UNIVERSIDAD POLITÉCNICA DE VALENCIA

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

TRABAJO FIN DE MÁSTER
EN
SISTEMAS DE INFORMACIÓN

CREACIÓN DE UN PORTAL WEB ORIENTADO A LA INFORMACIÓN GEOGRÁFICA VOLUNTARIA

Realizado por:

Rafael Castellet Ginard

Supervisado por:

Dr. D. Óscar Pastor López
Dra. Dña. Eloína Coll Aliaga

Septiembre 2013

MÁSTER UNIVERSITARIO EN INGENIERÍA DEL SOFTWARE, MÉTODOS FORMALES Y SISTEMAS DE INFORMACIÓN

Contenidos

Lista de figuras

Lista de tablas

Agradecimientos

En primer lugar quiero agradecer a mis tutores, los doctores D. Óscar Pastor López y Dña. Eloína Coll Aliaga, la labor de dirección y supervisión realizada, sin la cual este Trabajo Fin de Máster (TFM) no se hubiera podido llevar a cabo.

En segundo lugar, quiero mostrar mi agradecimiento al doctor D. Raúl Zurita-Milla por su tiempo y apoyo durante la realización de este trabajo, y asimismo, por sus más que interesantes observaciones y sugerencias.

Asimismo, quiero expresar mi gratitud a todas aquellas personas que de muy diversas maneras me han apoyado y siempre han estado dispuestas a ayudarme.

Finalmente quiero agradecer muy especialmente a mi familia y amigos más allegados su apoyo incondicional y constante, así como la enorme paciencia mostrada durante todo este tiempo.

Gracias a todos

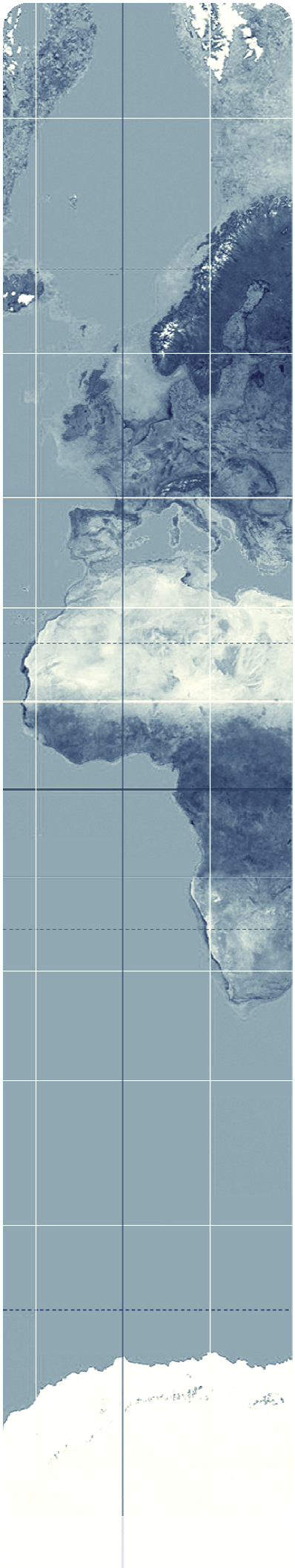
Lista de abreviaturas

AJAX	Asynchronous JavaScript And XML
API	Application Programming Interface
CSW	Catalog Service for the Web
DBMS	Database Management System
DCP	Distributed Computing Platform
ER	Engineering Reports
ESRI	Environmental Systems Research Institute
GeoSPARQL	Geographic SPARQL Protocol and RDF Query Language
GeoXACML	Geospatial eXtensible Access Control Markup Language
GIS	Geographic Information System
GML	Geography Markup Language
GMT	Greenwich Mean Time
GNU	General Public License
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Infraestructura de Datos Espaciales
IETF	Internet Engineering Task Force
ILWIS	Integrated Land and Water Information System
ITC	Faculty of Geo-Information Science and Earth Observation
ITIS	Integrated Taxonomic Information System
JOSM	Java OpenStreetMap Editor
JSON	JavaScript Object Notation
KML	Keyhole Markup Language
LGPL	Lesser General Public License
OASIS	Organization for the Advancement of Structured Information Standards
ODC	Open Data Commons
OGC	Open Geospatial Consortium
OpenJUMP	Open Java Unified Mapping Platform
ORM	OGC Reference Model
OSGeo	Open Source Geoespatial Foundation
OSM	OpenStreetMap
OWS	OGC Web Services
REST	REpresentation State Transfer
RPC	Remote Procedure Calls
SAAS	Software as a service
SAGA	System for Automated Geoscientific Analysis
SB	Standards Baseline
SensorML	Sensor Model Language
SFS	Simple Features – SQL
SLD	Styled Layer Descriptor
SOA	Service-Oriented Architecture
SOS	Sensor Observation Service
SPIDER	Space-Based Information for Disaster Management and Response
SPS	Sensor Planning Service
SWE	Sensor Web Enablement
TC	Technical Committee
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
UT	University of Twente
VGI	Volunteered Geographic Information
W3C	World Wide Web Consortium

