valores para generar gráficos y estadísticas sobre píxeles o áreas definidas por el usuario.

- Utilizar Python como mecanismo de articulación de procesos con el *software* GRASS GIS y gracias a su función r.mapcalc efectuar operaciones entre los ráster contenidos en la base de datos. La información de salida debe ser retornada a Python, que posteriormente la envía al complemento desarrollado en el aplicativo cliente.

Figura 4.1: Arquitectura del sistema.

Para ello se optó por implementar la plataforma OpenGeo Suite, construida por completo con componentes geoespaciales de código abierto robustos e independientes, que integran las herramientas necesarias para el desarrollo de un aplicativo WebGIS de alto rendimiento, compatible con los estándares de la OGC (MappingGIS, 2012). Presenta ventajas en su instalación, por ejemplo, cada componente se puede adicionar por separado (ver Tabla 4.1), con lo cual, el desarrollador puede elegir sólo lo necesario para el funcionamiento del sistema. Además ofrece la capacidad de desarrollar aplicaciones cliente desde cualquier ordenador y generar un archivo comprimido en formato war (Web application ARchive), que se puede desplegar posteriormente en cualquier servidor.

Paquete	Funcionalidad	Dependencias
opengeo	Paquete principal. Su función es instalar los componentes requeridos tanto del lado cliente como del servidor	opengeo-server y opengeo-client
opengeo-server	Instalar los componentes requeridos del lado del servidor	postgis21-postgresql93, geoserver, geowebcache, geoexplorer, opengeo-dashboard, opengeo-docs, opengeo-tomcat
opengeo-client	Instalar los componentes requeridos del lado del cliente	postgis21, pgadmin3, pdal
opengeo-webapp-sdk	Permitir el desarrollo de aplicaciones (Boundless SDK)	Variable dependiente

Tabla 4.1: Componentes de la OpenGeo Suite

Fuente: Boundless Spatial Inc. (2016b)

Por otra parte, incluye tutoriales para explorar el manejo de la herramienta y la personalización básica de una aplicación cliente preconstruida basada en GXP u OpenLayers, que además pueden integrar otros lenguajes de programación para el procesamiento de la información geoespacial contenida en el servidor de mapas (e.g., Python). A continuación se describen los componentes utilizados para el desarrollo del aplicativo:

4.4.1. Servidor de Mapas

Son componentes que tienen como objeto interactuar con los clientes (web o servidor) para brindar el acceso a la información geoespacial existente, en general cumpliendo los estándares de la OGC (WMS, WCS, WFS, entre otros) (Zader, 2016).

La Opengeo Suite implementa como servidor de mapas el GeoServer, que entre sus ventajas presenta una interfaz RESTful, a través de la cual los usuarios pueden realizar

configuraciones simplemente mediante llamada HTTP. De esta manera no es necesario acceder a la interfaz de administración web (Geoserver, 2017).

4.4.1.1. Espacios de trabajo

GeoServer, mediante su interfaz de administración web, permite organizar la información, gracias a una estructura simple (Figura 4.2) donde se generan **espacios de trabajo**, en los cuales se pueden clasificar datos pertenecientes a diferentes áreas temáticas.

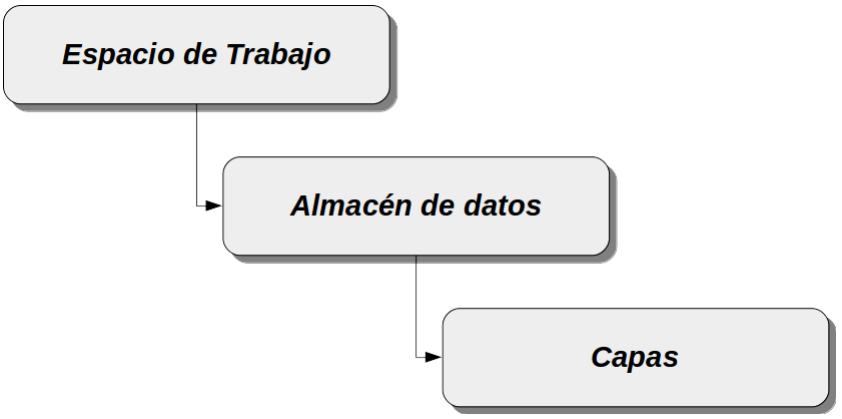


Figura 4.2: Estructura de almacenamiento en la interfaz de administración web de GeoServer.

Una vez creado el espacio de trabajo, es posible comenzar a almacenar cada una de las capas que este va a contener. Para esto, GeoServer presenta dos conceptos que se relacionan con ellas: el primero tiene que ver con los **almacenes de datos**, que son aquellos que brindan los datos de conexión a la base de datos, ruta de los ficheros en el sistema operativo y tipo de datos; el segundo concepto tiene que ver con las **capas**, aquí se define lo correspondiente a la visualización, título, sistema de referencia, extensión, simbología, entre otros (Fonts & González, 2013).

Teniendo en cuenta las capas que se pondrán a disposición en la fase inicial, según los requerimientos planteados, se crearon 5 espacios de trabajo para datos ráster (Figura 4.3), que se corresponden a NDVI, NDWI, Precipitación, Sequía y Temperatura.

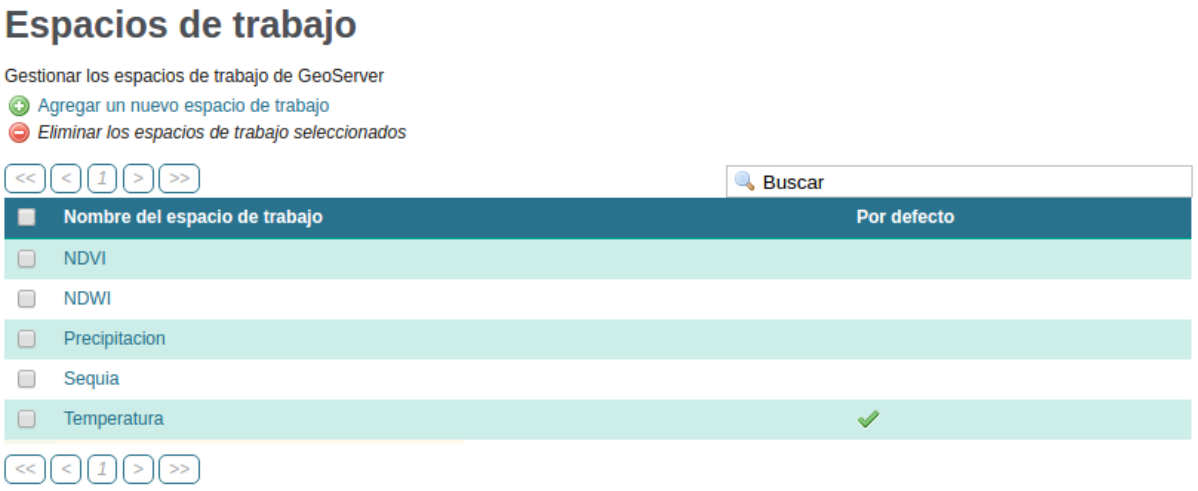


Figura 4.3: Espacios de trabajo creados para los índices requeridos.

4.4.1.2. Estilos

Una vez creados los espacios de trabajo, se configuraron los aspectos de visualización que tendrá cada índice en la aplicación cliente. Para esto, GeoServer brinda estilos predeterminados para capas vectoriales según sea su tipo (línea, punto o polígono), así como para datos ráster (Generic y ráster), permitiendo además crear estilos propios mediante su interfaz de administración web (Figura 4.4):

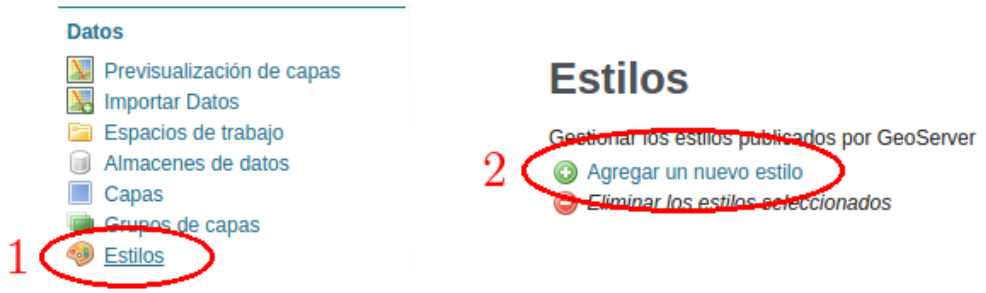


Figura 4.4: Menú de acceso a la herramienta para crear estilos desde la interfaz de administración web de GeoServer.

Utilizando la herramienta para la creación de estilos se creó uno propio para cada capa, así el usuario tendrá un acercamiento visual a la información espacial que se presenta (Figura 4.5)

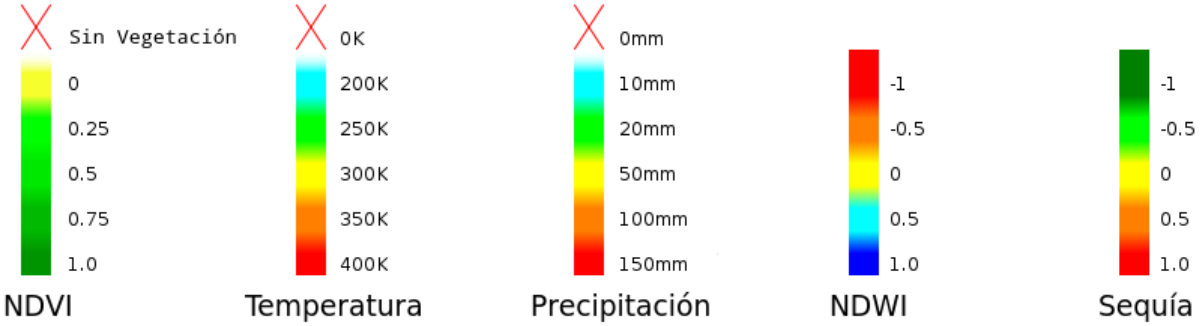


Figura 4.5: Ejemplo de estilos creados para los índices descargados del WebGIS ISAGRO.

4.4.1.3. Capas de Información

La información satelital aparece como una herramienta importante para estudios ambientales. Ésta provee detalle de las condiciones presentes en el lugar de interés al momento de toma, permitiendo derivar de ellas productos de tipo ráster o vectorial que ayuden a la comprensión de fenómenos que se presentan en determinado sitio.

Dado que las capas aún se encuentran en procesamiento, se utilizaron series de datos como ejemplo tomados del WebGIS ISAGRO. Las series de datos fueron remuestreadas a 1km de resolución espacial y su nomenclatura fue definida teniendo en cuenta como deberán nombrarse los ráster una vez termine su procesamiento (se presentará en detalle durante la descripción de cada producto). Lo anterior permitió desarrollar los *scripts* para automatizar los procesos de carga de capas al servidor de mapas (Anexo A), así como estructurar los complementos desarrollados para el aplicativo cliente y ejecutar las pruebas necesarias con cada uno. A continuación se presenta una descripción de las capas utilizadas:

- **Capas Ráster:** Para la muestra de los índices, se tomaron productos MODIS (*tile*[1] h12v12) que cubren parte de la provincia de Córdoba y los cuales se encuentran a disposición en el WebGIS ISAGRO, así como un MDE y una capa ráster de coberturas que se detallan a continuación:

[1]Para las imágenes MODIS se establece una grilla de 18 filas por 36 columnas de 10 grados de resolución, donde cada celda se denomina tesela (*tile* en inglés)

- **Índice Diferencial de Vegetación Normalizado (NDVI):**

  El NDVI (Normalized Difference Vegetation Index) propuesto por Rouse Jr *et al.* (1973), es un índice que se basa en el hecho que la clorofila absorbe radiación en las longitudes de onda del rojo, mientras que dispersa radiación en el infrarrojo cercano (NIR, por sus siglas en inglés). De esta manera, el índice toma la respuesta espectral de estas bandas para estimar la cantidad y estado de la vegetación. Es un índice ampliamente utilizado en estudios ambientales (e.g., Madonsela *et al.*, 2018; Weber *et al.*, 2018) dado que la vegetación influencia fuertemente la distribución de especies animales y su dinámica (Pettorelli *et al.*, 2005).

- **Índice Diferencial de Agua Normalizado (NDWI):**

  El NDWI (Normalized Difference Water Index) propuesto por Gao (1996), es un índice derivado de las bandas del infrarrojo de onda corta (SWIR, por sus siglas en inglés), donde la reflectancia refleja los cambios tanto en el contenido de agua de la vegetación como en la estructura de las copas, y NIR, que se ve afectada por la estructura interna de la hoja y el contenido de materia seca (Takeuchi & Gonzalez, 2009). Por lo tanto, este indicador es de gran importancia para determinar cómo las condiciones de la vegetación pueden afectar la dinámica y distribución de especies (e.g., Obsomer *et al.*, 2012; Wisz *et al.*, 2008).

- **Índice Diferencial de Sequía Normalizado (NDDI):**

  El NDDI (Normalized Difference Drought Index) propuesto por Gu *et al.* (2007), es un índice que surge de la relación entre el NDVI y el NDWI (Ec. 4.4.1). Es un fuerte indicador de las condiciones de sequía, uno de los fenómenos que contribuye notablemente a la degradación del suelo (Azurmendi, 2012). Este índice ha sido implementado en diversos estudios ambientales que utilizan series temporales para el análisis de los cambios que se presentan en el paisaje (e.g., Biswas, 2010; Gu *et al.*, 2007; Gulácsi & Kovács, 2015).

$$NDDI = \frac{NDVI - NDWI}{NDVI + NDWI} \tag{4.4.1}$$

- **Temperatura Superficial Terrestre LST:**

  El índice LST (Land Surface Temperature), es un parámetro de importancia en la reproducción de especies (Fischer, 1960). Por esta razón ha sido utilizado en diversos estudios que intentan determinar su papel como mecanismo que influencia este tipo de comportamiento (e.g., Beerling, 1993; Woodward, 1987).

  Entre los productos derivados de datos MODIS, se encuentra el LSTDay que proporciona información de temperatura superficial terrestre, apareciendo como alternativa para el desarrollo de este tipo de estudios.

- **Precipitación:**

  La precipitación es un parámetro que tiene relevancia en el crecimiento y distribución de la vegetación y de las especies, sirviendo como insumo para diferentes tipos de estudios que incorporan variables de humedad del sitio (e.g., Kawabata *et al.*, 2001; McCain & Colwell, 2011)

- **Modelo Digital de Elevación:**

  Las variables topográficas son ampliamente utilizadas como predictores en estudios ambientales como la modelación de distribución de especies, dado que se asocia con efectos microclimáticos relacionados con disponibilidad de agua, temperatura y radiación solar que incide sobre el área (Miller, 2010). El MDE utilizado fue el Shuttle Radar Topography Mission (SRTM)[2] de 90m remuestreado a 1km, el cual fue descargado de la página del Consortium for Spatial Information (CGIAR CSI).

- **Capas Vectoriales:**

  Además de las variables de clima y topografía mencionadas, existen otras en formato vectorial (e.g., ríos, lagos, usos de suelo, etc), que contribuyen a la descripción de la proximidad a ciertas perturbaciones o recursos importantes (e.g., Lane *et al.*, 2001; Osborne *et al.*, 2001). Para el WebGIS se pusieron a disposición capas de:

[2]https://www2.jpl.nasa.gov/srtm/

- **Ríos y Lagunas:**

  Estas capas vectoriales se descargaron de la página Natural Earth Data[3].

- **Límites Administrativos:**

  Corresponden a los límites de País, Estado/Provincia o Departamento (Adm0, Adm1 y Adm2), tomados de la página Global Administrative Areas[4].

- **Ecorregiones Terrestres:**

  La página de la ONG The Nature Conservancy[5] pone a disposición diversas capas vectoriales, de las cuales se tomaron las correspondientes a las de regiones ecológicas.

4.4.1.4. Nomenclatura

Con el fin de estandarizar la forma en que se nombran las capas de cada índice y asegurar un correcto funcionamiento de los complementos creados para la aplicación cliente, una vez descargadas las capas de muestra, se procedió a definir la nomenclatura que deberán utilizar los productos que se pongan a disposición en el servidor de mapas. Dicha nomenclatura, para todos los casos, sigue la estructura que se muestra a en la Figura 4.6:

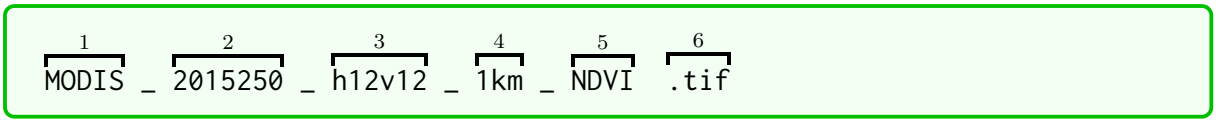


Figura 4.6: Ejemplo de nomenclatura de las capas ráster.

Donde:

1. ***Sensor:*** Su objetivo es brindar al usuario una noción acerca de la procedencia de los datos que va a utilizar, como la calidad al momento de toma (en caso de sensores que se conozca que presentan fallas) o si el sensor está aún activo y seguirá capturando datos.

2. ***Fecha:*** Corresponde al día de toma de la imagen. Se compone del año seguido del día juliano.

[3]http://www.naturalearthdata.com/
[4]http://gadm.org/version2
[5]http://maps.tnc.org/gis_data.html

3. ***Dominio Espacial:*** Se propuso un espacio para describir la cobertura espacial de las imágenes para dar al usuario una idea de la zona que está trabajando, por ejemplo, en el caso de las capas de prueba de NDVI, el valor de este campo es "h12v12" que es el tile MODIS que corresponde a su ubicación geográfica.

4. ***Resolución espacial:*** Parámetro que informa al usuario el tamaño del píxel con que se está trabajando.

5. ***Índice:*** Valor que contiene el nombre del producto a utilizar.

6. ***Formato de las imágenes***

En el caso de las capas vectoriales, el MDE y las coberturas, no se utilizó la nomenclatura descrita, dado que corresponden a un único resultado que no se actualizará constantemente.

4.4.2. Cliente WebGIS

La Opengeo Suite ofrece un entorno de trabajo basado en GXP, que posee un archivo principal llamado app.js donde se hacen las llamadas a los complementos que se desean utilizar para la personalización del cliente. Estos complementos se basan en librerías como OpenLayers o GeoExt y, pueden ser proporcionados por la suite o ser creados por el desarrollador incorporando a su vez funciones de otras librerías de código abierto. El acceso al aplicativo cliente es libre, esto hace que cualquier persona pueda utilizar sus funcionalidades sin previa autenticación.

4.5. Complementos Adicionados

Para la personalización del aplicativo cliente, se adicionaron plugins proporcionados por la suite con los que se efectúan consultas básicas como zoom a un área de interés o retornar la información contenida en un píxel o feature de una capa. También, se desarrollaron complementos para visualización (grilla de coordenadas) o manipulación de los datos (recorte, descarga, calculadora, perfil y regresión lineal).