WCS	Web Coverage Service
WCPS	Web Coverage Processing Service
WfMC	Workflow Management Coalition
WFS	Web Feature Service
WFS-T	Web Feature Service - Transactional
WGS84	World Geodetic System 1984
WMS	Web Map Service
WMTS	Web Map Tile Service
WPS	Web Processing Service
WSDL	Web Services Description Language
WWW	World Wide Web

CAPÍTULO 1

INTRODUCCIÓN



1. INTRODUCCIÓN

En la Conferencia de la ONU sobre el Medio Ambiente y el Desarrollo celebrada en Río de Janeiro en 1992, se aprobó una importante resolución con el fin de alcanzar un desarrollo sostenible y proteger el medioambiente. Se constituyeron una serie de medidas para hacer frente a la deforestación, la contaminación, la falta de la reserva de peces y el tratamiento de residuos tóxicos, entre otros. En esa Cumbre, se calificó a la Información Geográfica como crítica en relación con la toma de decisiones a todos los niveles (nacional, regional y global). Los encargados de tomar decisiones pueden beneficiarse de la localización, acceso y uso de la información geográfica en las áreas donde es necesario hallar soluciones a la delincuencia, al desarrollo empresarial, a la reducción de daños por desastres naturales, a la recuperación medioambiental, a la recuperación después de desastres, etc. Para ello es necesario que la información geográfica disponible cumpla una serie de necesidades o requisitos:

- **Información actualizada**: Bien por la acción del hombre o por causas naturales, La información geográfica es muy cambiante ya que las características de la Tierra son poco estables. Para tomar decisiones bien fundadas es necesario disponer de datos actualizados. La información geográfica es cara pues los medios para conseguirla son costosos. Por tanto las tomas de decisión deben hacerse sobre conjuntos de datos actualizados. La actualización implica siempre un gasto considerable.
- **Información instantánea**: La toma de decisiones en los momentos críticos requiere que la información esté disponible de forma inmediata. Esto implica que los centros de distribución de información deben tener agilidad a la hora de facilitar dicha información.
- **Acceso generalizado**: La información está en manos de quien la produce o la distribuye (instituciones, organismos, empresas, universidades, ...). El acceso más rápido, generalizado y extendido se realiza a través de Internet.

Una Infraestructura de Datos Espaciales (IDE) es un sistema informático integrado por un conjunto de datos y servicios, descritos a través de sus metadatos, que se gestionan a través de Internet, conforme a unos estándares y a acuerdos políticos. Los estándares regulan y garantizan la interoperabilidad de sus datos, mientras que los acuerdos políticos permiten que un usuario, a través de un navegador Web, pueda encontrar, visualizar, acceder y preparar la información geográfica según sus necesidades.

El sistema de información está integrado por una serie de recursos informáticos (programas, catálogos de datos, catálogos de servicios, servidores de mapas, páginas Web, etc. La información geográfica gestionada por una IDE pueden ser ortoimágenes (imágenes aéreas corregidas), imágenes de satélite, mapas, topónimos, etc. Esta información debe concordar con ciertas normas y estándares, del mismo modo que los recursos informáticos lo hacen con los protocolos y las interfaces para garantizar la interoperabilidad.

Uno de los objetivos de las IDE es poder compartir la información geográfica dispersa en Internet para poderla visualizar y utilizar en la medida de lo posible. Para esto es necesario que los dispositivos se entiendan entre sí mediante protocolos de comunicación compartidos. Además, todas las máquinas que usen los datos a compartir deben deben ser capaces de entenderlos y utilizarlos.

La definición que ofrece la norma ISO 19119 sobre la **interoperabilidad** dice que esta es la capacidad para comunicar, ejecutar programas, o transferir datos entre varias unidades funcionales sin necesitar que el usuario tenga conocimiento de las características de esas unidades. Dos sistemas de Información tendrán interoperabilidad geográfica si:

- Pueden intercambiar libremente información espacial.
- Ejecutan software distribuido para manipular esa información espacial a través de las redes.

Dos componentes X e Y de un sistema son interoperables si X puede enviar peticiones R de servicios a Y, basados en el entendimiento común de R por X e Y, y si Y puede devolver igualmente respuestas S comprensibles para X.

El hecho de que dos sistemas interoperables se entiendan entre sí conduce a la posibilidad de que puedan transferirse los datos (independientemente del sistema de coordenadas, pertenecia al mismo huso, unidades de medida, etc.) y entenderse sus significados.

El acceso e interoperabilidad de la información cumplen un papel fundamental en una situación de emergencia medioambiental. El rol del usuario de una IDE es el de observador y este analizará cómo actúa el personal técnico de Protección Civil ante una situación de alarma. Es necesario destacar una IDE hace posible la interoperabilidad y el flujo de información, proveniente de distintas bases de datos y distintos organismos.

En una IDE, entendida como sistema distribuido en la red, intervienen todo tipo de organismos y entidades, conocidos como actores, cada uno con su papel:

- **Productores de datos**: Capturan, producen y difunden datos geográficos (mapas, modelos digitales del terreno, imágenes, ortofotos, etc.) a través de servicios de visualización, de descarga, de consulta, etc. Habitualmente estos productores son organismos públicos, como el Instituto Geográfico Nacional (IGN) en España, la Dirección General del Catastro o el Instituto Nacional de Estadística. Hoy en día, gracias a las nuevas tecnologías emergentes, cualquier individuo puede producir datos actualizados desde su PC o incluso desde un simple smartphone, sin coste alguno. A esto se le conoce como Información Geográfica Voluntaria (IGV).
- **Desarrolladores de software**: Diseñan y crean los programas y aplicaciones que permiten publicar un servicio software o implementan un geoportal desde el que

poder visualizar y utilizar los datos. Estos desarrolladores suelen ser una empresa privada o una universidad.

- **Intermediarios (brokers)**: Generalmente son empresas privadas que adaptan e integran las soluciones y componentes existentes para proporcionar un sistema completo y a la medida para usuarios y organizaciones no expertos.
- **Universidades**: Su propósito es investigar e innovar. Desarrollan algoritmos, métodos, programas y soluciones que no existen en el mercado, para que la tecnología progrese y evolucione.
- **Usuarios**: Demandan información y utilizan los servicios que proporciona una IDE para solucionar sus problemas. El usuario es el actor más importante de una IDE. Los usuarios pueden ser ciudadanos individuales, organismos públicos, empresas privadas, universidades, asociaciones o cualquier agente social. Todo se hace pensando por y para él. Cada vez se valora más su opinión, su capacidad de decisión así como su grado de satisfacción.

En este trabajo fin de máster especializado en sistemas de información (SI) nos centraremos precisamente en la producción y en el flujo de datos geográficos a partir de la colaboración voluntaria por parte de los ciudadanos y no tanto en aquella que generan las instituciones u organismos oficiales.

1.1. INTEROPERABILIDAD Y OPEN GEOSPATIAL CONSORTIUM

En la última década se está produciendo una clara evolución tanto de los Sistemas de Información Geográficos (SIG) como de las tecnologías geoespaciales en general. Solamente un reducido grupo de científicos y profesionales en la gestión del territorio utilizaba este tipo de tecnologías hace unos, pero en la actualidad su ha generalizado ámplia y libremente su uso. Este hecho se debe en gran medida a Internet, la aparición de la Web 2.0 así como otras iniciativas innovadoras como las de Google, concretamente GoogleMaps y Google Earth en los últimos años así como el fenómeno imparable de los Mashups, basados muchos de ellos en servicios geográficos. De este modo se está pasando de un entorno opaco, con software monolítico, formatos propietarios incompatibles y pocos datos digitales a un panorama mucho más integrado en la corriente general de las Tecnologías de la Información y Comunicación (TIC), orientado a los sistemas distribuidos y servicios Web, donde los estándares y el intercambio cobran cada vez más importancia.