Los complementos hacen uso de un *script* desarrollado en Python que brinda tres opciones para definir la ip del servidor (Anexo B), la primera retorna la ip pública desde la página "http://www.cualesmiip.com/", la segunda utiliza la librería socket para establecer la ip local y la tercera permite definir manualmente la ip del servidor. De esta manera el administrador habilitará la opción que considere adecuada y ya no será necesario modificar cada complemento y recompliar el aplicativo cliente en caso de un cambio de ip del servidor o dominio.

La Figura 4.7 presenta un diagrama de la configuración del aplicativo donde los complementos disponibles utilizados hacen referencia a aquellos que proporciona la Opengeo Suite, los cuales no necesariamente vienen cargados en la plantilla, sino que su adición se hace desde el archivo app.js. Adicionalmente se muestran los complementos desarrollados que consisten en los archivos nuevos creados para ofrecer funcionalidades adicionales al visor, los cuales hacen uso de dependencias para su funcionamiento que en algunos casos corresponden a herramientas dentro de las librerías como OpenLayers2, *scripts* en Python o *software* utilizado para el manejo de información desde el lado del servidor.

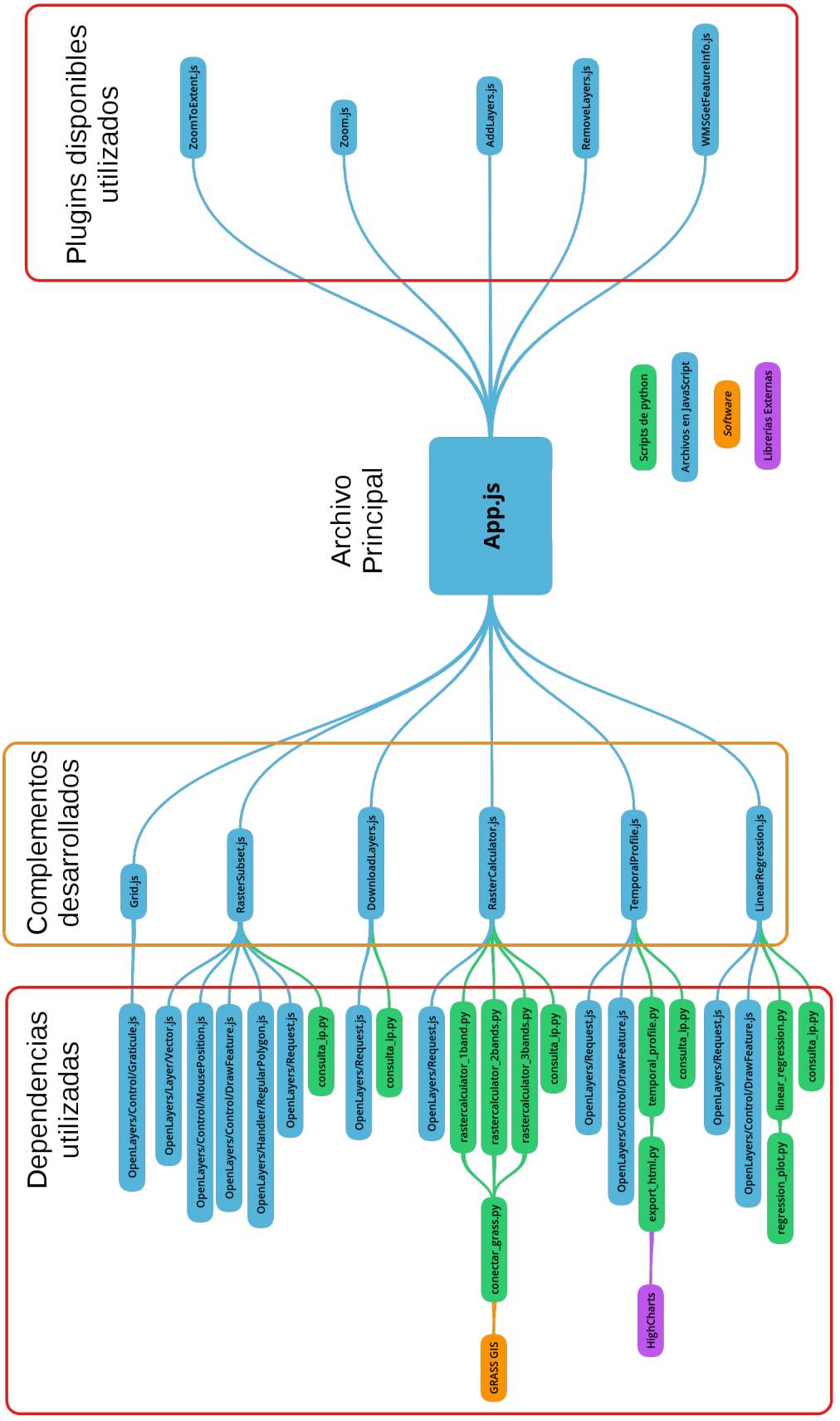


Figura 4.7: Diagrama general de los complementos utilizados.

4.5.1. Configuración de Funcionalidades Suministradas por la Opengeo Suite

A continuación se describe la configuración de aquellos plugins que ofrece el framework utilizado y que en la mayoría de los casos no vienen incorporados al aplicativo cliente:

- **Importar Capas:**

  La herramienta para cargar las capas en el aplicativo cliente, viene por defecto en la plantilla GXP proporcionada por la suite. En este caso, el plugin utiliza el protocolo WMS para consultar y cargar la información procedente de las fuentes configuradas en el archivo principal (app.js), en la sección ***sources*** (Figura 4.8). Estas fuentes corresponden a espacios de trabajo definidos en el servidor de mapas y se definen como se muestra a continuación:

```
1 //layer sources
2 sources: {
3     NDWI: {
4         ptype: "gxp_wmscsource",
5         url: "/geoserver/NDWI/wms?",
6         version: "1.1.0"
7     }, ...
8 }
```

Figura 4.8: Ejemplo de configuración del espacio de trabajo NDWI.

Después de configurar cada espacio de trabajo que se quiere visualizar, es posible comenzar a cargar las capas de interés. Para esto, al hacer clic en el botón  se despliega un menú con cada uno de los espacios de trabajo disponibles. Al seleccionarlos, se visualizarán las capas almacenadas en ellos, las cuales podrán ser adicionadas al mapa con un doble clic sobre cada una o haciendo uso del botón **Añadir Capas** en la parte inferior de la ventana (Figura 4.9).

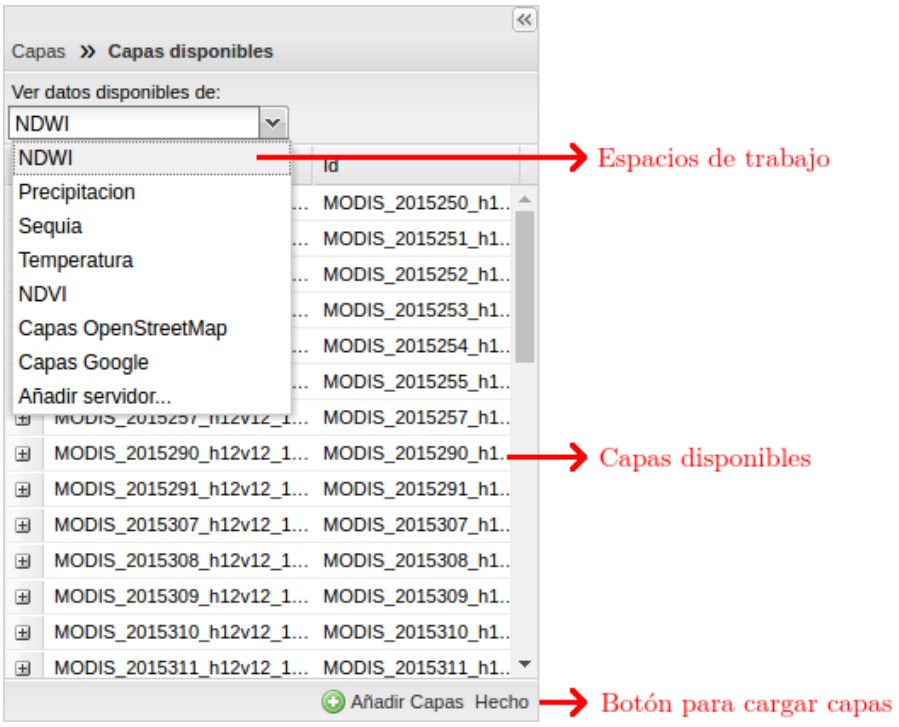


Figura 4.9: Ventana para adicionar capas al visor.

- **Consultar Información:**

  Este plugin aunque se encuentra por defecto al crear la plantilla GXP no viene cargado en el cliente. Hace uso de la dependencia WMSGetFeatureInfo para realizar la consulta de la información disponible en el servidor de mapas y retornar al aplicativo cliente las características de los datos. Al igual que los demás plugins, se adiciona desde el archivo principal app.js (Figura 4.10), agregando primero la dependencia requerida y posteriormente haciendo la llamada mediante la variable "ptype" en la sección *tools* del *script*, como se muestra a continuación:

```
1 * Dependencias:
2 * @require plugins/WMSGetFeatureInfo.js
3
4 // configuración de los plugins para la aplicación
5 Consulta por clic de un punto específico
6 tools: [{
7     ptype: "gxp_wmsgetfeatureinfo",
8     outputConfig: {width: 400},
```

```
9       actionTarget: {
10          target: "map.tbar",
11          index: 1
12      }
13  },...]
```

Figura 4.10: Configuración del plugin para consulta de información.

Una vez configurado el plugin, se podrá visualizar en la barra de herramientas el botón  que utiliza el protocolo WMS para retornar al cliente una ventana con la información de las capas activas que se superponen espacialmente en el punto seleccionado (Figura 4.11).

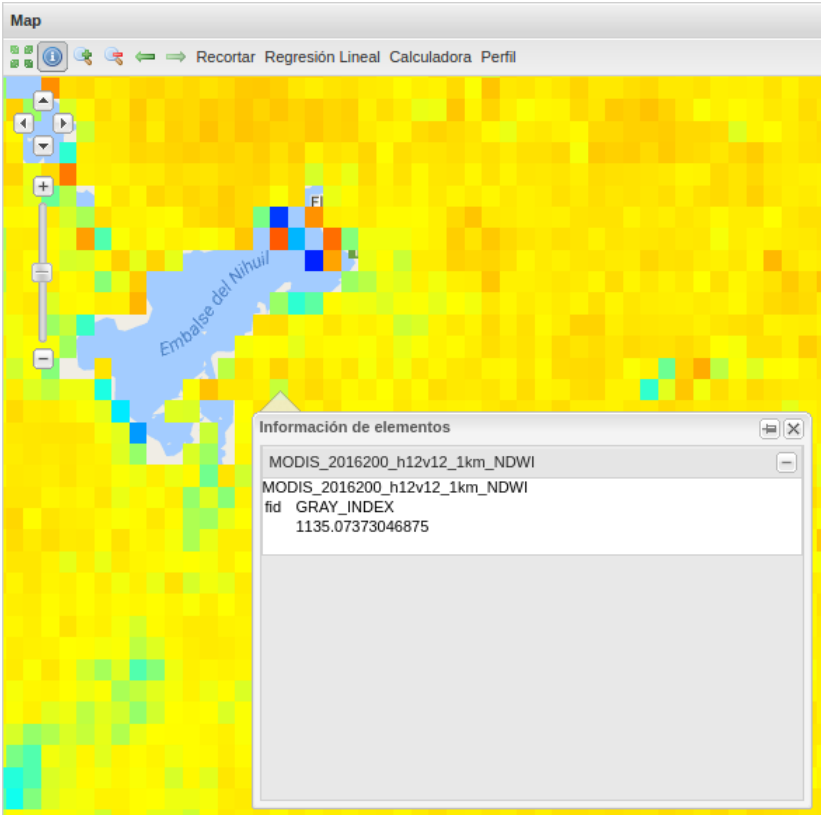


Figura 4.11: Consulta por clic de un punto específico.

Este plugin responde a los requerimientos planteados de consultar información por píxel de las capas ráster, así como atributos de las capas vectoriales.

- **Tabla de Atributos:**

  Este plugin hace uso de las dependencias FeatureManager.js y FeatureGrid.js para

entregar al aplicativo cliente la información correspondiente a la tabla de atributos de las capas vectoriales seleccionadas. Es suministrado una vez se crea la plantilla y requiere, para su adición, definir un panel inferior que contendrá la tabla (Figura 4.12).

```
1  * Dependencias:
2  * @require plugins/FeatureManager.js
3  * @require plugins/FeatureGrid.js
4 // configuración del panel que contiene de la tabla
5     items: [...
6     {
7         id: "south",
8         region: "south",
9         xtype: "gxp_crumbpanel",
10        collapsible: true,
11        collapsed: true,
12        split: true,
13        border: true,
14        height: 200
15    }...
16    ],
17 // configuración del plugin
18    tools: [...
19    {
20        ptype: "gxp_featuremanager",
21        id: "shp_manager",
22        paging: false,
23        autoSetLayer: true,
24        autoLoadFeatures: true,
25        autoZoomPage:true
26    }, {
27        ptype: "gxp_featuregrid",
28        featureManager: "shp_manager",
29        outputConfig: {
30            loadMask: true
31        },
32        outputTarget: "south"
33    },...
34    ]
```

Figura 4.12: Configuración del plugin para visualizar la tabla de atributos de las capas vectoriales.

La tabla muestra los atributos de las capas vectoriales visibles cuando son seleccionadas en el aplicativo cliente. Adicionalmente, la herramienta cuenta con un botón con el cual el usuario puede visualizar en el mapa alguna característica que se marque en la tabla (Figura 4.13).

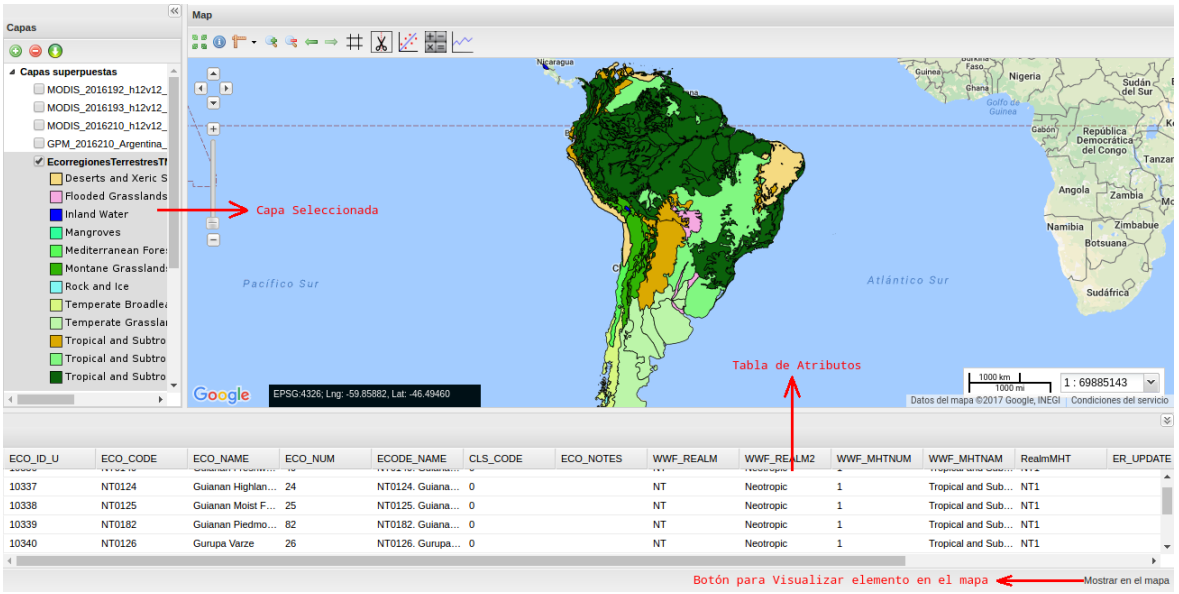


Figura 4.13: Ejemplo de la tabla de atributos para la capa vectorial de ecorregiones terrestres.

Este plugin complementa los requerimientos de consulta de información de las capas vectoriales.

4.5.2. Implementación de Nuevas Funcionalidades

A continuación se presentan aquellas funcionalidades que se implementaron para dar respuesta a los requerimientos faltantes:

- **Grilla de Coordenadas:**

  Se desarrolló el complemento Grid.js que hace uso de la dependencia "OpenLayers/Control/Graticule.js" proporcionado por la librería OpenLayers2 (Anexo C) que genera una retícula sobre el mapa con líneas etiquetadas que representan los meridianos y paralelos (Figura 4.14). Viene por defecto desactivada, sin embargo, el usuario puede visualizarla haciendo uso del botón  .

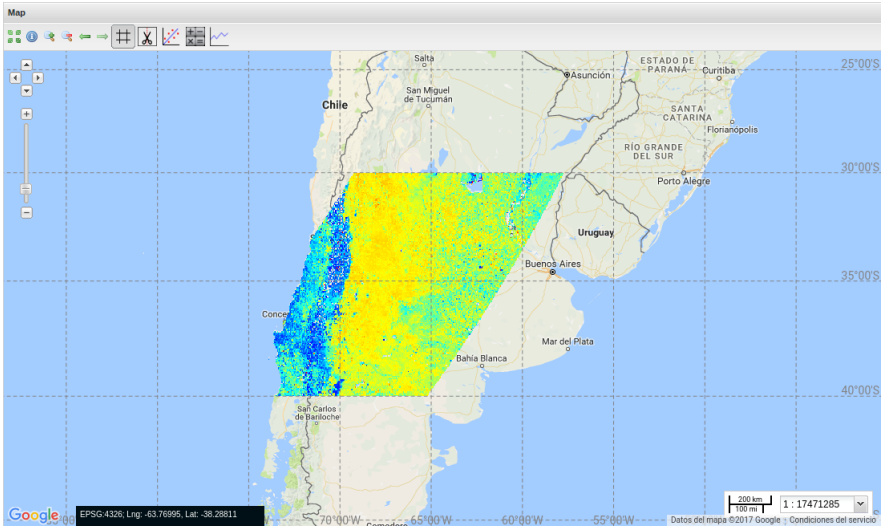
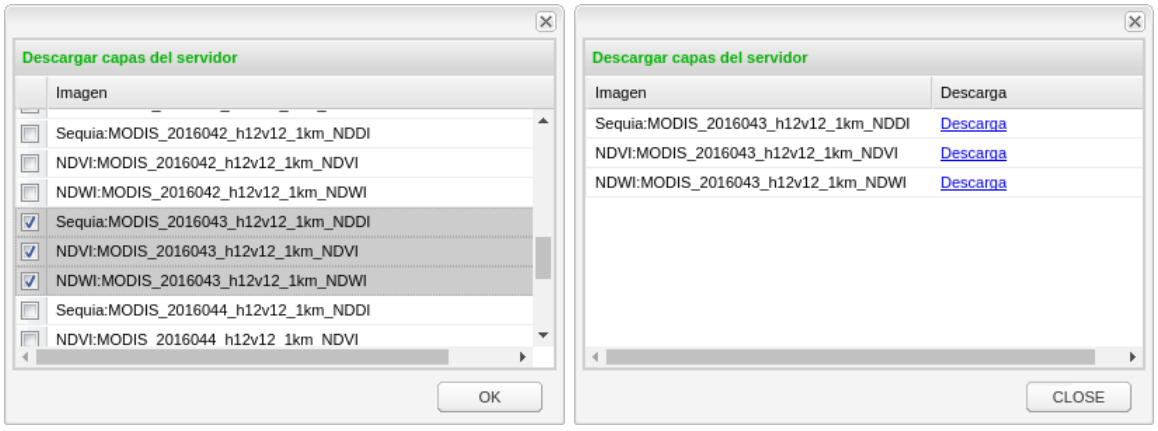


Figura 4.14: Grilla de coordenadas.

- **Descargar Capas:**

  El complemento DownloadLayers.js se activa mediante el botón  , proporcionando al usuario acceso a los datos ráster y vectoriales contenidos en el servidor teniendo en cuenta la totalidad de su extensión geográfica (Anexo D), retornando un enlace de descarga por cada elemento seleccionado por este (Figura 4.15).



(a) Selección de capas. (b) Enlaces de descarga.

Figura 4.15: Secuencia para realizar descargas.

- **Recorte:**

  El WebGIS está pensado para almacenar mosaicos de Suramérica, de esta manera se

garantiza un tratamiento previo en las zonas de empalme de las imágenes, facilitando al usuario final el preprocesamiento de la información. Sin embargo, no a todos los usuarios les interesa el área completa del mosaico sino aquel lugar donde se lleva a cabo el estudio. Por esta razón y con el objetivo de dar respuesta al requerimiento de obtener el recorte de capas ráster a partir de selección de un recuadro en el visor, se desarrolló un complemento cuyo funcionamiento es similar al de la descarga, con la diferencia que permite al usuario definir la extensión geográfica de las capas que desea descargar.

Para comenzar, se activa el botón  que habilita el cursor para la selección del aŕea de interés. Posteriormente, se hace uso de la librería OpenLayers para extraer las coordenadas de los lados del recuadro (izquierda, derecha, superior e inferior), adicionalmente crea un listado de las imágenes disponibles en el servidor de mapas mediante una consulta GetCapabilities del protocolo WMS, que muestra al usuario en una ventana emergente, donde este podrá seleccionar las que son de su interés (Figura 4.16).

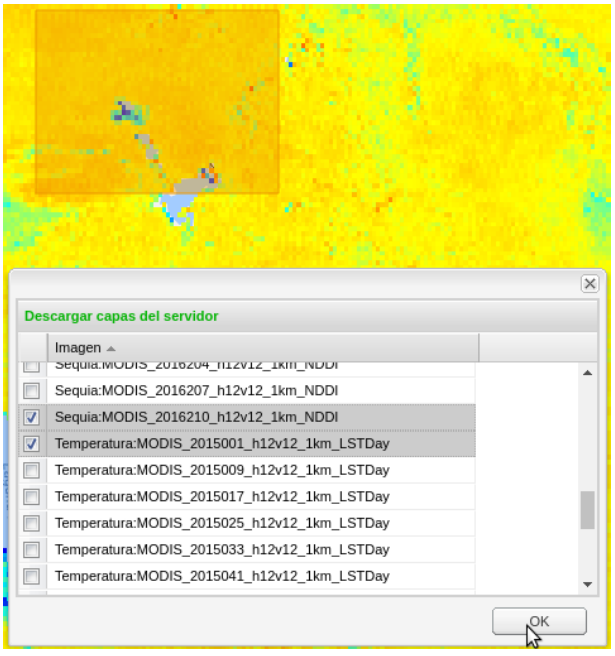


Figura 4.16: Selección del área y las capas de interés para su descarga.

Al producirse la selección de imágenes, el complemento toma la información de las

capas y las coordenadas del recuadro para construir, por cada una, un enlace que utiliza el protocolo WCS para hacer una solicitud al servidor de dicha capa y su respectiva extensión. Se devuelve entonces una nueva ventana emergente con dos columnas, en la primera se listan los nombres de las capas ráster seleccionadas y en la segunda se muestran los enlaces de descarga correspondientes (Figura 4.17).

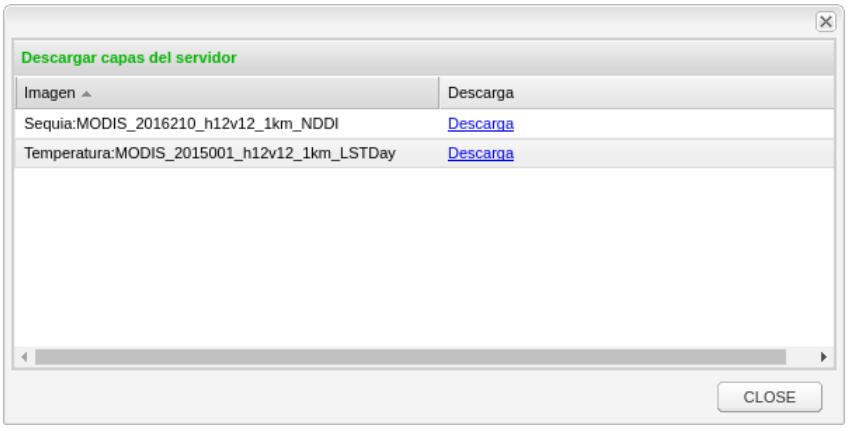


Figura 4.17: Retorno de la consulta y enlaces de descarga.

- **Perfil Temporal:**

  Existen variables como la temperatura y la vegetación cuyo comportamiento presenta ciclos temporales y que pueden contribuir a la explicación de fenómenos ambientales (e.g., Bechtel, 2012; Yuan & Bauer, 2007). Por esta razón se desarrolló una funcionalidad que permitiera al usuario tener un primer acercamiento al comportamiento temporal de determinado índice en un lugar dado, brindando un acercamiento visual a medidas descriptivas básicas (tendencia, estacionalidad y outliers). Además, esta funcionalidad proporciona algunos estadísticos de la serie como valor máximo, mínimo, media aritmética, mediana y desviación estándar como información complementaria de la serie de datos.

  Para su ejecución se utiliza el botón  , el cual habilita el cursor para seleccionar un punto de interés sobre el mapa. Entonces se despliega una ventana donde se elige entre los diferentes índices disponibles (Figura 4.18).

Figura 4.18: Menú para selección de índices.

Una vez seleccionado el índice, el complemento hace una petición al servidor proporcionando los valores de los parámetros requeridos (coordenadas del píxel e índice), éste ejecuta un *script* de Python que para cada imagen extrae el valor del píxel en el punto dado con su respectiva fecha y almacena los datos en dos vectores (Anexo E). Los valores extraídos son utilizados para calcular los estadísticos (máximo, mínimo, media, mediana y desviación estándar) y construir un archivo de salida en formato html que utiliza la librería HighCharts para realizar el gráfico (Anexo F). El servidor retorna el resultado a la aplicación cliente, que lo carga en una nueva ventana, siendo disponible para su impresión o descarga en diferentes formatos (PNG, PDF, SVG y JPEG), además de proporcionar un botón para la descarga de un archivo de texto con los datos de la serie. En la Figura 4.19 se muestra la serie de temperatura superficial comprendida entre enero de 2015 y julio de 2016 (latitud: -34.11°, longitud: -69.29°). En el gráfico que se puede observar la predominancia del ciclo anual que viene condicionado por la estación del año donde valores más altos de temperatura corresponden a verano (entre diciembre y marzo), mientras que valores bajos vienen dados por el invierno (entre junio y septiembre).

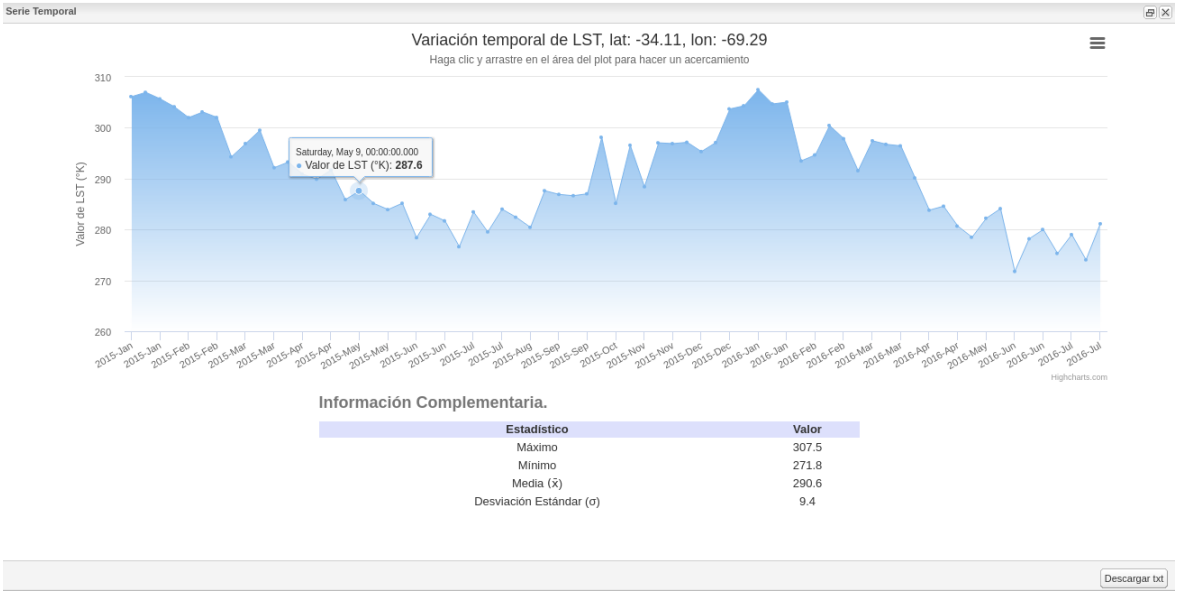


Figura 4.19: Perfil temporal de temperatura superficial para las coordenadas Lat: -34.11°, Lon: -69.29°.

- **Regresión Lineal:**

  Es una herramienta de gran utilidad en estudios de tipo ambiental (e.g., Ahmed & Akter, 2017; Julien & Sobrino, 2009; Olivera-Guerra *et al.*, 2017; Yuan & Bauer, 2007). Su propósito es ajustar la relación de dependencia entre dos variables seleccionadas por el usuario directamente en el aplicativo cliente, de esta manera podrá explorar las relaciones entre diferentes capas y para cualquier zona de Suramérica sin necesidad de su descarga.

  Utiliza la función OpenLayers.Request.GET(), que mediante una petición de tipo HTTP GET entrega al servidor los parámetros necesarios para la ejecución del *script* "linear_regression.py" (Anexo G). El servidor retorna al aplicativo cliente el gráfico de la dispersión con su respectiva línea de ajuste, así como la información adicional correspondiente (pendiente, intercepto y coeficiente de correlación lineal).

  Para su ejecución se utiliza el botón  , el cual permite dibujar un recuadro que contenga el área de interés e inmediatamente despliega una ventana con el listado de índices disponibles que al seleccionarlos activan la información de las capas que cada uno contiene (Figura 4.20).

Figura 4.20: Ventana para selección de capas.

A modo de ejemplo se presenta el uso de esta funcionalidad para estimar la relación entre el LST (LSTDay) y el NDVI para el 14 de septiembre de 2015 (día juliano 257) (Figura 4.21).

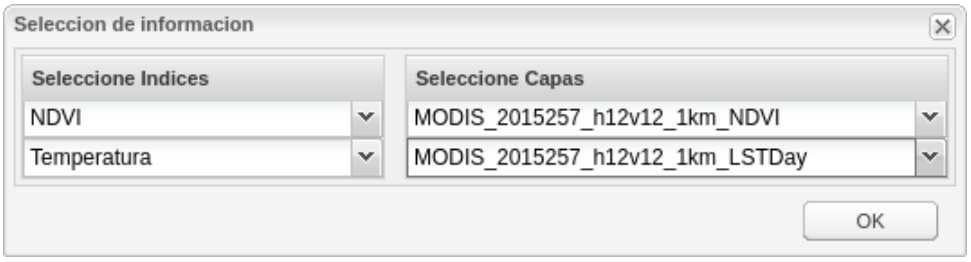


Figura 4.21: Selección de capas para la regresión lineal.

Después de la selección, el complemento efectúa la petición al servidor con las coordenadas que definen el rectángulo (izquierda, derecha, superior e inferior), así como las bandas seleccionadas. Desde el lado del servidor se cargan los recortes de cada banda y se genera un vector de datos para cada capa, que son utilizados para calcular el coeficiente de correlación lineal y los parámetros del ajuste de la recta haciendo uso de la librería Scipy (Jones *et al.*, 2001).

Los puntos que conforman la dispersión, así como la línea de ajuste, son graficados utilizando la librería Matplotlib (Anexo H) creando un archivo de salida en formato png. Además de la dispersión, se proporciona información complementaria del valor de correlación lineal (r) y la ecuación de la recta. El nombre del archivo es enviado al aplicativo cliente que carga esta salida en una nueva ventana emergente, brindando al usuario la opción de descargar tanto la imagen (en formato PNG) como un archivo

de texto que contiene los vectores utilizados para cada capa con sus respectivas coordenadas. La Figura 4.22 muestra la salida de la regresión realizada para las capas de LST y NDVI seleccionadas.

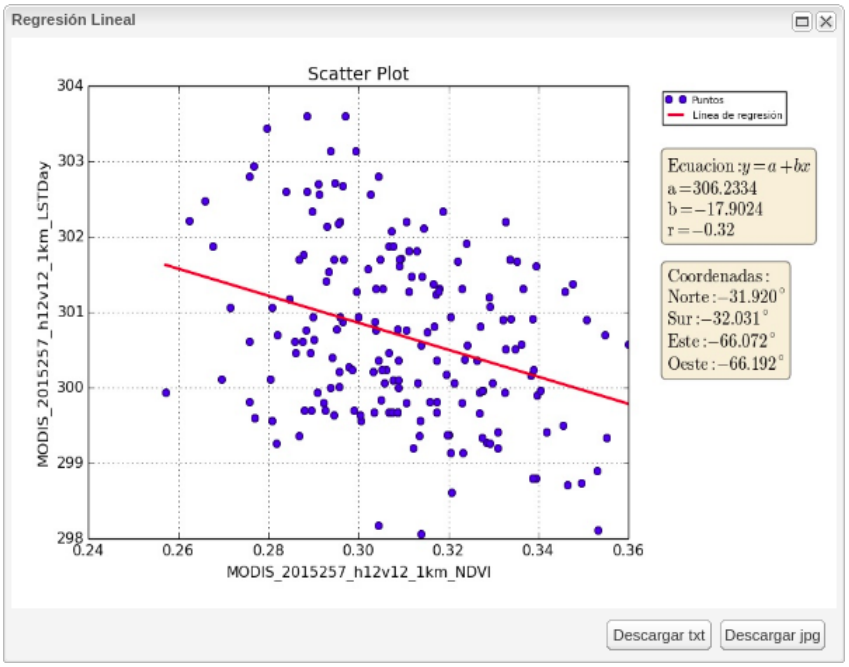


Figura 4.22: Ejemplo, Regresión lineal Temperatura (LST) vs Vegetación (NDVI).

La correlación es acorde con lo esperado si se tiene en cuenta que la temperatura superficial (LST) tiene una correlación negativa con el NDVI (Yuan & Bauer, 2007). Sin embargo, un análisis más profundo del resultado corresponde al usuario, dado que existen factores como la estacionalidad y la topografía que pueden intervenir en el resultado obtenido.

- **Calculadora Ráster:**

  Este complemento lleva a cabo operaciones simples con una, dos o tres capas ráster contenidas en el servidor. Integra el *software* GRASS GIS desde el lado del servidor para hacer uso de su función r.mapcalc (Anexo I), brindando así una opción de procesamiento a través del aplicativo cliente que permita derivar, a partir de la información disponible en el servidor, nuevos índices que sean de interés al usuario final. De esta manera, el usuario tiene la posibilidad de realizar un procesamiento

previo de los datos, evitando en ocasiones la descarga de imágenes para un posterior proceso en un SIG de escritorio.

Para su ejecución el usuario debe utilizar el botón  , que habilita el cursor para dibujar un recuadro sobre el mapa que contenga el área de interés. En ese momento se despliega una primer ventana (Figura 4.23) que permite determinar el número de imágenes a procesar (1, 2 o 3).

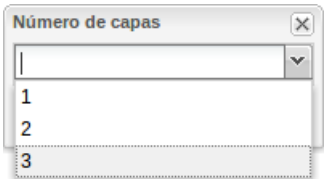


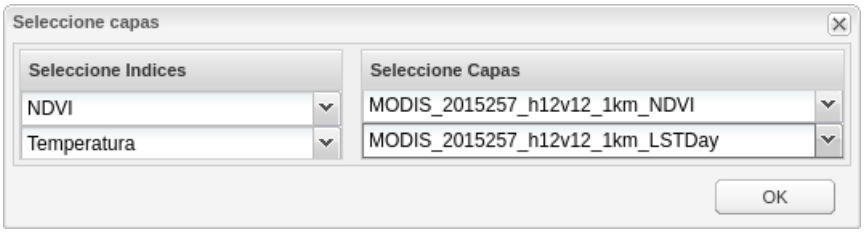
Figura 4.23: Ventana para seleccionar el número de capas.

En este caso se tomará como ejemplo el Índice de Susceptibilidad de Incendio (FSI) según lo descrito por Peng *et al.* (2007), por lo que se asumirán las constantes descritas y la fórmula empírica descrita en el artículo para estimar el contenido de humedad del combustible (FMC) (Ec. 4.5.1).

$$FMC = 49906 * \frac{NDVI}{LST} + 92 \tag{4.5.1}$$

Se realizó entonces la selección de dos capas correspondientes a NDVI y LST para el 14 de septiembre de 2015 (día juliano 257) (Figura 4.24).

(a) Ventana para selección de capas.



(b) Capas seleccionadas.

Figura 4.24: Secuencia para la selección de capas.

Una vez realizada la selección, se despliega una casilla que funciona como entrada de texto para escribir la ecuación que se desea ejecutar. En este caso, se reemplazaron las variables correspondientes en cada ecuación para generar una sola ecuación que permita estimar el FSI (Figura 4.25) según lo propuesto por Peng *et al.* (2007).