Para permitir este nuevo entorno, variado y complejo, es necesario trabajar y profundizar en la interoperabilidad. Ahí es donde el Open Geospatial Consortium (OGC) [1] cumple su cometido. El OGC es un consorcio internacional constituido por diferentes organismos gubernamentales, empresas y universidades que lidera la generación de estándares en el área geoespacial. Dentro de estos estándares destacan los OGC Web Services (OWS), un conjunto de servicios web definidos que utilizan estándares de Internet no propietarios: WWW, HTTP, URLs, tipos MIME y el lenguaje XML.

Los estándares WMS (Web Map Service) y WFS (Web Feature Service) suponen importantes avances en este área, favoreciendo a la publicación de mapas y datos espaciales a través de la Red. Otro estándar importante es el WPS (Web Processing Service) permite publicar y consumir operaciones remotas de geoprocesamiento que está suponiendo un avance fundamental en los SIG.

1.2. MOTIVACIÓN Y PLANTEAMIENTO DEL PROBLEMA

Desde su aparición, la Información Geográfica Voluntaria (IGV) se ha convertido en una sensación y hace referencia claramente a la información geoespacial producida por personas sin una formación específica en campos como la geografía o la cartografía entre otros. La IGV es un caso especial de lo que está ocurriendo dentro de la corriente principal de la evolución de la Web 2.0. Esta información geográfica se crea, maneja y difunde generalmente a través de redes de voluntarios, dispositivos y software que están interconectados por internet. En situaciones críticas, la IGV ofrece un nuevo dispositivo para establecer gigantescos esfuerzos voluntarios dentro de un tiempo limitado. Las innovaciones en la computación geoespacial aconsejan nuevas herramientas y procedimientos para informar sobre la tarea a producir y analizar los datos generados por los ciudadanos. Este tipo de información sería imposible sin el acceso general a internet, preferiblemente a través de una conexión de alta velocidad. Hoy en día en los países desarrollados millones de personas disponen de dichas conexiones: tecnologías por satélite, cable o telefonía móvil [2].

En situaciones catastróficas, como la del terremoto de Haití en 2010, los servicios de emergencia tienen que saber dónde se encuentran las personas que más lo necesitan y cómo obtener asistencia y socorrerles. Grandes partes de Haití carecían de suficiente cobertura en los servicios de mapas Web estándar como Google Maps. Haití, uno de los países más pobres del mundo, no presentaba la suficiente demanda de mapas online. A partir de este suceso, la demanda de información espacial y mapas online aumentó de forma extraordinaria [3]. Resulta evidente que pueden salvarse más vidas y correrse menos riesgos si el flujo en la localización de la información geográfica es rápido. Obviamente pueden producirse duplicidades en las labores de producción cartográfica (por ejemplo, que varios voluntarios digitalicen la misma calle) aunque la velocidad con la que se producen estos conjuntos de datos mitiga en gran medida este hecho. Además, la duplicidad de información no tiene porque verse como algo negativo, ya que esta puede dar lugar a muchas posibilidades de acceso a la información [3]. Algunos de los sitios Web más populares donde los voluntarios pueden producir y distribuir este tipo de información son Google Earth, OpenStreetMap, Ushahidi, MapTube, GeoCommons, Wikimapia, ArcGIS Explorer Online y Panoramio, entre otros.

Para la realización de este trabajo fin de máster se ha realizado el diseño de un prototipo de aplicación Web basada en la información geográfica voluntaria (a modo de visualizador cartográfico) con el objetivo de mejorar la interoperabilidad entre diferentes

servicios. Así, uniendo fuerzas, se tiene toda la información espacial producida por ciudadanos proveniente de los sitios Web más populares en la red, disponible de forma actualizada y en un único geoportal. Aquí la innovación se encuentra en el hecho de aunar servicios, traer un espacio donde la gente pueda compartir información y tratar de mejorar la situación actual gracias a las nuevas tecnologías. La información geográfica estará disponible gracias a la utilización de distintos servicios Web que ofrecen cada una de las plataformas IGV. En principio, esta aplicación no provee herramientas que faciliten tareas de edición ni producción cartográfica, solamente permite la visualización de los datos geo-espaciales. Además, también se describe, a modo de mejora, un modelo que puede ayudar a que la producción de información geográfica voluntaria sea más rápida evitando las duplicidades. Este modelo consistiría en la incorporación de una nueva capa, llamada mapa de estados, que indicaría al usuario (voluntario) las zonas en las que es más necesaria o urgente su ayuda, en base a ciertos hechos: número de mediciones, tiempo transcurrido desde la última observación, variabilidad de las mediciones, etc. Para ello ha sido necesario estudiar y analizar las herramientas ofrecidas por cada uno de los servicios disponibles, así como también escoger la tecnología apropiada o que mejor se ajuste al prototipo deseado, para crear el prototipo de aplicación Web orientada a la información geográfica voluntaria satisfactoriamente.

1.3. PLANIFICACIÓN DEL POYECTO

La siguiente tabla muestra cuál ha sido el calendario de ejecución de este trabajo.

Actividad	**Ene**				**Feb**				**Mar**				**Abr**				**May**				**Jun**				**Jul**			
Revisión de literatura complementaria																												
Establecer los requisitos																												
Implementación																												
Diseño del sitio Web																												
Visualización de las aplicaciones IGV																												
Implementación de las aplicaciones IGV																												
Implementación de otras funciones																												
Análisis de resultados (investigación)																												
Escribir el informe o memoria final																												

Tabla 1 - Planificación del proyecto

1.4. FINALIDAD

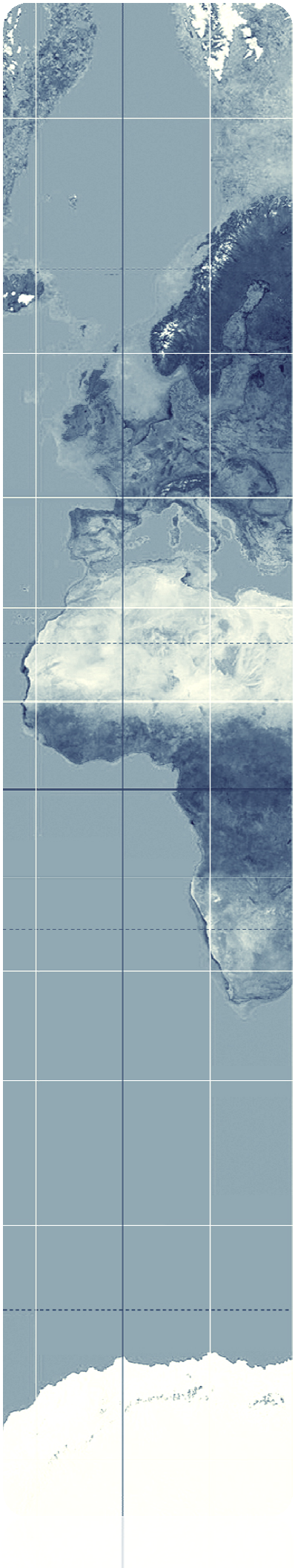
Este proyecto ha sido realizado con el objetivo de obtener el título de Máster Universitario en Ingeniería de Software, Métodos Formales y Sistemas de Información (MISMFSI) por la Universidad Politécnica de Valencia (UPV), ya que éste supone un requisito exigido. En concreto, este Trabajo Fin de Máster (TFM) tiene una orientación básica de tipo profesional, es decir, que en ella se expone un desarrollo de tipo práctico, relacionado con la actividad profesional propia de un titulado en informática. El TFM está pensado para realizar un trabajo de síntesis de los conocimientos adquiridos en

otras asignaturas impartidas en el máster. Esto requiere poner dichos conocimiento en práctica de forma conjunta en un trabajo eminentemente práctico y centrado en el ejercicio profesional de la informática.