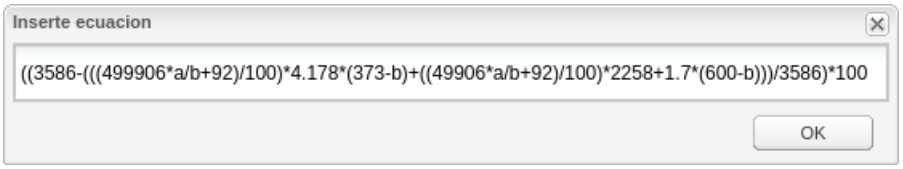


Figura 4.25: Ecuación para el cálculo del FSI.

Una vez se introduce la ecuación, el complemento ejecuta el *script* de Python con los parámetros suministrados (Anexo J). Python se conecta al *software* GRASS GIS y hace uso de los comandos g.region para definir la región de acuerdo a las coordenadas suministradas y r.mapcalc para llevar a cabo la operación con las capas seleccionadas, generando como resultado un nuevo ráster en formato geotiff en el sistema de coordenadas WGS84.

En caso de ser exitoso el proceso, el *script* de Python retorna el nombre de la capa

de salida, la cual se entregará al cliente en forma de un enlace que permitirá su descarga (Figura 4.26(a)). De lo contrario, el servidor devuelve un mensaje de error donde solicita al usuario la revisión de parámetros de entrada (Figura 4.26(b)).

(a) Enlace de descarga de la imagen resultado.

(b) Ventana de error.

Figura 4.26: Posibles respuestas del servidor al ejecutar la calculadora ráster.

Como resultado de lo anterior se obtuvo una imagen de FSI con valores de susceptibilidad entre el 19 % y el 21 % (Figura 4.27), implicando un bajo riesgo para la zona, lo que es coherente teniendo en cuenta que las coordenadas corresponden a la Provincia de Córdoba (Argentina) y que para esa fecha está terminando el invierno. Sin embargo, cabe aclarar que la validación de los resultados corresponden al usuario final del WebGIS y que estos dependerán del estudio que lleven a cabo y la zona elegida para realizarlo.

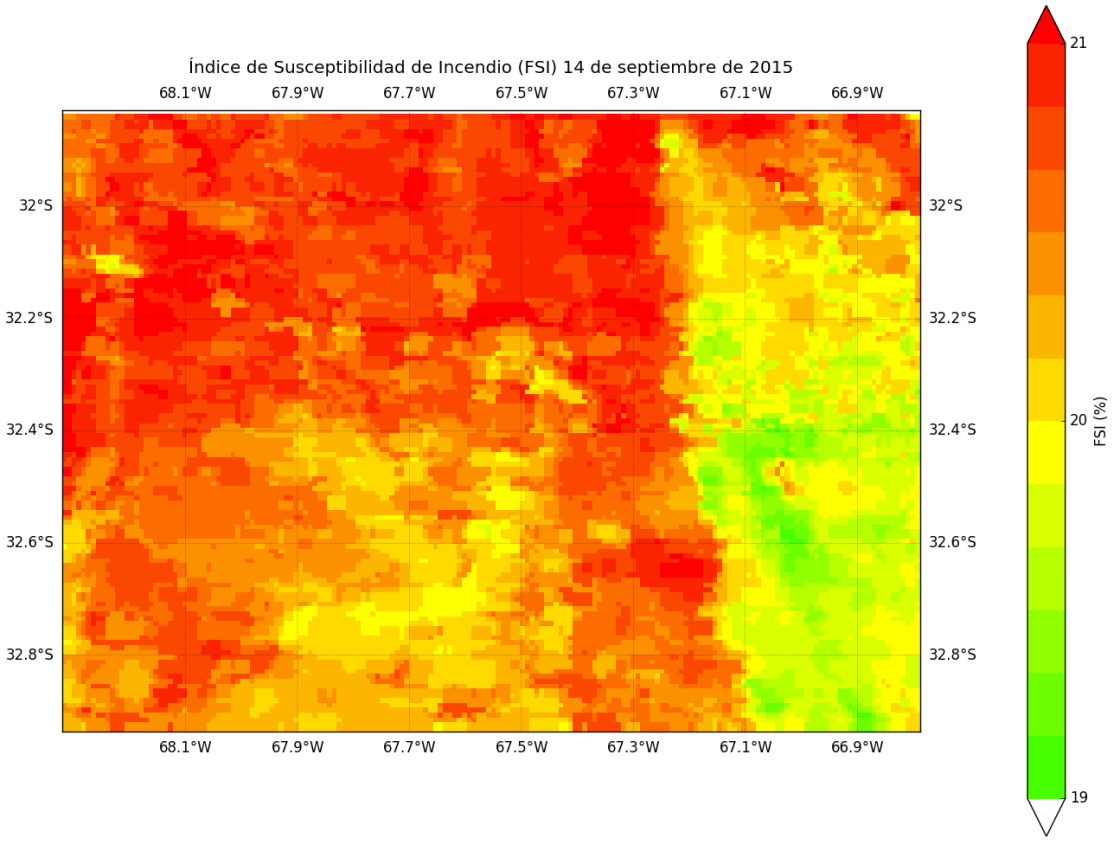


Figura 4.27: Resultado para el cálculo del FSI a partir de la calculadora ráster incorporada en el WebGIS.

Capítulo 5

Resultados y Validación de Funcionalidades

En este capítulo se presentan las pruebas realizadas para verificar el funcionamiento de las funcionalidades obtenidas e implementadas para el aplicativo cliente, comparando los resultados obtenidos mediante los complementos de regresión lineal, perfil temporal y calculadora implementados, con *scripts* que siguen la misma metodología de trabajo y que, al igual que para el WebGIS, han sido construidos teniendo en cuenta herramientas de código abierto (ver Tabla 5.1). Se utilizó el lenguaje R por ser una herramienta que proporciona una gran variedad de técnicas gráficas y estadísticas para el procesamiento avanzado de datos (R Development Core Team, 2018). Asimismo, se utilizaron los programas GRASS GIS (cuando no interviene en el proceso) y QGIS dado que además de ser herramientas de uso libre, permiten su ejecución desde scripts desarrollados en Python facilitando así la ejecución de rutinas de prueba.

Complemento	Lenguaje/*software* utilizado	Lenguaje/*software* comparación
Regresión Lineal	Python	R, GRASS GIS
Perfil Temporal	Python	R, GRASS GIS
Calculadora Ráster	Python, GRASS GIS	R, QGIS

Tabla 5.1: Lenguajes y *software* utilizados para construcción y pruebas de los complementos desarrollados

Las pruebas se desarrollaron teniendo en cuenta un mismo entorno de trabajo (ver Tabla 5.2), llevando a cabo mediciones de los tiempos de ejecución de los diferentes procesos que cada uno realiza, con el fin de tener una noción del desempeño de dichas rutinas, sin implicar esto un análisis estadístico a profundidad por encontrarse por fuera de los alcances del proyecto.

Item	Descripción
Procesadores	4
Memoria RAM	4GB
Sistema Operativo	Ubuntu 14.04
Kernel	Linux
Versión del Kernel	4.4.0-79

Tabla 5.2: Entorno para el desarrollo de las pruebas

5.1. Recorte

Para realizar la prueba se tuvieron en cuenta los parámetros del tipo de solicitud *WebCoverage* (ver Tabla 2.5). Se construyó entonces una petición que utiliza la información que se muestra en la Figura 5.1 para obtener el recorte de una imagen de NDDI:

```
servidor                                  servicio requerido  solicitud
http://localhost:8080/geoserver/wcs ? service=WCS & request=GetCoverage
   versión del servicio                   índice y capa
& version=2.0.1 & coverageId=Sequia:MODIS_2016204_h12v12_1km_NDDI &

                    Longitud mínima y máxima del recorte
subset=http://www.opengis.net/def/axis/OGC/0/Long(-68.912,-63.638) &

                    Latitud mínima y máxima del recorte
subset=http://www.opengis.net/def/axis/OGC/0/Lat(-36.204,-32.875) &

  formato de salida     Sistema de referencia
format=geotiff & crs=EPSG:4326
```

Figura 5.1: Línea de prueba utilizada para el recorte.

El resultado obtenido mediante la prueba, se verificó haciendo uso del programa gdalinfo. Este programa retorna la información de metadatos de la imagen que se consulta, permitiendo llevar a cabo una revisión de datos como el sistema de referencia y tamaño del píxel, para así comprobar que son iguales a los de capa original (Figura 5.2). Además proporciona el valor de las coordenadas de las esquinas del ráster de salida, para verificar que el recorte se hizo de acuerdo a la extensión geográfica suministrada.

```
Driver: GTiff/GeoTIFF
Files: Sequia-MODIS_2016204_h12v12_1km_NDDI.tif
Size is 611, 385
Coordinate System is:
GEOGCS[‘‘WGS 84’’,
     DATUM[‘‘WGS_1984’’,
          SPHEROID[‘‘WGS 84’’,6378137,298.257223563,
               AUTHORITY[‘‘EPSG’’,‘‘7030’’]],
          AUTHORITY[‘‘EPSG’’,‘‘6326’’]],
     PRIMEM[‘‘Greenwich’’,0],
     UNIT[‘‘degree’’,0.0174532925199433],
     AUTHORITY[‘‘EPSG‘‘,’’4326‘‘]]
Origin = (-68.912665515769604,-32.878573239197863)
Pixel Size = (0.008626704385179,-0.008626704385179)
Metadata:
     AREA_OR_POINT=Area
     TIFFTAG_RESOLUTIONUNIT=1 (unitless)
     TIFFTAG_XRESOLUTION=1
     TIFFTAG_YRESOLUTION=1
Image Structure Metadata:
     INTERLEAVE=BAND
Corner Coordinates:
Upper Left ( -68.9126655, -32.8785732) ( 68d54’45.60”W, 32d52’42.86”S)
```

```
Lower Left ( -68.9126655, -36.1998544) ( 68d54'45.60"W, 36d11'59.48"S)
Upper Right ( -63.6417491, -32.8785732) ( 63d38'30.30"W, 32d52'42.86"S)
Lower Right ( -63.6417491, -36.1998544) ( 63d38'30.30"W, 36d11'59.48"S)
Center ( -66.2772073, -34.5392138) ( 66d16'37.95"W, 34d32'21.17"S)
Band 1 Block=611x8 Type=Float32, ColorInterp=Gray
```

Figura 5.2: Salida del programa gdalinfo.

En la información de salida que muestra el programa gdalinfo se puede verificar que los datos de proyección y tamaño del píxel son iguales a la capa original. En cuanto a las coordenadas existe una ligera diferencia, esto es debido a que las coordenadas suministradas en el WebGIS no necesariamente coinciden con la esquina del píxel, tomándose entonces la esquina más cercana. Lo anterior es lo deseable en este tipo de procesos, ya que se mantienen las propiedades del ráster original, evitándose un remuestreo que conlleve a modificar los valores originales de la capa utilizada.

Para verificar que no hay cambio en los valores de los píxeles se tomaron las coordenadas de la salida y se realizó un recorte de la imagen original utilizando la herramienta *clipper* del *software* QGIS que utiliza la función gdal_translate [1] para este propósito. Con los dos recortes, se procedió a realizar una dispersión de puntos y así verificar así el ajuste de los valores para las dos imágenes (Figura 5.3).

[1] http://www.gdal.org/gdal_translate.html

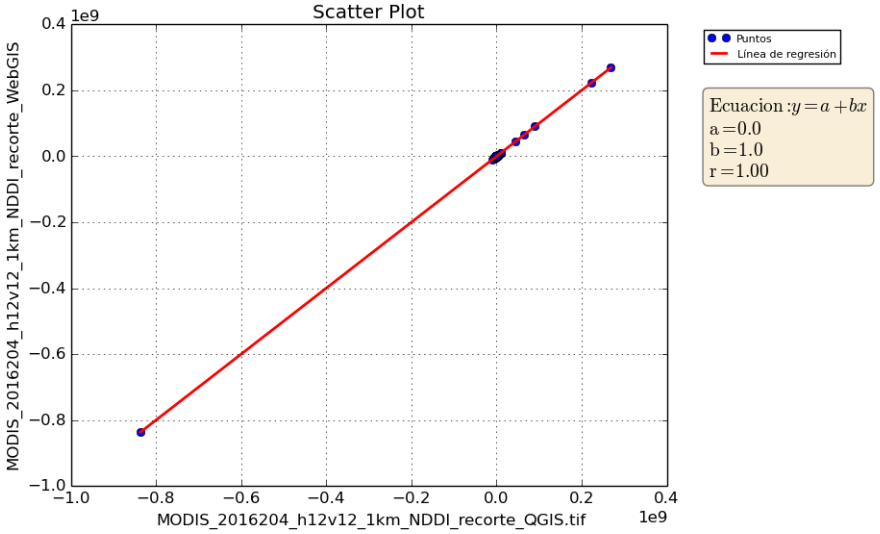


Figura 5.3: Dispersión realizada para verificar los valores de cada píxel en el recorte realizado con el WebGIS.

Los resultados muestran un coeficiente de correlación lineal (r)[2] de 1.0, así como una pendiente de la recta (b) igual a 1.0 y un intercepto (a) en 0.0. Reemplazando estos valores en la ecuación de la recta $y = a + bx$ se obtiene $y = x$, lo que indica que cada valor en la imagen original se corresponde con el recorte realizado desde el WebGIS.

5.2. Regresión Lineal

Para la prueba se construyó una línea de consulta que ejecuta el *script* "linear_regression.py", al que se le proporcionan los parámetros de índices, capas a utilizar, coordenadas del área de interés y, al igual que en el aplicativo cliente, un número aleatorio (Figura 5.4). Se seleccionaron los índices NDVI y NDWI, los cuales son útiles para verificar dinámicas de cambios en las coberturas de suelos (eg., Ahmed & Akter, 2017).

[2]Medida de relación lineal entre dos variables aleatorias cuantitativas

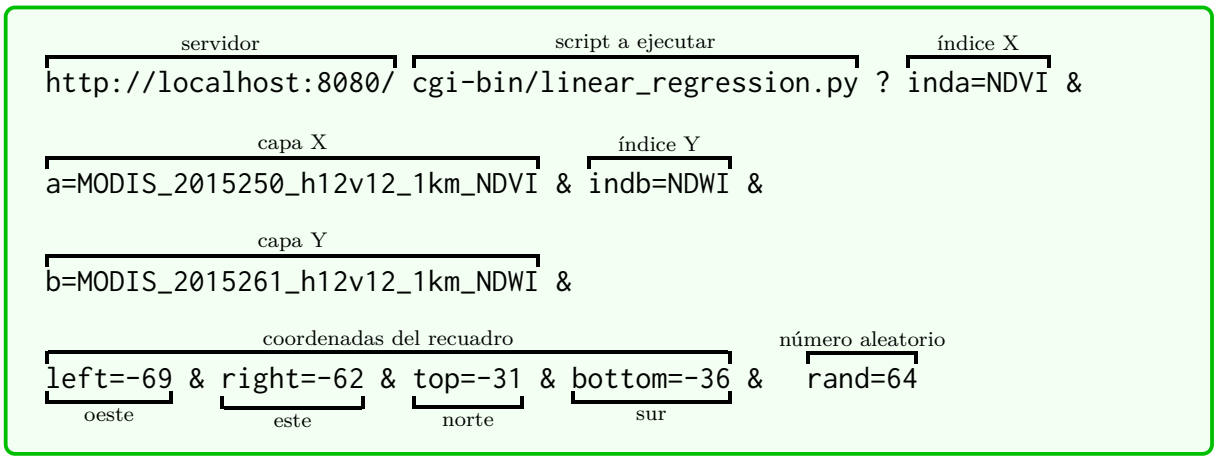


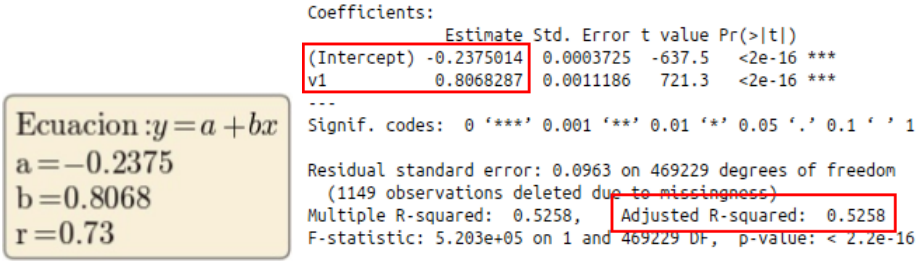
Figura 5.4: Línea para ejecutar la consulta de regresión lineal.

El *script* recibe la información y hace uso del módulo stats de la librería *scipy* para el cálculo de los parámetros de la regresión, posteriormente utiliza la librería *matplotlib* para realizar el gráfico de salida y finalmente retorna el nombre del archivo generado a la aplicación cliente.

Para verificar el resultado obtenido se hizo uso de las funciones *r.regression.line* y *lm* implementadas en el *software* GRASS GIS y en el lenguaje R respectivamente. Los *scripts* construidos siguen la misma metodología que el desarrollado para el WebGIS, con la diferencia que algunas variables son definidas directamente.

Al revisar los resultados de pendiente, intercepto y coeficiente de correlación lineal (r) obtenidos para las regresiones realizadas con cada una de las herramientas (Figura 5.5), se observaron resultados similares en los tres casos (teniendo en cuenta que el lenguaje R informa el coeficiente general de correlación (R^2)[3]), mientras que el complemento desarrollado y GRASS GIS informan el r, corroborando el buen funcionamiento del complemento desarrollado.

[3]Coeficiente de correlación lineal elevado al cuadrado



(a) Parámetros WebGIS. (b) Parámetros R.

```
y = a + b*x
   a (Offset): -0.237607
   b (Gain): 0.807128
   r (sumXY - sumX*sumY/N): 0.725551
   N (Number of elements): 469225
```

(c) Parámetros GRASS GIS.

Figura 5.5: Parámetros de la regresión lineal obtenidos con cada herramienta.

Cada proceso se dividió en las siguientes etapas:

- **Capturar parámetros de entrada:** proceso de ingestión de valores de entrada por parte del script, como imagen, espacio de trabajo donde se encuentra, coordenadas del área de interés y número aleatorio generado para el usuario.
- **Cargar las imágenes:** proceso de ingestión de datos por parte del *software*/lenguaje utilizado.
- **Extraer vectores:** proceso donde se selecciona el área de interés y se generan los vectores con los datos que se encuentran dentro de dicha área.
- **Calcular regresión:** en esta parte se calculan los valores de interés para el usuario, tal como los parámetros de ajuste de la recta y el r.
- **Generar resultados:** esta parte del proceso es la encargada de generar los gráficos en formato png para la descarga de los usuarios, así como un archivo en formato txt con los vectores generados. De esta manera, el usuario tendrá a disposición los valores con que fue calculada la regresión y podrá realizar cálculos posteriores.