En este caso, el TFM pertenece al área de Sistemas de Información, y más concretamente a los Sistemas de Información Geográfica (SIG).

CAPÍTULO 2

INFORMACIÓN GEOGRÁFICA VOLUNTARIA



2. INFORMACIÓN GEOGRÁFICA VOLUNTARIA

En este capítulo se describe el uso de información espacial para la gestión del riesgo y los desastres naturales. La importancia de los datos provenientes de disciplinas como la teledetección, los sistemas de información geográficos (SIG) y los sistemas globales de navegación por satélite se destacan comparando estudios que utilizan estas tecnologías para la gestión de catástrofes naturales. La información espacial compartida se debate en el contexto del establecimiento de las infraestructuras de datos espaciales (IDEs) para desastres naturales. Al analizar algunos ejemplos de aplicación de dichas infra-estructuras en la gestión de catástrofes, la necesidad de la participación de diferentes organizaciones y gobiernos de todo el mundo se ve reforzada para facilitar el intercambio de información y mejorar los planes preventivos y de emergencia. Además, se examina la posible participación de los ciudadanos en el proceso de gestión del riesgo de desastres al suministrar datos recogidos de aplicaciones de información geográfica voluntaria [4].

2.1. ORÍGENES

La frecuencia y la intensidad de los desastres naturales han crecido en las últimas décadas. Entre los años 2001 y 2010 ocurrieron más de 4,000 desastres naturales en todo el mundo, provocando la muerte de más de 1.200.000 personas. El alto crecimiento de la población, una intensa urbanización, industrialización así como una ocupación del territorio desordenada originan la presencia de poblaciones de alta densidad en zonas de riesgo, hecho que podría ser responsable del incremento de dichas catástrofes naturales [5]. La comunidad internacional reconoce que la magnitud y ocurrencia de estos sucesos, así como el número de víctimas, está aumentando considerablemente [6]. Los principales responsables en la pérdida de vidas humanas son la falta de planificación urbanística así como la ocupación desordenada del territorio.

Algunos desastres naturales ocurren de manera violenta y afectan a grandes áreas, además es difícil desarrollar planes preventivos para fenómenos naturales como los maremotos (tsunamis), tornados, terremotos, … Por otro lado, las inundaciones y los desprendimientos de tierra tienden a ser cartografiados más fácilmente, y la gente que potencialmente será afectada puede ser prevenida con anterioridad ya que se conocen las áreas vulnerables en las que suceden tales sucesos [7]. Debido a su amplia aplicabilidad, las técnicas de teledetección, los sistemas de información geográficos y los sistemas globales de navegación por satélite se han convertido en herramientas para la gestión de riesgos y catástrofes [8]. Sin embargo, para utilizar las técnicas de teledetección y los sistemas de información geográfica para la gestión de riesgos y desastres naturales, los datos y la información debe estar disponible. Por otra parte, el gobierno necesita disponer de un personal técnico preparado para manejar y analizar la información asociada. La comunidad científica propuso una infraestructura de datos espaciales que se implementó como alternativa para abordar los temas de datos compartidos.

Las Naciones Unidas estableció la plataforma de respuesta y gestión de desastres basada en información espacial, conocida internacionalmente con las siglas SPIDER, para asegurar que las organizaciones y los países tengan acceso a la misma y desarrollen la capacidad de utilizar información espacial para apoyar a todo el ciclo de gestión de riesgos y desastres. El principal propósito de esta plataforma es promover el uso de información proveniente de los satélites para observar las condiciones geológicas, climatológicas e hidrológicas para facilitar la planificación, mitigación y la rápida respuesta en las catástrofes naturales [9].

El uso de técnicas de teledetección, sistemas de información geográficos y sistemas globales de navegación por satélite son muy importantes en la gestión de desastres naturales y los debates sobre el estado real de las aplicaciones y metodologías que permiten la identificación de lagunas en la usabilidad y proporciona oportunidades para facilitar el uso de datos. Además, el establecimiento de datos e información intercambiables así como el uso de iniciativas, tales como la implementación de una IDE, son muy importantes.

El uso de información espacial en la gestión de desastres naturales demuestra la importancia de mapas en situaciones de emergencia. La información espacial puede utilizarse para reconocer las áreas afectadas, localizar e identificar los objetos necesarios, vías de rescate, relocalizaciones, la distribución de comida y medicinas para las áreas afectadas, planificar las acciones para mitigar el problema, etc. Sin embargo, los mapas convencionales pueden no ser efectivos en estas situaciones dependiendo del grado de destrucción porque el área afectada por el desastre puede ser altamente modificada, invalidando así los mapas que hayan sido realizados días antes a la ocurrencia de la catástrofe [10]. Así, esta tecnología es de gran valor, ya que permite una comunicación más rápida y elimina el tiempo perdido. Desde el terremoto de Haití en 2010, se han hecho muchos esfuerzos en colaboración con la sociedad civil para producir mapas en áreas afectadas por los desastres naturales, y esta colaboración ha sido posible gracias al desarrollo de tecnologías de información y comunicación, tales como el GPS, la web 2.0 y la telefonía móvil.

La contribución en la creación de información geográfica por parte de un gran número de ciudadanos, con pocos o nulos conocimientos en la materia, llama mucho la atención, ya que esta era una función que durante siglos ha sido reservada para instituciones oficiales. Estos ciudadanos están actuando voluntariamente, y su información puede no ser muy precisa. Sin embargo, en conjunto, esto representa una gran innovación que está teniendo un fuerte impacto en los sistemas de información geográfica y se define como Información Geográfica Voluntaria (IGV) [2], es decir, información geográfica obtenida por ciudadanos, de forma voluntaria y colectiva, sin necesidad de tener algún tipo de titulación [11].

Las personas pueden comportarse como sensores, ya que durante toda su vida adquieren conocimientos acerca de aquellos lugares donde viven, trabajan o visitan, como por ejemplo los nombres de lugares o topónimos, las características topográficas,

las redes de transporte, etc. Un ser humano puede considerarse como si se tratara de un sensor móvil inteligente que está equipado con capacidades de interpretación e integración que varían de acuerdo a las experiencias de cada persona. Estas habilidades pueden ser mejoradas a través del uso de teléfonos móviles con un GPS incorporado, cámaras digitales y dispositivos de seguimiento [12]. Las personas, que antes eran consideradas como usuarios pasivos, han pasado a ser considerados como usuarios activos a la hora de producir y compartir datos [13].

De acuerdo a [14], tanto el contenido de la información como las tecnologías utilizadas para adquirir esta información, las cuestiones sobre la calidad, los métodos y técnicas relacionadas; y los procedimientos sociales involucrados en la creación e impacto de la información geográfica voluntaria crean una plataforma diferente para adquirir, compartir, diseminar y utilizar información geográfica. Aunque todavía persisten muchas dudas sobre la IGV, como las razones que conducen a las personas a contribuir información, la calidad de los datos y los métodos adecuados para la síntesis y análisis de dicha información geográfica, la gran cantidad de datos disponibles a través del sistema IGV constituye una fuente rica e inmediata de información geográfica para varios propósitos. Recientemente el número de páginas web que permiten a los usuarios contribuir a un rango diversificado de información geográfica se ha visto claramente incrementado: Ushahidi o Crowdmap, WikiMapia, OpenStreetMap, GeoCommons / GeoIQ, FixMyStreet, kosmosnimki y WhoIsSick, entre otras.

Con respecto a la calidad de los datos, esta puede evaluarse mediante métodos de comparación [11], es decir, que la información puede compararse con otra información de la misma temática. Otra indicación puede ser el número de personas que contribuyen o facilitan información, los hechos acontecidos en lugares más poblados tienden a ser más precisos que los hechos sobre lugares menos poblados. Además, en muchas ocasiones, la revisión de los datos es conducida por un grupo de voluntarios, con lo cual se reducen también el número de posibles errores [14]. Lo ideal sería que las instituciones que producen información espacial crearan y mantuvieran más infraestructuras de bases de datos espaciales focalizadas en los desastres naturales, con el fin de generar información que pueda ser de utilidad para la gestión del riesgo y los desastres naturales. Además, el intercambio de conocimientos del análisis de los datos en los sistemas de soporte y toma de decisiones debe ser frecuente y dinámico, aunque este sólo será posible con el establecimiento de tratos y acuerdos políticos.