Para cada proceso se realizó la medición de tiempos, lo cual, a modo exploratorio brinda una idea del rendimiento del complemento creado. Cada *script* se corrió 10 veces bajo las

mismas condiciones y se tomó como valor de referencia el promedio de los tiempos de cada etapa como se detalla en la tabla 5.3:

Proceso	WebGIS	GRASS GIS	R
Capturar parámetros de entrada	0.00017s	0.00002s	0.0365s
Cargar las imágenes	0.00048s	0.98916s	0.0248s
Extraer vectores	1.68833s	1.25393s	0.6659s
Calcular regresión	0.01833s	0.10240s	0.6222s
Generar resultados	5.23399s	6.34425s	16.0480s
Tiempo total	6.94130s	8.68976s	17.3974s

Tabla 5.3: Tiempos de procesamiento en segundos, para la regresión lineal, mediante los diferentes lenguajes y *software* utilizados

Al revisar los tiempos obtenidos, el WebGIS y GRASS GIS presentan una amplia diferencia con respecto al procesamiento desarrollado en R, marcada al momento de generar el archivo de dispersión. Esto puede estar dado por la cantidad de puntos al momento de graficar la dispersión.

Finalmente se evidencia un buen comportamiento en el tiempo que tarda la funcionalidad implementada para generar los resultados en relación a las demás herramientas utilizadas.

5.3. Perfil Temporal

El complemento desarrollado utiliza la función OpenLayers.Request.GET(), que realiza una petición de tipo HTTP GET, entregando al servidor los parámetros necesarios en la ejecución del archivo “temporal_profile.py” (nombre del índice seleccionado y las coordenadas del píxel), además de un número aleatorio entre 1 y 100 cuyo objetivo es minimizar la posibilidad de tener dos consultas simultáneas por parte de los usuarios del

sistema (Figura 5.6). Para las pruebas del servidor se construyó una consulta que genera la serie temporal de LST (LSTDay) como se detalla a continuación:

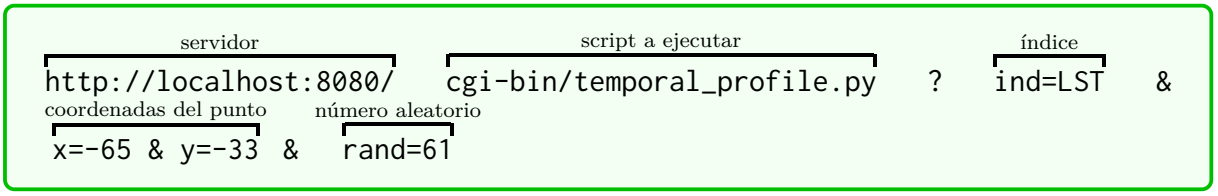
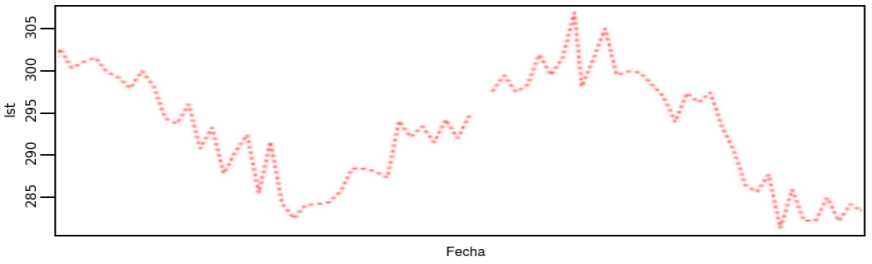
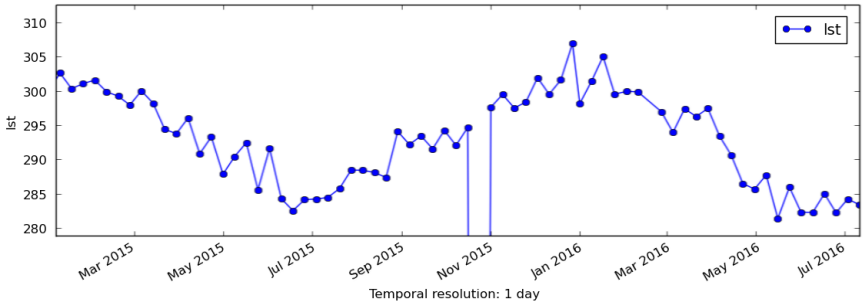


Figura 5.6: Línea de consulta utilizada para la prueba realizada al perfil temporal.

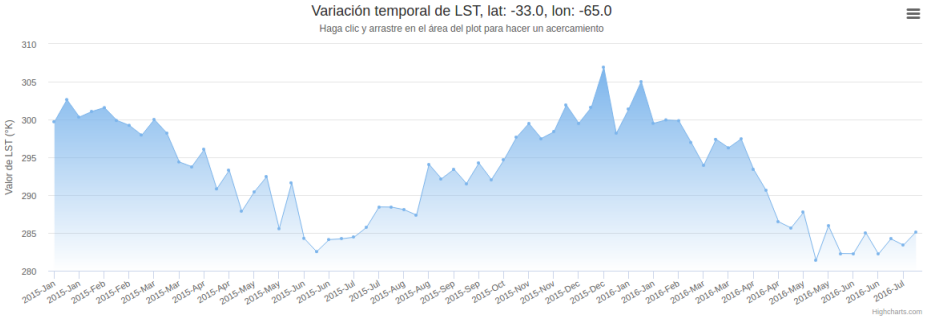
Para verificar el resultado de este proceso se construyeron dos rutinas, la primera hace uso de la función *extract* implementada en R para extraer los valores del píxel, mientras que la segunda utiliza herramientas para la manipulación de series temporales en GRASS GIS, obteniendo en cada caso el gráfico de la serie temporal de LST entre enero de 2015 y julio de 2016 (latitud: -33.0°, longitud: -65.0°)(Figura 5.7). Al observar los gráficos se puede inferir que los valores de los vectores son iguales, así como la predominancia del ciclo anual que, como se dijo anteriormente, viene condicionado por la estación del año.



(a) Perfil temporal R.



(b) Perfil temporal GRASS GIS.



(c) Perfil temporal WebGIS.

Figura 5.7: Perfiles generados durante las pruebas realizadas.

Existe una discontinuidad en los gráficos generados con R y GRASS GIS alrededor de octubre de 2015, esto se debe a que en la generación del gráfico para estos procesos no fueron eliminados los valores nulos[4], aunque si se descartaron durante el cálculo de los estadísticos evitando así una posible interferencia. Sin embargo, en el caso de la funcionalidad, se descartan los valores nulos y se realiza una interpolación lineal del dato faltante en el gráfico, sin incluir este valor en el cálculo de los estadísticos. A continuación se muestran los resultados de los estadísticos obtenidos con cada uno de los procesos (Figura 5.8):

[4]Valores que se encuentran por fuera del rango válido del conjunto de datos de prueba seleccionado

(a) Estadísticos WebGIS.

(b) Estadísticos R.

(c) Estadísticos GRASS GIS.

Figura 5.8: Estadísticos calculados a partir de las series generadas para el perfil temporal.

Al comparar los resultados, se evidencia una ligera diferencia entre los valores obtenidos con GRASS GIS y el aplicativo cliente. Se realizó entonces una revisión al procedimiento, donde se pudo corroborar los vectores eran similares, pero al momento de calcular los estadísticos, desde GRASS GIS se exportó la serie a un archivo de texto para después continuar con el proceso. Esto hizo que se aproximaran los datos con menor número de cifras decimales, explicando esa diferencia observada, lo que no ocurre con R y el complemento desarrollado.

Al igual que para la regresión lineal, cada proceso se dividió en diferentes etapas:

- **Capturar parámetros de entrada:** proceso de ingestión de valores de entrada por parte del script, como imagen, espacio de trabajo donde se encuentra, coordenadas del área de interés y número aleatorio generado para el usuario.

- **Cargar las imágenes y extraer vectores:** proceso de ingestión de datos por parte del *software*/lenguaje utilizado, selección del píxel de interés y extracción de los vectores con los valores a procesar.

- **Generar resultados:** esta parte del proceso es la encargada de generar el archivo html que contiene el perfil temporal, el archivo txt con el vector de la serie con sus respectivas fechas y los valores de los estadísticos calculados.

la rutina se corrió 10 veces bajo las mismas condiciones y se tomaron medidas de tiempos de ejecución para las diferentes etapas del procesamiento. Los valores fueron promediados y se detallan en la tabla 5.4:

Proceso	WebGIS	GRASS GIS	R
Capturar parámetros de entrada	0.000115s	0.000003s	0.011400s
Cargar las imágenes y extraer vectores	0.036471s	23.527577s	16.442300s
Generar resultados	0.000589s	36.735524s	0.009700s
Tiempo total	0.037175s	60.263104s	16.463400s

Tabla 5.4: Tiempos de procesamiento en segundos, para el perfil temporal, mediante los diferentes lenguajes y *software* utilizados

En este caso, se evidencia un mayor consumo de tiempo a partir de la importación de las imágenes y generación de los vectores que intervienen en el perfil cuando se utilizan R y GRASS GIS. Además, en el caso de GRASS GIS no se produjo directamente el gráfico, este se generó de manera manual para tener un punto de comparación visual con las demás herramientas, lo que hace que el tiempo no esté contemplado.

5.4. Calculadora

Su propósito es efectuar operaciones con una, dos o tres capas contenidas en el servidor. Al igual que los complementos de regresión y perfil temporal, se lleva a cabo sobre una región de interés, la cual es definida por el usuario dibujando un recuadro sobre el mapa.

La ejecución del complemento genera tres ventanas adicionales que capturan los parámetros de entrada: la primera permite elegir el número de capas a utilizar, la segunda habilita un

menú para la selección de los índices, así como los productos y la tercera contiene un cuadro de entrada de texto para introducir la ecuación. De esta manera, se construye la consulta que mediante una petición de tipo HTTP GET se comunica con el servidor proporcionando la totalidad de la información necesaria para el funcionamiento de uno de los *scripts* que intervienen en el proceso (rastercalculator_1band.py, rastercalculator_2bands.py o rastercalculator_3bands.py), así como un número aleatorio para disminuir la posibilidad que se crucen dos consultas simultáneas.

A diferencia de los complementos anteriores, el *script* que ejecuta el proceso del lado del servidor no utiliza solamente Python, sino que llama la función r.mapcalc del *software* GRASS GIS para el cálculo de la ecuación entre las capas, ampliando la gama de posibilidades de la calculadora a las operaciones que esta herramienta tiene establecidas.

La prueba se realizó solamente para el caso de 3 bandas dado que el funcionamiento para 1 y 2 es básicamente el mismo. Se construyó entonces una consulta que ejecuta el *script* "rastercalculator_3bands.py" y le entrega los parámetros que se muestran en la Figura 5.9

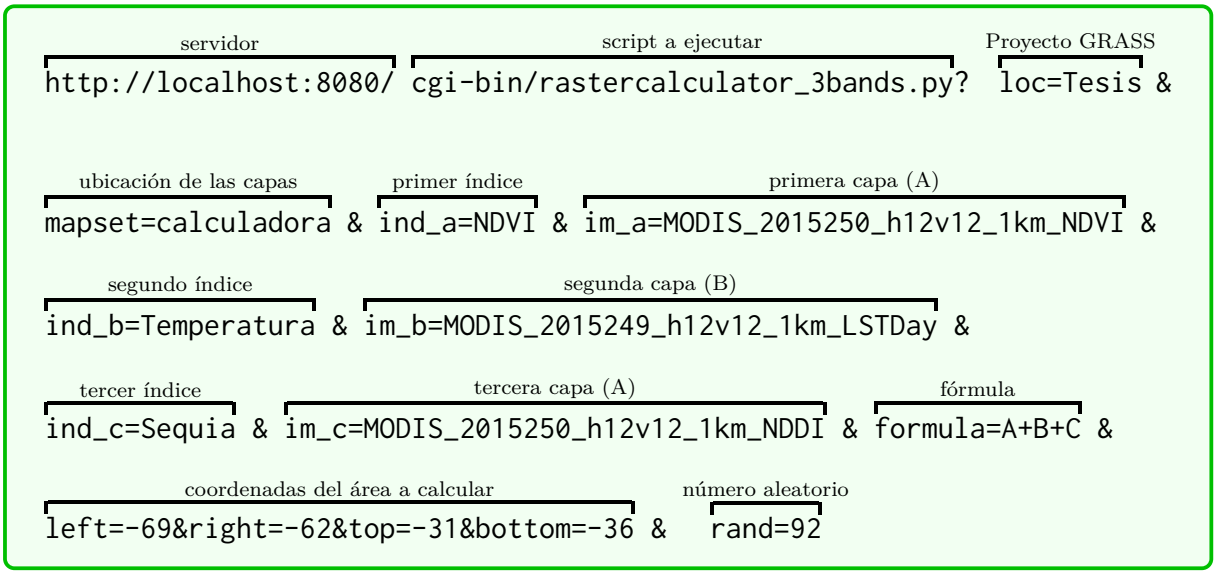


Figura 5.9: Línea de consulta utilizada para la prueba realizada a la calculadora ráster.

Para evaluar los tiempos de ejecución y los resultados, se desarrollaron *scripts* para R y QGIS que utilizan la misma metodología. Las etapas en que se dividieron los procesos fueron las siguientes:

- **Capturar parámetros de entrada:** proceso de ingestión de valores de entrada por parte del script, como imagen, espacio de trabajo donde se encuentra, coordenadas del área de interés y número aleatorio generado para el usuario.

- **Cargar las imágenes:** proceso de ingestión de datos por parte del *software*/lenguaje utilizado.

- **Definir región y procesar las imágenes:** esta parte del proceso es la encargada de procesar las imágenes para el área definida por el usuario.

Los tiempos de ejecución de los diferentes procesos se detallan en la tabla 5.5.

Proceso	WebGIS	R	QGIS
Capturar parámetros de entrada	0.000338s	0.018500s	0.000043s
Cargar las imágenes	1.217024s	0.043400s	0.082815s
Definir región y procesar las imágenes	2.306758s	1.720600s	0.588268s
Tiempo total	3.524121s	1.782500s	0.671125s

Tabla 5.5: Tiempos de procesamiento en segundos, obtenidos al ejecutar la calculadora con tres capas ráster, mediante los diferentes lenguajes y *software* utilizados

En este caso los tiempos de ejecución son mejores en QGIS y R que en el complemento desarrollado para la aplicación cliente y revisando el *script*, se encontró que GRASS GIS utiliza más tiempo en la importación de capas que los demás complementos. Además cuando se establecen los valores nulos, éste lo hace sobre la capa completa mientras que QGIS y R lo hacen solamente sobre el área recortada. Sin embargo QGIS modifica el tamaño del píxel teniendo en cuenta el área del recorte, lo que no es conveniente para el usuario final dado que, si bien esa diferencia no se traduce significativamente en los valores de las capas, tener diferentes tamaños de píxel puede hacer que las capas no sean compatibles en procesos posteriores.

Teniendo esto en cuenta y con el fin de aprovechar las diferentes operaciones que proporciona la función del *software* GRASS GIS, se tomó la decisión de continuar con el uso de esta herramienta como *software* de procesamiento en *background* del sistema. Además se considera que al ser 3 el mayor número de capas que se van a importar en la calculadora, el tiempo no aumentaría tanto como por ejemplo en el caso del perfil temporal, donde se trabaja con el total de la serie y se aumentaría considerablemente el tiempo con respecto a otras herramientas.

Se intentó hacer entonces un recorte previo en GRASS GIS generando salidas que permitieran luego continuar con el proceso, sin embargo tardó más en las pruebas realizadas y se decidió no modificar la estructura que ya se tenía. Las capas de salida fueron importadas en un *software* de escritorio para su revisión, verificando la información correspondiente al sistema de referencia y tamaño de píxel verificando que correspondieran a lo esperado.

Capítulo 6

Conclusiones y Trabajos Futuros

6.1. Conclusiones

Se logró la configuración de un aplicativo WebGIS con una estructura adecuada para el almacenamiento de datos ambientales geoespaciales de tipo vectorial y ráster para Suramérica. Este cuenta con funcionalidades desarrolladas a partir de *software* de uso libre orientadas a proporcionar la capacidad de efectuar consultas, procesamientos y descargas de la información contenida en el servidor. Dicha información es compatible entre sí, es decir, cuenta con iguales características (formato, proyección cartográfica y resolución espacial) que eliminan tiempos de búsqueda y preprocesamiento.

Las funcionalidades implementadas permiten al usuario explorar las capas contenidas en la base de datos directamente desde el aplicativo cliente, sin implicar necesariamente su descarga. Asimismo, se desarrollaron complementos para la descarga de los productos del servidor de mapas que, en los casos de las capas ráster, pueden proporcionarlas en su extensión completa o realizar un recorte de un área especificada por el usuario. El sistema cuenta además con complementos para graficar series temporales o regresiones lineales, brindando adicionalmente estadísticos calculados a partir de los valores extraídos en cada consulta. Finalmente proporciona una calculadora ráster que hace uso de las funcionalidades de la herramienta "r.mapcalc" del *software* GRASS GIS para efectuar operaciones sobre un área específica con hasta 3 capas contenidas en el servidor de mapas, permitiendo al usuario final la descarga del resultado de la operación ingresada.

Si bien existen herramientas similares en otras plataformas como Google Earth Engine para el procesamiento de capas, el aplicativo desarrollado presenta ventajas al usuario ya que este no necesariamente debe conocer de programación para llevar a cabo operaciones básicas entre las capas disponibles. Además no se presenta restricción en cuanto a área procesada, de esta manera, el usuario puede procesar su región de interés en su totalidad y descargar su resultado sin necesidad de realizar mosaicos posteriores. Por otra parte, los datos se descargan directamente a su disco duro, así no existen requerimientos de espacio en la nube para almacenamiento de la información.

En el caso de las series temporales, también existen herramientas para generar gráficos y descarga de información, sin embargo, algunas de estas herramientas no cuentan con información actualizada, por ejemplo, el visor de series del INPE cuenta con información disponible hasta julio de 2017, lo que no garantiza que se cuente con la misma cantidad de información que se pondrá a disposición en el aplicativo desarrollado.

Las pruebas realizadas a los complementos que se construyeron ayudaron a verificar que los productos alcanzados con cada uno son acordes a lo esperado. Los valores que arrojan algunos *software* SIG de escritorio como GRASS GIS y QGIS, además de lenguajes estadísticos como R, en ninguno de los casos difiere de lo que se obtuvo desde el lado del cliente. Lo anterior supone un procedimiento adecuado al momento de escalar los valores o eliminar los que se encuentran por fuera del rango, brindando seguridad al usuario al hacer uso de del aplicativo.

Al evaluar los tiempos de respuesta con cada complemento se puede concluir que los scripts de procesamiento se encuentran optimizados de acuerdo a los requerimientos del sistema, sólamente al ejecutar la calculadora con 3 capas el resultado tardó más que con QGIS y R (cerca de 1.7 seg más que R y 2.9 seg más que QGIS). Sin embargo, dado que la diferencia no es grande y evaluando los beneficios adquiridos al utilizar las funcionalidades de la herramienta “r.mapcalc”, se optó por continuar con esta herramienta desde el lado del servidor para el procesamiento de la información.

6.2. Trabajos Futuros

A continuación se presentan posibles líneas de trabajo para un futuro mejoramiento del WebGIS:

- Si bien, el sistema ofrece más de una forma de establecer la IP del servidor, sería adecuado utilizar librerías internas que no dependen de una página web externa para su consulta, dado que si esta no se encuentra disponible, puede generar una caída en el sistema temporal.

- Incorporar otro tipo de índices como complemento para aquellas zonas donde los índices actuales pueden presentar deficiencias.

- Continuar con el desarrollo de funcionalidades que contribuyan al desarrollo de estudios ambientales, incorporando herramientas provistas por *software* como GRASS GIS, el cual ya se encuentra configurado en el lado del servidor.

- Trabajar en un diseño responsivo que se adapte a pantallas de dispositivos móviles.

- Explorar el estándar WPS como opción para el desarrollo de nuevas funcionalidades de procesamiento de información.

- Generar rutinas para la automatización de la actualización de la base de datos contenida en el aplicativo cliente.

ANEXOS

Anexo A

Carga de Información

Script para cargar información en el servidor de mapas

```
# -*- coding: utf-8 -*-

# Importar librerías
import glob
import os
import sys

# Usuario y contraseña del servidor de mapas
gsUser = 'admin'
gsPasswd ='geoserver'
#Nombre del espacio de trabajo y el estilo que se quiere aplicar
workspace = 'NDWI'
estilo='NDWI'

#Ruta donde se encuentran los datos que se van a cargar en el servidor de mapas
path_datos = "/media/andres/Andres/Datos_Tesis/"+workspace+"/"

#bloque para crear un workspace
try :
  os.system("curl -v -u "+gsUser+":"+gsPasswd+" -XPOST -H 'Content-type: application/xml' \
  -d '<workspace><name>"+workspace+"</name></workspace>' \
  http://127.0.0.1:8080/geoserver/rest/workspaces")
except:
  sys.exit(1)

#Listado de capas a cargar en el servidor
lista_imagenes=glob.glob(path_datos+"/*.tif")

```

```
29 for img in lista_imagenes:
30   coverage = img.split("/")[-1] #extrae el nombre de la capa
31
32   #bloque para cargar el tiff
33   try:
34     os.system("curl -u "+gsUser+":"+gsPasswd+" -XPUT -H 'Content-type:image/tiff' \
35     --data-binary '@"+img+"' http://127.0.0.1:8080/geoserver/rest/workspaces/"+\
36     workspace+"/coveragestores/"+str(coverage[:-4])+"/file.geotiff")
37   except:
38     print "No se pudo cargar el tif"
39     sys.exit(2)
40
41   #bloque para aplicar el estilo
42   try:
43     os.system("curl -u "+gsUser+":"+gsPasswd+" -XPUT -H 'Content-type: text/xml' -d \
44     '<layer><defaultStyle><name>"+estilo+"</name></defaultStyle><enabled>true</enabled>\
45     </layer>'http://127.0.0.1:8080/geoserver/rest/layers/"+workspace+":"+str(coverage))
46   except:
47     print "No se pudo aplicar el estilo"
48     sys.exit(2)
```

Anexo B

Definir ip del servidor

Script para establecer de forma manual o automática, la ip del servidor

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import json
import cgi
import cgitb
cgitb.enable()
print "Content-type: application/json\n\n"
data = cgi.FieldStorage()

#opción 1: Consultar y establecer la ip pública de la máquina
import urllib2
import re

texto = urllib2.urlopen('http://www.cualesmiip.com/').read()
ip = re.search(r'\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b',texto).group(0)
print json.dumps(str(ip)+":8080")

#opción 2: Consultar y establecer la ip local de la máquina
import socket

ip = socket.gethostbyname(socket.gethostname())
print json.dumps(str(ip)+":8080")

#opción 3: Establecer una IP fija
ip = "127.0.1.1"
print json.dumps(str(ip)+":8080")
```

Anexo C

Complemento Grid.js

Script para visualizar la grilla de coordenadas en el aplicativo cliente

```
1  //Dependencias requeridas
2
3  /**
4   * @require plugins/Tool.js
5   * @require OpenLayers/Control/Graticule.js
6   */
7
8  Ext.ns("myapp.plugins");
9
10 var miip = OpenLayers.Request.GET({
11   url: '/cgi-bin/consulta_ip.py',
12   async: false,
13   success: function(reply) {
14     var b = JSON.parse(reply.responseText);
15     console.log(b.toString())
16   },
17   failure: function(reply){
18     console.log(reply)
19   }
20 });
21
22 var host = JSON.parse(miip.responseText).toString();
23
24 myapp.plugins.Grilla = Ext.extend(gxp.plugins.Tool, {
25
26   ptype: "myapp_grilla",
27
28   addActions: function() {
```

```
29      var map = this.target.mapPanel.map;
30      var action = new GeoExt.Action({
31        tooltip: "Grilla",
32        iconCls: "mi_icono",
33        icon: "http://"+host+"/iconos/grilla.png",
34        scale: 'medium',
35        enableToggle: true,
36        control: new OpenLayers.Control.Graticule({
37          autoActivate: false,
38          numPoints: 2,
39          labelled: true,
40          displayInLayerSwitcher: false,
41          lineSymbolizer: {
42            strokeColor: "#848484", //Cambia color
43            strokeWidth: 1,  //Cambia ancho
44            strokeOpacity: 1,  //Cambia opacidad
45            strokeDashstyle: "dash"  //Cambia tipo de linea
46          },
47          labelSymbolizer: {
48            fontSize: 14, //Tamaño de letra
49            fontColor: "#848484" //Color de letra
50            //~ fontFamily: "Verdana" //Tipo de letra
51          }
52        }),
53        map: map
54      });
55
56      return myapp.plugins.Grilla.superclass.addActions.apply(this,[action]);
57    }
58  });
59
60  Ext.preg(myapp.plugins.Grilla.prototype.ptype, myapp.plugins.Grilla);
```

Anexo D

Complemento DownloadLayers.js

Script para realizar la descarga de capas ráster contenidas en el servidor

```
1  //Dependencias requeridas
2
3  /**
4   * @require plugins/Tool.js
5   * @require OpenLayers/Layer/Vector.js
6   * @require GeoExt/widgets/Action.js
7   * @require OpenLayers/Format.js
8   */
9
10 Ext.ns("myapp.plugins");
11 var miip = OpenLayers.Request.GET({
12   url: '/cgi-bin/consulta_ip.py',
13   async: false,
14   success: function(reply) {
15     var b = JSON.parse(reply.responseText);
16     console.log(b.toString())
17   },
18   failure: function(reply){
19     console.log(reply)
20   }
21 });
22
23 var host = JSON.parse(miip.responseText).toString();
24 var url_server = "http://"+host+"/geoserver/wcs?";
25 var service = "WCS";
26 var version = "2.0.1";
27 var request = "GetCoverage";
28 var namespace = "local";
```

```
29 var format = "geotiff";
30 var crs = "EPSG:4326";
31
32 myapp.plugins.Descarga = Ext.extend(gxp.plugins.Tool, {
33   ptype: "myapp_descarga",
34   addActions: function() {
35     action = new Ext.Button({
36           iconCls: "mi_icono",
37           tooltip: "Descarga",
38           icon: "http://"+host+"/iconos/descarga.png",
39           scale: 'small',
40                 //text: "Descarga",
41                 handler: function() {
42                     this.displayPopup()
43                 },
44                 scope: this
45             })
46
47     return myapp.plugins.Descarga.superclass.addActions.apply(this,[action]);
48   },
49
50   displayPopup: function() { //Mostrar ventana
51     //url sin workspace para listar todas las capas
52     var url = '/geoserver/ows?request=GetCapabilities&service=WMS';
53     var xml = new OpenLayers.Format.XML();     //cargo la funcion para leer el xml
54     var req = OpenLayers.Request.GET({      //cargo la funcion para leer el xml
55       url: url,
56       success: function(reply) {
57         var array = [];   //si es éxito se declara el array y se llena
58         var doc = xml.read(reply.responseText);
59         var nomb = doc.documentElement.getElementsByTagName('Layer');
60         var keywords = doc.documentElement.getElementsByTagName('KeywordList');
61         for (i = 1; i < keywords.length; i++) {
62           //Filtro los que no son WCS y lleno el array
63           if(keywords[i].firstElementChild.innerHTML == "WCS"){
64             array.push([nomb[i].firstElementChild.innerHTML,keywords[i].
65             firstElementChild.innerHTML]);
66           }
67         };
68         //crear el store para almacenar el array anterior
69         var store = new Ext.data.ArrayStore({
70           autoDestroy: true,
71           storeId: 'store',
72           fields: [
73           {name: 'imagen'},
```

```
74            {name: 'wcs'}
75            ]
76          });
77          store.loadData(array);
78
79          var sm = new Ext.grid.CheckboxSelectionModel({ //Columna checkbox
80            header: '',
81            singleSelect: false,
82            checkOnly: true
83          });
84
85          var grid = new Ext.grid.GridPanel({ //creo el grid
86            id: 'gridactive',
87            title: '<span style="color:#04B404; text-align:left;">Descargar capas del
88            servidor</span>',
89            renderTo: Ext.getBody(),
90            autowidth: true,
91            height: 300,
92            store: store,
93            sm: sm,
94            multiSelect: true,
95            columns: [
96            sm,
97            {header: "Imagen",dataIndex: "imagen", sortable: true,width:300},
98            {header: "Keywords", dataIndex: "wcs"}
99            ]
100
101         });
102
103         win= new Ext.Window({   //creo la ventana que contiene el grid
104           layout: 'fit',
105           width: 600, height: 300,
106           title: '',
107           items: [grid],
108           closable: true, //quita la cruz para cerrar
109           html:' <div id="config"><span id="showGrid"></span></div>',
110           modal: false, //inhabilita el background
111           frame: true, //quita el marco
112           draggable: true, //desplazable
113           buttons: [{     //creo el boton ok que crea los links de descarga
114             text: 'OK',
115             handler : function(){
116               console.log(sm.getSelections());
117               var arr = []; //Crear Array
118               //Llenar Array con nombre de la imagen y url de descarga
```

```
119                 Ext.each(sm.getSelections(), function(rec){
120                   arr.push([rec.data.imagen,
121                   url_server + "service=" + service + "&version=" + version +
122                   "&request=" + request + "&namespace=" + namespace + "&coverageId=" +
123                   rec.data.imagen + "&format=" + format + "&crs=" + crs ]); //raw data
124                 });
125                 win.close();//cerrar primera ventana
126                 //crear el store para almacenar el array anterior
127                 var store2 = new Ext.data.ArrayStore({
128                   autoDestroy: true,
129                   storeId: 'store2',
130                   fields: [
131                   {name: 'imagen'},
132                   {name: 'descarga'}
133                   ]
134                 });
135                 store2.loadData(arr);
136                 //crear el panel con las columnas a cargar
137                 var grid2 = new Ext.grid.GridPanel({
138                   id: 'gridactive',
139                   title: '<span style="color:#04B404; text-align:left;">Descargar capas del
140                   servidor</span>',
141                   renderTo: Ext.getBody(),
142                   autowidth: true,
143                   height: 300,
144                   store: store2,
145                   multiSelect: true,
146                   columns: [
147                   {id:'imagen', header: "Imagen",dataIndex: 'imagen', sortable: true,
148                   width:300},
149                   {id:'descarga', header: "Descarga", dataIndex: 'descarga' ,width:300,
150                   renderer:function(myValue){ //, myDontKnow, myRecord
151                     return '<a href="'+myValue+'">Descarga</a>'
152                     }
153                   }
154                   ]
155                 });
156                 //crear la ventana con los datos seleccionados para descargar
157                 win2= new Ext.Window({
158                   layout: 'fit',
159                   width: 600,
160                   height: 300,
161                   title: '',
162                   items: [grid2],
163                   closable: true, //quita la cruz para cerrar
```

```
164                 html:' <div id="config"><span id="showGrid"></span></div>',
165                 modal: false, //inhabilita el background
166                 frame: true, //quita el marco
167                 draggable: true, //desplazable
168                 buttons: [{
169                   text: 'CLOSE',
170                   handler : function(){
171                     win2.close()
172                   }
173                 }]
174               });
175               win2.show()
176             }
177           }]
178         });
179         win.show()
180 
181       },
182       failure: function(reply){
183         console.log("Error")
184       }
185     });
186 
187   }
188 
189 });
190 Ext.preg(myapp.plugins.Descarga.prototype.ptype, myapp.plugins.Descarga);
```

Anexo E

Script temporal_profile.py

Script en python para crear el perfil temporal

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-

#Librerías a utilizar
import json
import cgi
import cgitb
cgitb.enable()
import gdal
import os
import glob
import datetime
import export_html
import numpy as np

print "Content-type: application/json\n\n"

#Capturar parámetros de entrada
data = cgi.FieldStorage()
prod = data.getvalue("prod")
x = data.getvalue("x")
y = data.getvalue("y")
rand = data.getvalue("rand")

#Define parámetros de procesamiento según índice
if prod == "NDVI":
   path = "/media/andres/Andres/Tesis/geoserver/data/NDVI/"
   producto = prod
```

```
29    escala = 0.0001
30    min_x,max_x = -3000,10000
31    un = ""
32
33  elif prod == "LST":
34    path = "/media/andres/Andres/Tesis/geoserver/data/Temperatura/"
35    producto = "LSTDay"
36    escala = 0.02
37    min_x,max_x = 7500,65535
38    un = "(K)"
39
40  elif prod == "NDDI":
41    path = "/media/andres/Andres/Tesis/geoserver/data/Sequia/"
42    producto = prod
43    escala = 0.0001
44    min_x,max_x = -32000,32000
45    un = ""
46
47  elif prod == "Precipitación":
48    path = "/media/andres/Andres/Tesis/geoserver/data/Precipitacion/"
49    producto = "Precipitacion"
50    escala = 1
51    min_x,max_x = 0,1000
52    un = "(mm)"
53
54  elif prod == "NDWI":
55    path = "/media/andres/Andres/Tesis/geoserver/data/NDWI/"
56    producto = prod
57    escala = 0.0001
58    min_x,max_x = -3000,10000
59    un = ""
60  try:
61    #Remueve archivos anteriores si existen
62    os.system("rm /var/lib/tomcat7/webapps/ROOT/temp/Perfil*.html")
63    lista = glob.glob(path+"*%s"%(str(producto)))
64    lista.sort()
65    lista2 = []
66
67    for l in lista: #Accede a cada directorio y lista capas disponibles
68      lista2.append(glob.glob(l+"/*.geotiff")[0])
69    lista2.sort()
70
71    valor = [] #Vector para almacenar valores
72    i=0
73    tiempo = "[" #Variable para almacenar datos de tiempo
```

```
74
75    for l2 in lista2:
76      i+=1
77      #Por cada imagen carga matriz de datos y calcula píxel de interés
78      ds = gdal.Open(l2)
79      geoT=ds.GetGeoTransform()
80      ulx = geoT[0]
81      uly = geoT[3]
82      resx = geoT[1]
83      resy = geoT[5]
84      c_ulx = ulx + resx/2.
85      c_uly = uly + resy/2.
86      dist_x = abs(int(round((c_ulx-float(x))/resx,0)))
87      dist_y = abs(int(round((c_uly-float(y))/resy,0)))
88      #Extrae valor del píxel de interés
89      m = ds.GetRasterBand(1).ReadAsArray(dist_x,dist_y,1,1)
90      #Extrae fecha a partir del nombre de la capa
91      date = (l2.split("/")[-1]).split("_")[1]
92      yy = int(date[0:4])
93      dd = int(date[4:])
94      date = datetime.datetime(yy,1,1) + datetime.timedelta(dd)
95      datestr=str(date)[0:10].split("-")
96      #Escala valores y elimina datos nulos, crea vectores con la información
97      if m[0][0]>min_x and m[0][0]<max_x:
98        fechasTXT.append(date.strftime("%Y/%m/%d"))
99        valor.append(float(m[0][0])*escala)
100       tiempo = tiempo+"Date.UTC("+datestr[0]+","+str(int(datestr[1])-1)+","+ \
101       str(int(datestr[2]))+"),"
102   tiempo = tiempo[0:len(tiempo)-1]+"]"
103   #Define ruta y nombre de salida
104   path_sal="/var/lib/tomcat7/webapps/ROOT/temp/"
105   salida_str = "Perfil_"+prod+"_"+rand+".html"
106   headerTXT = "Fecha  Valor: "+producto
107   matriz = np.concatenate(([fechasTXT],[valor]),axis=0)
108   #Archivo de salida en formato txt
109   np.savetxt(path_sal+salida_str[:-4]+"txt",matriz.T,fmt="%s",delimiter=" ", \
110   header=headerTXT)
111   #Función para generar el archivo html con el gráfico correspondiente
112   export_html.html(prod,round(float(x),2),round(float(y),2),tiempo,valor,path_sal+ \
113   salida_str,un)
114   #Envía respuesta al aplicativo cliente
115   print json.dumps(salida_str)
116 except:
117   #Envía respuesta de error en caso que el proceso anterior no se pueda completar
118   print json.dumps("error")
```

Anexo F

Script export_html.py

Script para generar archivo de salida html con perfil temporal

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-
import numpy as np

def html(prod,lon,lat,x,y,salida,unidad):
  #Estadisticos de la serie
  media = np.mean(y)
  sigma = np.std(y,ddof=1)
  maximo = np.max(y)
  minimo = np.min(y)

  salida=open(salida,'w')
  texto = "<!DOCTYPE html PUBLIC '-//W3C//DTD XHTML 1.0 Strict//EN'\n\
  'http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd'>\n\
  <html xmlns='http://www.w3.org/1999/xhtml' xml:lang='en' lang='en'>\n\
  \n\
  <head>\n\
    <title>Perfil Temporal</title>\n\
    <!-- Complete CSS -Html -->\n\
    \n\
    <link href='https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css' \
    rel='stylesheet' \
    integrity='sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u'
    crossorigin='anonymous'>\n\
    <!-- jQuery Core -->\n\
    <script src='https://code.jquery.com/jquery-3.1.1.min.js' \
    integrity='sha256-hVVnYaiADRTO2PzUGmuLJr8BLUSjGIZsDYGmIJLv2b8='\
    crossorigin='anonymous'></script>\n\
```

```
29      <!-- Complete JavaScript -Html -->\n\
30      <script src='https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js' \
31      integrity='sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa' \
32      crossorigin='anonymous'></script>\n\
33      \n\
34      <!-- Highcharts -->\n\
35      <script src='https://code.highcharts.com/stock/highstock.js'></script>\n\
36      <script src='https://code.highcharts.com/stock/modules/exporting.js'></script>\n\
37      \n\
38      <meta http-equiv='content-type' content='text/html;charset=utf-8' />\n\
39      <meta name='generator' content='Geany 1.23.1' />\n\
40    </head>\n\
41    \n\
42    <body>\n\
43      <div class='container'> \n\
44      <div class='row'>\n\
45    <!--\n\
46      <div class='col-sm-6 col-md-6'>\n\
47    -->\n\
48        <div id='container1'></div>\n\
49      </div>\n\
50      </div>\n\
51      </div>\n\
52      \n\
53        <script type='text/javascript'>\n\
54          \n\
55          var chart = new Highcharts.Chart({\n\
56            chart: {\n\
57              zoomType: 'x',\n\
58              renderTo: 'container1'\n\
59            },\n\
60            title: {\n\
61              text: 'Variaci\u00F3n temporal de "+str(prod)+", lat: "+str(lat)+", \
62              lon: "+str(lon)+"'\n\
63            },\n\
64            subtitle: {\n\
65              text: document.ontouchstart === undefined ?\n\
66              'Haga clic y arrastre en el área del plot para hacer un acercamiento' : \
67              'Pinch the chart to zoom in'\n\
68            },\n\
69            xAxis: {\n\
70              type: 'datetime',\n\
71              categories: "+str(x)+",\n\
72              labels: {\n\
73                format: '{value:%Y-%b}',\n\
```

```
74              align: 'right',\n\
75              rotation: -30\n\
76            }\n\
77          },\n\
78          yAxis: {\n\
79            title: {\n\
80              text: 'Valor de "+str(prod)+" "+str(unidad)+"'\n\
81            }\n\
82          },\n\
83          legend: {\n\
84            enabled: false\n\
85          },\n\
86          plotOptions: {\n\
87            area: {\n\
88              fillColor: {\n\
89                linearGradient: {x1:0, y1:0, x2:0, y2:1},\n\
90                stops: [\n\
91                [0, Highcharts.getOptions().colors[0]],\n\
92                [1, Highcharts.Color(Highcharts.getOptions().colors[0]).setOpacity(0).\
93                get('rgba')]\n\
94                ]\n\
95              },\n\
96              marker: {\n\
97                radius: 2\n\
98              },\n\
99              lineWidth: 1,\n\
100             states: {\n\
101               hover: {\n\
102                 lineWidth: 1\n\
103               }\n\
104             },\n\
105             threshold: null\n\
106           }\n\
107         },\n\
108         series: [{\n\
109           type: 'area',\n\
110           name: 'Valor de "+str(prod)+" "+str(unidad)+"',\n\
111           data: "+str(y)+",\n\
112           \n\
113         }]\n\
114       });\n\
115     </script>\n\
116     <table width=600px align='center'>\n\
117       <caption><font size='4'><b>Información Complementaria.</b></font></caption>\n\
118       <tr> \n\
```

```
119         <th bgcolor='#DDE0FC'><font size='2'><center>Estadístico</center></font></th> \n\
120         <th bgcolor='#DDE0FC'><font size='2'><center>Valor</center></font></th>\n\
121         </tr>\n\
122         <tr> <td align='center'><font size='2'>Máximo</font></td> <td align='center'>\
123         <font size='2'>"+str(round(maximo,1))+"</font></td>\n\
124         </tr>\n\
125         <tr> <td align='center'><font size='2'>Mínimo</font></td> <td align='center'>\
126         <font size='2'>"+str(round(minimo,1))+"</font></td>\n\
127         </tr>\n\
128         <tr> <td align='center'><font size='2'>Media (x&#772;)</font></td> \
129         <td align='center'><font size='2'>"+str(round(media,1))+"</font></td>\n\
130         </tr>\n\
131         <tr> <td align='center'><font size='2'>Desviación Estándar (&#x3C3;)</font>\
132         </td> <td align='center'><font size='2'>"+str(round(sigma,1))+"</font></td>\n\
133         </tr>\n\
134       </table>\n\
135   </body>\n\
136   </html>"
137   salida.write(texto)
138   salida.close()
```

Anexo G

Script linear_regression.py

Script en python para realizar la regresión lineal

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-

#Importar librerías
import gdal
import os
from scipy import stats
import regression_plot as pl
import numpy as np

import json
import cgi
import cgitb
cgitb.enable()

print "Content-type: application/json\n\n"

#Capturar parámetros de entrada
data = cgi.FieldStorage()
store_x = data.getvalue("inda")
im_x = data.getvalue("a")
store_y = data.getvalue("indb")
im_y = data.getvalue("b")
#Datos para la región de interes
n = data.getvalue("top")
s = data.getvalue("bottom")
e = data.getvalue("right")
w = data.getvalue("left")
```

```
29 rand = data.getvalue("rand")
30 #Ruta de salida y de datos de entrada
31 path_sal="/var/lib/tomcat7/webapps/ROOT/temp/"
32 path = "/ruta/base/datos/geoserver/data/"
33 #Ruta imágenes de entrada
34 x = path+store_x+"/"+im_x
35 y = path+store_y+"/"+im_y
36 #Índice de la imagen
37 indice_x = im_x.split("_")[4]
38 indice_y = im_y.split("_")[4]
39 #Escalas según el índice
40 escala_NDVI = 0.0001
41 escala_LSTDay = 0.02
42 escala_NDDI = 0.0001
43 escala_Precipitacion = 1
44 escala_NDWI = 0.0001
45 #Valores máximos y mínimos para cada índice
46 min_NDVI,max_NDVI = -3000,10000
47 min_LSTDay,max_LSTDay = 7500,65535
48 min_NDDI,max_NDDI = -32000,32000
49 min_Precipitacion,max_Precipitacion = 0,1000
50 min_NDWI,max_NDWI = -3000,10000
51 #Valores a utilizar según las escalas anteriores
52 escala_x = globals()["escala_"+str(indice_x)]
53 escala_y = globals()["escala_"+str(indice_y)]
54 min_x = globals()["min_"+str(indice_x)]
55 min_y = globals()["min_"+str(indice_y)]
56 max_x = globals()["max_"+str(indice_x)]
57 max_y = globals()["max_"+str(indice_y)]
58
59 #Función para calcular el número de pixel según coordenadas
60 def calc_pixel(imagen,lon,lat):
61   ds = gdal.Open(imagen)
62   geoT=ds.GetGeoTransform()
63   ulx,uly = geoT[0],geoT[3]
64   resx,resy = geoT[1],geoT[5]
65   dist_x = abs(int(round((ulx-float(lon))/resx,0)))
66   dist_y = abs(int(round((uly-float(lat))/resy,0)))
67   return (dist_x,dist_y)
68
69 #Función para generar la matriz de coordenadas que se incluirá en el archivo txt de salida
70 def coordenadas(low_x,w_box,celdas_lon,resX,up_y,celdas_lat,n_box,resY):
71   dist_x1 = abs(int(round((w_box-low_x)/resX,0)))
72   cx1 = low_x+dist_x1*resX
73   vectorX = (np.arange(cx1,cx1+celdas_lon*resX,resX)+resX/2.).astype(str)
```

```
74    dist_y1 = abs(int(round((up_y-n_box)/resY,0)))
75    cy1 = up_y+dist_y1*resY
76    vectorY = (np.arange(cy1,cy1+celdas_lat*resY,resY)+resY/2.).astype(str)
77    mat_coords = []
78    for i in range(len(vectorY)):
79      for j in range(len(vectorX)):
80        mat_coords.append(str(vectorY[i])+" "+str(vectorX[j]))
81    mat_coords = np.array(mat_coords)
82    return (mat_coords)
83
84
85  try:
86    #Cargar imagen
87    dsx = gdal.Open(x+"/%s.geotiff"%(im_x))
88    dsy = gdal.Open(y+"/%s.geotiff"%(im_y))
89    #Calcular pixel de inicio y tamaño de la matriz
90    ulx_lon, ulx_lat = calc_pixel(x+"/%s.geotiff"%(im_x),w,n)
91    uly_lon, uly_lat = calc_pixel(y+"/%s.geotiff"%(im_y),w,n)
92    lrx_lon, lrx_lat = calc_pixel(x+"/%s.geotiff"%(im_x),e,s)
93    dif_lon = abs(ulx_lon - lrx_lon)
94    dif_lat = abs(ulx_lat - lrx_lat)
95    #Generar las coordenadas lat/lon para cada punto de la matriz
96    geoT = dsx.GetGeoTransform()
97    matriz_coords = coordenadas(float(geoT[0]),float(w),dif_lon,float(geoT[1]),\
98    float(geoT[3]),dif_lat,float(n),float(geoT[5]))
99
100   #Matrices de cada imagen para la región de interés
101   matriz_x = dsx.GetRasterBand(1).ReadAsArray(ulx_lon,ulx_lat,dif_lon,dif_lat)*\
102   float(escala_x)
103   matriz_y = dsy.GetRasterBand(1).ReadAsArray(uly_lon,uly_lat,dif_lon,dif_lat)*\
104   float(escala_y)
105   #Convierte matrices en vectores
106   vec_x = np.reshape(matriz_x,(1,dif_lon*dif_lat))[0][:]
107   vec_y = np.reshape(matriz_y,(1,dif_lon*dif_lat))[0][:]
108
109   #Hallar valores nulos
110   rx1 = np.where(vec_x<min_x*escala_x)
111   rx2 = np.where(vec_x>max_x*escala_x)
112   nanx = np.where(vec_x.astype(str)==str('nan'))
113   ry1 = np.where(vec_y<min_y*escala_y)
114   ry2 = np.where(vec_y>max_y*escala_y)
115   nany = np.where(vec_y.astype(str)==str('nan'))
116   exclude = np.append(rx1,rx2)
117   exclude = np.append(exclude,nanx)
118   exclude = np.append(exclude,ry1)
```

```
119    exclude = np.append(exclude,ry2)
120    exclude = np.append(exclude,nany)
121    exclude = np.unique(exclude)
122    exclude = np.flipud(exclude)
123
124    #Vector para almacenar los datos válidos para la regresión
125    vect_plot = []
126    vec_x=np.delete(vec_x,exclude)#Elimina valores nulos en x
127    vec_y=np.delete(vec_y,exclude)#Elimina valores nulos en y
128    coords = np.delete(matriz_coords,exclude)
129
130    for val in range(len(vec_x)): #Vector para procesamiento
131      vect_plot.append([vec_x[val],vec_y[val]])
132    #Calcula parámetros de la regresión
133    par_a, par_b, r_value, p_value, std_err = stats.linregress(vec_x,vec_y)
134    #Remueve regresiones anteriores si las hay
135    os.system("rm "+path_sal+"Dispersion_*.jpg")
136    os.system("rm "+path_sal+"Dispersion_*.txt")
137    #Plotear la dispersión y generar txt de salida
138    salida_str = "Dispersion_"+str(rand)+".jpg"
139    pl.plotear(vec_x,vec_y,par_b,par_a,r_value,im_x,im_y,rand,path_sal+salida_str)
140    txt = np.concatenate(([coords],[vec_x],[vec_y]),axis=0)
141    headerTXT = "latitud  longitud  %s  %s" %(im_x,im_y)
142    np.savetxt(path_sal+salida_str[:-3]+"txt",txt.T,fmt="%s",delimiter="  ",header=headerTXT)
143    #Enviar información de salida al aplicativo cliente
144    print json.dumps(salida_str)
145 except:
146    print json.dumps("error") #Mensaje de error en caso que el proceso no se lleve a cabo
```

Anexo H

Script regression_plot.py

Script para generar el gráfico de regresión lineal

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-
#Importar librerías
import numpy as np
import matplotlib
matplotlib.use('Agg')
import pylab as plt

def plotear(x,y,a,b,r,imx,imy,rand,sal):
  ecuacion = "y=%s+%s*x"%(str(a),str(b))
  py1=a+(b*np.min(x))
  py2=a+(b*np.max(x))
  #Cuadro de texto con información complementaria
  textstr = '$\mathrm{Ecuacion}:y=a+bx$\n$\mathrm{a}= %s$\n$\mathrm{b}= %s$\n$\mathrm{R} \
  =%.2f$'%(round(a,4),round(b,4),r)
  fig = plt.figure(1)
  ax = fig.add_subplot(111)
  props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)
  # place a text box in upper left in axes coords
  texto = ax.text(1.07, 0.85, textstr, transform=ax.transAxes, fontsize=14,\
  verticalalignment='top', bbox=props)
  ux=[np.min(x),np.max(x)]
  uy=[py1,py2]
  ax.plot(x,y,ls='*',marker='o',label='Puntos')
  ax.plot(ux,uy,color=("red"),lw=2,label='Línea de regresión'.decode('utf-8'))
  plt.title('Scatter Plot')
  plt.xlabel(imx)
  plt.ylabel(imy)
```

```
29  handles, labels = ax.get_legend_handles_labels()
30  lgd = ax.legend(handles, labels,loc=1,prop={'size':8},bbox_to_anchor=(1.3, 1))
31  plt.grid()
32  #Guardar figura en png
33  plt.savefig(sal, bbox_extra_artists=(texto,), bbox_inches='tight',pad_inches=0.3)
```

Anexo I

Script conectar_grass.py

Script para ejecutar GRASS GIS mediante scripts de python

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#Importar librerías
import os
import sys

def conectar(location,mapset,gisdb):
  # Ruta al script de ejecución de GRASS GIS
  grass7bin = '/usr/bin/grass72'

  # Establecer la variable de entorno GISDBASE, donde se encuentra el proyecto
  os.environ['GISDBASE'] = gisdb

  # Consulta a GRASS 72 por su GISBASE
  gisbase = "/usr/lib/grass72"
  os.environ['GISBASE'] = gisbase

  # Definir entorno GRASS-Python
  gpydir = "/usr/lib/grass72/etc/python"
  sys.path.append(gpydir)

  # importar enlaces GRASS-Python
  import grass.script.setup as gsetup

  # Inicia sesión
  gsetup.init(gisbase, gisdb, location, mapset)
```

Anexo J

Script rastercalculator_3bands.py

Script para ejecutar el complemento calculadora con tres capas raster

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-
# Librerías utilizadas
import conectar_grass
import datetime
import os
import json
import cgi
import cgitb
cgitb.enable()

# Parámetros de entrada
print "Content-type: application/json\n\n"
data = cgi.FieldStorage()
location = "Tesis"
mapset =  "calculadora"
gisdb = "/media/andres/Andres/TesisGrass"
a = data.getvalue("im_a")
workspace_a = data.getvalue("ind_a")
b = data.getvalue("im_b")
workspace_b = data.getvalue("ind_b")
c = data.getvalue("im_c")
workspace_c = data.getvalue("ind_c")
formula = data.getvalue("formula")
rand = data.getvalue("rand")

#Índice de cada imagen
indice_a = str(a).split("_")[4]
```

```
29 indice_b = str(b).split("_")[4]
30 indice_c = str(c).split("_")[4]
31
32 #Valores máximos y mínimos para cada índice
33 min_NDVI,max_NDVI = -3000,10000
34 min_LSTDay,max_LSTDay = 7500,65535
35 min_NDDI,max_NDDI = -32000,32000
36 min_Precipitacion,max_Precipitacion = 0,1000
37 min_NDWI,max_NDWI = -3000,10000
38
39 min_a = globals()["min_"+str(indice_a)]
40 min_b = globals()["min_"+str(indice_b)]
41 min_c = globals()["min_"+str(indice_c)]
42 max_a = globals()["max_"+str(indice_a)]
43 max_b = globals()["max_"+str(indice_b)]
44 max_c = globals()["max_"+str(indice_c)]
45
46 #Datos para definir la region
47 n = data.getvalue("top")
48 s = data.getvalue("bottom")
49 e = data.getvalue("right")
50 w = data.getvalue("left")
51 #Resolución de las imágenes
52 res = 0.008626704385179
53 #Fecha en la que se ejecuta el proceso
54 fecha = datetime.datetime.now().strftime("%Y%j")
55 #Ruta de cada imagen
56 path_a="/media/andres/Andres/Tesis/geoserver/data/"+workspace_a+"/"+str(a)+"/"
57 path_b="/media/andres/Andres/Tesis/geoserver/data/"+workspace_b+"/"+str(b)+"/"
58 path_c="/media/andres/Andres/Tesis/geoserver/data/"+workspace_c+"/"+str(c)+"/"
59 path_temp = "/var/lib/tomcat7/webapps/ROOT/temp/" #Ruta de salida temporal
60
61 #Escalas según el índice
62 escala_NDVI = 0.0001
63 escala_LSTDay = 0.02
64 escala_NDDI = 0.0001
65 escala_Precipitacion = 1
66 escala_NDWI = 0.0001
67 escala_a = globals()["escala_"+str(indice_a)]
68 escala_b = globals()["escala_"+str(indice_b)]
69 escala_b = globals()["escala_"+str(indice_c)]
70
71 #Reescribir fórmula
72
73 formula2=""
```

```
74 for i in range(len(formula)):
75   if formula[i]=='A':
76     formula2=formula2+a+"*"+str(escala_a)
77   elif formula[i]=='B':
78     formula2=formula2+b+"*"+str(escala_b)
79   elif formula[i]=='C':
80     formula2=formula2+c+"*"+str(escala_c)
81   else:
82     formula2=formula2+formula[i]
83 
84 #Conectar a la base de datos
85 conectar_grass.conectar(location,mapset,gisdb)
86 import grass.script as gscript
87 
88 try: #Eliminar capa anterior si existe
89   gscript.run_command("g.remove", type="rast", pattern="*", flags='f')
90 except:
91   pass
92 
93 try:
94   #Función para eliminar valores nulos
95   def setnull(img,minimo,maximo):
96     salidaTXT = "/var/lib/tomcat7/webapps/ROOT/temp/"+img+".txt"
97     gscript.run_command('r.univar', map=img, output=salidaTXT, overwrite=True)
98     vmin_img = float(open(salidaTXT).readlines()[6].split(" ")[1][:-1])
99     vmax_img = float(open(salidaTXT).readlines()[7].split(" ")[1][:-1])
100     if vmin_img<minimo:
101       nval=str(vmin_img-1)+"-"+str(minimo)
102       gscript.run_command('r.null', map=img, setnull=nval)
103     else:
104       pass
105     if vmax_img>maximo:
106       nval=str(maximo)+"-"+str(vmax_img+1)
107       gscript.run_command('r.null', map=img, setnull=nval)
108     else:
109       pass
110     os.system("rm "+salidaTXT)
111 
112   #Cargar capas
113   gscript.run_command("r.in.gdal", input=path_a+str(a)+".geotiff", output = str(a))
114   gscript.run_command("r.in.gdal", input=path_b+str(b)+".geotiff", output = str(b))
115   gscript.run_command("r.in.gdal", input=path_c+str(c)+".geotiff", output = str(c))
116   #Definir región de interés
117   gscript.run_command('g.region', n=float(n), s=float(s), e=float(e), w=float(w), \
118   ewres=res, nsres=res,align = str(b))
```

```
119   #Establecer valores nulos
120   setnull(str(a),min_a,max_a) #eliminar valores nulos
121   setnull(str(b),min_b,max_b) #eliminar valores nulos
122   setnull(str(c),min_c,max_c) #eliminar valores nulos
123   #Procesar datos
124   salida = "resultado_%s_%s_1km_temp"%(str(fecha),str(rand))
125   gscript.mapcalc(salida + " = " + formula2, overwrite = True, quiet = True)
126   #Salida
127   output = path_temp+salida+".tif"
128   gscript.run_command("r.out.gdal", input=salida+"@"+mapset, format="GTiff", \
129   output=output,overwrite = True)
130   #Borrar datos
131   gscript.run_command("g.remove", type="rast", name=str(a), flags='f')
132   gscript.run_command("g.remove", type="rast", name=str(b), flags='f')
133   gscript.run_command("g.remove", type="rast", name=str(c), flags='f')
134   gscript.run_command("g.remove", type="rast", name=salida+"@"+mapset, flags='f')
135   print json.dumps(salida+".tif")
136 except:
137   print json.dumps("error")
```

Bibliografía

Aguirre Patiño, M., España, R. E., Solís Granda, I. & Aranda Segovia, A. (2011). Diseño y simulación de un data center cloud computing que cumpla con la norma pci–dss. URL http://www.dspace.espol.edu.ec/xmlui/handle/123456789/15924.

Ahmed, K. R. & Akter, S. (2017). Analysis of landcover change in southwest bengal delta due to floods by ndvi, ndwi and k-means cluster with landsat multi-spectral surface reflectance satellite data. *Remote Sensing Applications: Society and Environment* **8**, 168–181.

Bechtel, B. (2012). Robustness of annual cycle parameters to characterize the urban thermal landscapes. *IEEE Geoscience and Remote Sensing Letters* **9**(5), 876–880.

Beerling, D. J. (1993). The impact of temperature on the northern distribution limits of the introduced species fallopia japonica and impatiens glandulifera in north-west europe. *Journal of Biogeography* , 45–53.

Biswas, T. (2010). *A Spatio-Temporal Analysis of Landscape Change within the Eastern Terai, India: Linking Grassland and Forest Loss to Change in River Course and Land Use.* Tesis de Doctorado, Utah State University. URL https://digitalcommons.usu.edu/etd/976/.

Bocher, E. & Neteler, M. (2010). *Geospatial free and open source software in the 21st century.* Springer.

Cabra Forero, M. D. R., Cardona Gómez, D. & Palacios Palacios, A. L. (2015). Los valores ecológicos como recurso pedagógico en la conservación y cuidado del medio ambiente a los

estudiantes del grado tercero de primaria de la institución educativa general santander. URL http://repository.libertadores.edu.co/handle/11371/274.

Cantos, J. O. & Ferrer, J. M. G. (2015). Riesgo de inundaciones en tierras alicantinas. método y resultados. *NIMBUS nº 09-10* (9-10), 99–123.

Carmona, A. & Monsalve, J. J. (2004). Sistemas de información geográficos. En: *Congreso de Ingeniería de Sistemas en la Universidad San Buenaventura de Medellín Colombia*. URL https://dds.cepal.org/infancia/guia-para-estimar-la-pobreza-infantil/bibliografia/capitulo-IV/Carmona%20Alvaro%20y%20Monsalve%20Jhon%20%281999%29%20Sistemas%20de%20informacion%20geografica.pdf.

Ceccarelli, S., Balsalobre, A., Susevich, M. L., Echeverria, M. G., Gorla, D. E. & Marti, G. A. (2015). Modelling the potential geographic distribution of triatomines infected by triatoma virus in the southern cone of south america. *Parasites & vectors* **8**(1), 153.

de La Beaujardiere, J. (2006). Opengis® web map server implementation specification. *Open Geospatial Consortium Inc., OGC* , 06–042.URL http://portal.opengeospatial.org/files/?artifact_id=14416. Accedido el 26–11–2016.

de Souza, R. d. C. M., Campolina Silva, G. H., Bezerra, C. M., Diotaiuti, L. & Gorla, D. E. (2015). Does triatoma brasiliensis occupy the same environmental niche space as triatoma melanica? *Parasites & vectors* **8**(1), 1.

Erena, M., García, P., Pérez, P., Sánchez, D., Paya, D., Atenza, J., Rosa, J., Baños, I., Ortega, C., Pérez, F., Collado Jiménez, A. M. & Aroca, A. (2009). El geoportal de la sequía y los recursos hídricos de la región de murcia. En: *XXVII Congreso Nacional de Riegos, Murcia*.

Fernandez Gómez, M. J., Vicente Villardón, J. L. & Zarza, C. A. (2001). Análisis de las relaciones entre especies y medioambiente: estudio comparativo de diferentes técnicas. En: *Conferencia Internacional de Estadística en Estudios Medioambientales*.

Fischer, A. G. (1960). Latitudinal variations in organic diversity. *Evolution* **14**(1), 64–81.

Flanagan, D. (2007). *JavaScript. La Guía Definitiva*. O'Reilly Media, Inc.

Fonts Bartolomé, Ó. & Pericay, M. (2012). Adaptación de opengeo suite para la gestión integral de información geográfica en el ayuntamiento de castellbisbal. En: *VI Jornadas de SIG Libre*. Universitat de Girona. Servei de Sistemes d'Informació Geográfica i Teledetecció.

Gao, B.-C. (1996). Ndwi—a normalized difference water index for remote sensing of vegetation liquid water from space. *Remote sensing of environment* **58**(3), 257–266.

Giuliani, G., Ray, N. & Lehmann, A. (2011). Grid-enabled spatial data infrastructure for environmental sciences: Challenges and opportunities. *Future Generation Computer Systems* **27**(3), 292–303.

Gorla, D. E. (2002). Variables ambientales registradas por sensores remotos como indicadores de la distribución geográfica de triatoma infestans (heteroptera: Reduviidae). *Ecología austral* **12**(2), 117–127.

Gu, Y., Brown, J. F., Verdin, J. P. & Wardlow, B. (2007). A five-year analysis of modis ndvi and ndwi for grassland drought assessment over the central great plains of the united states. *Geophysical Research Letters* **34**(6).

Gulácsi, A. & Kovács, F. (2015). Drought monitoring with spectral indices calculated from modis satellite images in hungary. *Journal of Environmental Geography* **8**(3-4), 11–20.

Gutiérrez Lozano, J., Vargas Tristán, V., Romero Rodríguez, M., de la Cruz, P., Manuel, J., Aguirre Bortoni, M. d. J. & Silva Espinoza, H. T. (2011). Periodos de retorno de lluvias torrenciales para el estado de tamaulipas, méxico. *Investigaciones geográficas* (76), 20–33.

Hazzard, E. (2011). *Openlayers 2.10 beginner's guide*. Packt Publishing.

Ihaka, R. (1998). R: Past and future history. *Computing Science and Statistics* **392396**.

Julien, Y. & Sobrino, J. A. (2009). The yearly land cover dynamics (ylcd) method: An analysis of global vegetation from ndvi and lst parameters. *Remote Sensing of Environment* **113**(2), 329–334.

Kawabata, A., Ichii, K. & Yamaguchi, Y. (2001). Global monitoring of interannual changes in vegetation activities using ndvi and its relationships to temperature and precipitation. *International Journal of Remote Sensing* **22**(7), 1377–1382.

Korte, G. (2001). *The GIS book*. Cengage Learning.

Lane, S. J., Alonso, J. C. & Martín, C. A. (2001). Habitat preferences of great bustard otistarda flocks in the arable steppes of central spain: are potentially suitable areas unoccupied? *Journal of applied ecology* **38**(1), 193–203.

Li, X., Di, L., Han, W., Zhao, P. & Dadi, U. (2010). Sharing geoscience algorithms in a web service–oriented environment (grass gis example). *Computers & Geosciences* **36**(8), 1060–1068.

Lott, R. (2015). Geographic information–well–known text representation of coordinate reference systems. URL http://docs.opengeospatial.org/is/12-063r5/12-063r5.html.

Madonsela, S., Cho, M. A., Ramoelo, A., Mutanga, O. & Naidoo, L. (2018). Estimating tree species diversity in the savannah using ndvi and woody canopy cover. *International Journal of Applied Earth Observation and Geoinformation* **66**, 106–115.

McCain, C. M. & Colwell, R. K. (2011). Assessing the threat to montane biodiversity from discordant shifts in temperature and precipitation in a changing climate. *Ecology letters* **14**(12), 1236–1245.

Miller, J. (2010). Species distribution modeling. *Geography Compass* **4**(6), 490–509.

Neteler, M. & Mitasova, H. (2008). *Open Source GIS: A GRASS GIS Approach*. Springer, New York.

Obsomer, V., Defourny, P. & Coosemans, M. (2012). Predicted distribution of major malaria vectors belonging to the anopheles dirus complex in asia: ecological niche and environmental influences. *PLoS One* **7**(11), e50475.

Olaya, V. (2011). Sistemas de información geográfica. URL http://volaya.github.io/libro-sig/.

Olivera-Guerra, L., Mattar, C., Merlin, O., Durán-Alarcón, C., Santamaría-Artigas, A. & Fuster, R. (2017). An operational method for the disaggregation of land surface temperature to estimate actual evapotranspiration in the arid region of chile. *ISPRS Journal of Photogrammetry and Remote Sensing* **128**, 170–181.

Orfeo, O. & Ruberto, A. (2000). Zonas de riesgo para obras contra inundaciones (Río Bermejo, Chaco). URL https://www.researchgate.net/publication/266491756_Zonas_de_riesgo_para_obras_contra_inundaciones_Rio_Bermejo_Chaco.

Osborne, P. E., Alonso, J. & Bryant, R. (2001). Modelling landscape-scale habitat use using gis and remote sensing: a case study with great bustards. *Journal of applied ecology* **38**(2), 458–471.

Peng, G.-X., Jing, L., Chen, Y.-H. & Norizan, a.-P. (2007). A forest fire risk assessment using aster images in peninsular malaysia. *Journal of China University of Mining and Technology* **17**(2), 232–237.

Pettorelli, N., Vik, J. O., Mysterud, A., Gaillard, J.-M., Tucker, C. J. & Stenseth, N. C. (2005). Using the satellite-derived ndvi to assess ecological responses to environmental change. *Trends in ecology & evolution* **20**(9), 503–510.

R Core Team (2000). R language definition. *Vienna, Austria: R foundation for statistical computing*. URL http://web.mit.edu/~r/current/arch/amd64_linux26/lib/R/doc/manual/R-lang.pdf.

Ravelo, A. C. & Pascale, A. (1997). Identificación de ocurrencia de sequías mediantes imágenes del satélite noaa e información terrestre. *Rev. Facultad de Agronomía* **17**(1), 105–107.

Romeu Carrasco, A., Rey Pérez, A. d. & Montesinos Lajara, M. (2012). Implantación de geoportales con soporte técnico profesionalizado en software libre. En: *VI Jornadas de SIG Libre*. Universitat de Girona. Servei de Sistemes d'Informació Geográfica i Teledetecci'o.

Rouse Jr, J. W., Haas, R. H., Schell, J. & Deering, D. (1973). Monitoring the vernal advancement and retrogradation (green wave effect) of natural vegetation. URL https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19730016613.pdf.

Skidmore, A. (2003). *Environmental modelling with GIS and remote sensing*. CRC Press.

Takeuchi, W. & Gonzalez, L. (2009). Blending modis and amsr-e to predict daily land surface water coverage. En: *Proceedings of the International Remote Sensing Symposium (ISRS), Busan, Korea*, vol. 29.

van Rossum, G. & de Boer, J. (1991). Linking a stub generator (ail) to a prototyping language (python). En: *Proceedings of the Spring 1991 EurOpen Conference, Troms, Norway*.

Van Rossum, G. & Drake Jr, F. L. (1995). *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands.

Van Rossum, G. & Drake Jr, F. L. (2004). Tutorial python. URL http://python.org.

Wang, L., Ma, Y., Yan, J., Chang, V. & Zomaya, A. Y. (2018). pipscloud: High performance cloud computing for remote sensing big data management and processing. *Future Generation Computer Systems* **78**, 353–368.

Weber, D., Schaepman-Strub, G. & Ecker, K. (2018). Predicting habitat quality of protected dry grasslands using landsat ndvi phenology. *Ecological Indicators* **91**, 447–460.

Whiteside, A. & Evans, J. (2006). Web coverage service (wcs) implementation specification. *OGC*. URL http://portal.opengeospatial.org/files/?artifact_id=18153. Accedido el 26–11–2016.

Wisz, M. S., Tamstorf, M. P., Madsen, J. & Jespersen, M. (2008). Where might the western svalbard tundra be vulnerable to pink-footed goose (anser brachyrhynchus) population expansion? clues from species distribution models. *Diversity and Distributions* **14**(1), 26–37.

Woodward, F. I. (1987). Temperature and the distribution of plant species. En: *Symposia of the Society for Experimental Biology*, vol. 42.

Yuan, F. & Bauer, M. E. (2007). Comparison of impervious surface area and normalized difference vegetation index as indicators of surface urban heat island effects in landsat imagery. *Remote sensing of Environment* **106**(3), 375–386.

Zader, P. J. (2016). *Implementación de geoprocesos como servicios web*. Tesis de Maestría, Facultad de Matemática, Astronomía, Física y Computación–FAMAF. URL http://www.famaf.unc.edu.ar/wp-content/uploads/2017/08/43-Zader.pdf.

Websites Consultados

Azurmendi, L. (2012). Las graves consecuencias de la sequía. URL https://www.guioteca.com/medio-ambiente/las-graves-consecuencias-de-la-sequia/. Accedido el 11-12-2016.

Boundless Spatial Inc. (2016a). Opengeo suite for ubuntu linux. URL https://connect.boundlessgeo.com/docs/suite/4.8/intro/installation/ubuntu/index.html. Accedido el 16-05-2018.

Boundless Spatial Inc. (2016b). Ubuntu linux packages. URL http://suite.opengeo.org/docs/latest/intro/installation/ubuntu/packages.html. Accedido el 25-03-2017.

Butler, H., Daly, M., Doyle, A., Gillies, S., Schaub, T. & Schmidt, C. (2008). The geojson format specification. URL http://geojson.org/geojson-spec.html. Accedido el 4-12-2016.

Embrapa (2017). Produtos, processos e serviços. URL https://www.embrapa.br/monitoramento-por-satelite/busca-de-produtos-processos-e-servicos/-/produto-servico/2301/webgis---base-de-dados-virtual-dos-ecossistemas-costeiros-da-bacia-de-campos. Accedido el 26-01-2017.

ESRI (2012). Tres representaciones fundamentales de capas de información geográfica. URL http://resources.arcgis.com/es/help/getting-started/articles/026n0000000n000000.htm. Accedido el 4-12-2016.

Fonts, O. & González, F. (2013). Geoserver: Publicación de datos vectoriales. URL http://snmb-admin.readthedocs.io/en/latest/geotalleres/geoserver-vector/geoserver_vector.html. Accedido el 06-02-2017.

GeoExt (2016). Geoext - javascript toolkit for rich web mapping applications. URL http://www.geoext.org/. Accedido el 6-12-2016.

Geoserver (2014). What is geoserver? URL http://geoserver.org/about/. Accedido el 26-11-2016.

Geoserver (2016a). Gdal image formats. URL http://docs.geoserver.org/stable/en/user/data/raster/gdal.html. Accedido el 1-12-2016.

Geoserver (2016b). History. URL http://docs.geoserver.org/stable/en/user/introduction/history.html. Accedido el 30-11-2016.

Geoserver (2016c). Overview. URL http://docs.geoserver.org/stable/en/user/introduction/overview.html. Accedido el 30-11-2016.

Geoserver (2017). Overview. URL http://docs.geoserver.org/stable/en/user/rest/. Accedido el 3-02-2017.

ISAGRO (2017a). Isagro, información satelital para el agro. URL https://www.isagro.org.ar/seccion/1. Accedido el 25-01-2017.

ISAGRO (2017b). Programa regional de empleo de información satelital para la productividad agrícola. URL https://www.isagro.org.ar/seccion/2. Accedido el 25-01-2017.

Jones, E., Oliphant, T., Peterson, P. *et al.* (2001). SciPy: Open source scientific tools for Python. URL http://www.scipy.org/. Accedido el 21-06-2017.

MappingGIS (2012). ¿qué es opengeo suite? URL https://mappinggis.com/2012/05/que-es-opengeo-suite/. Accedido el 01-02-2017.

MapServer (2016). About. URL http://mapserver.org/about.html#about. Accedido el 30-11-2016.

NASA (2007a). Earth system science data resources: tapping into a wealth of data, information, and services. URL http://www.pseudology.org/webmaster/NASA_ESSDR112007.pdf. Accedido el 03-05-2017.

NASA (2007b). Earth system science data resources: tapping into a wealth of data, information, and services. URL http://daac.ornl.gov/ESSDR112007.pdf. Accedido el 13-01-2016.

OGC (2016a). About ogc. URL http://www.opengeospatial.org/ogc. Accedido el 26-11-2016.

OGC (2016b). Geography markup language. URL http://www.opengeospatial.org/standards/gml. Accedido el 03-12-2016.

OGC (2016c). Web feature service implementation specification with corrigendum. URL http://docs.opengeospatial.org/is/04-094r1/04-094r1.html. Accedido el 26-11-2016.

OGC (2016d). Web map service. URL http://www.opengeospatial.org/standards/wms. Accedido el 26-11-2016.

OGC (2017). Web processing service. URL http://www.opengeospatial.org/standards/wps. Accedido el 02-05-2017.

OpenLayers (2008). What is openlayers 2? URL http://docs.openlayers.org/. Accedido el 5-12-2016.

OSGeo (2016). About the open source geospatial foundation. URL http://www.osgeo.org/content/foundation/about.html. Accedido el 4-12-2016.

QGIS Development Team (2018). Guía de usuario de qgis. URL https://docs.qgis.org/2.18/es/docs/user_manual/index.html. Accedido el 16-05-2018.

R Development Core Team (2018). Opengeo suite for ubuntu linux. URL https://www.r-project.org/about.html. Accedido el 24-05-2018.

Sencha (2016). Sencha ext js. URL https://www.sencha.com/products/extjs/#overview. Accedido el 10-12-2016.

SUR Emprendimientos Tecnológicos (2017a). Características de suriwebgis. URL http://www.suremptec.com/es/productos/suri/suriwebgis/caracteristicas.html. Accedido el 26-01-2017.

SUR Emprendimientos Tecnológicos (2017b). Casos de estudio. URL http://www.suremptec.com/es/casos-de-estudio/65-webgis-calidad-de-aguai.html. Accedido el 26-01-2017.

Universidad de Valencia (2015). Javascript y java. URL http://www.uv.es/jac/guia/jscript/javascr01.htm. Accedido el 03-02-2017.