2.2. PLATAFORMAS

Imagínese un modo en que gente de todo el mundo cuente la historia de lo que le sucedió, ya sea directamente o a su alrededor, durante una catástrofe o situación de emergencia. La herramienta en cuestión debería ser fácil de usar, algo que casi cualquier persona pudiera hacer y que además fuese desarrollable en todo el mundo. Estas son las razones por las que muchos desarrolladores han creado varias herramientas IGV tales como las ya mencionadas Ushahidi, OpenStreetMap, Wikimapia, Geocommons y

muchas otras. A continuación se describen las herramientas de información geográfica voluntaria más utilizadas en la actualidad.

2.2.1. USHAHIDI/CROWDMAP

Ushahidi, que significa "testimonio" en Swahili, fue desarrollado en el Congo para generar mapas dinámicos dedicados a gastionar estados en alerta o en crisis (como crisis políticas, desastres naturales y conflictos locales), es decir, que fue originalmente creado para colaborar colectivamente con información referida a cualquier crisis. Surgió a raíz de la violencia provocada por la elecciones de 2008 en Kenya. Esta herramienta mantuvo a los kenianos al corriente de información vital, proporcionando asistencia y alivio. Desde entonces Ushahidi ha crecido convirtiéndose en una gran plataforma gratuita que ha llegado a un gran número de comunidades en todo el mundo. Además, la herramienta es de código abierto, con lo cual cualquiera que lo desee puede mejorar el servicio. Ushahidi combina varios servicios Web, tales como los mapas, las bases de datos, las herramientas de manipulación de datosy la funcionalidad visual, entre otros.

Ushahidi fue utilizado como asistente durante el terremoto de Haití en 2010 así como también para rastrear la violencia en Gaza por Al Jazeera, para ayudar a supervisar las elecciones de 2009 en la India, para ayudar a reunir informes a nivel mundial sobre un brote de gripe porcina que hubo hace unos años, etc. Ushahidi también ha sido utilizado en otras situaciones de crisis para proporcionar ayuda a las victimas así como a las organizaciones no gubernamentales y autoridades en la respuesta a estas situaciones o sucesos [15]. Cualquiera puede contribuir información, ya sea mediante un simple mensaje de texto vía teléfono móvil (SMS), una fotografía o video desde un smartphone, un informe online, un correo electrónico entre otras formas disponibles en el portal Web. Ushahidi puede recopilar información desde cualquier dispositivo con una conexión de datos digital. Una vez el informe ha sido enviado éste es publicado casi en tiempo real en un mapa interactivo que puede visualizarse tanto en un ordenador como en un teléfono inteligente o smartphone. Pero la característica más poderosa que ofrece Ushahidi es la capacidad de tomar la base de la aplicación y mostrarla al usuario para satisfacer las necesidades de su comunidad. Gracias a Ushahidi es más fácil que nunca obtener información crítica y oportuna para aquellos que más la necesitan, en un escenario que casi cualquiera puede utilizar.

La creciente comunidad de desarrolladores está trabajando constantemente para mejorar y hacer llegar Ushahidi a tanta gente como sea posible - incluyendo el desarrollo de aplicaciones de los dispositivos móviles más populares que existen en la actualidad.

Crowdmap está diseñado y creado por personal bajo Ushahidi. A medida que la plataforma ha ido evolucionando, también lo han hecho sus posibles usos. Crowdmap permite a los usuarios configurar o establecer su propio mapa Ushahidi sin tener que instalarlo en su propio servidor web (Apache).

Crowdmap permite al usuario:

- Coleccionar información proveniente de móviles, notícias y la Web.
- Agregar dicha información a una simple plataforma.
- Visualizarla en un mapa con la línea de tiempo.

2.2.2. OPENSTREETMAP

OpenStreetMap (OSM) es un proyecto colaborativo orientado a la creación de un mapa editable de todo el mundo. Tanto la disponibilidad de la información de los mapas en gran parte del mundo como la llegada de los dispositivos de navegación por satélite portátiles de bajo coste han sido los principales impulsores para la creación y el crecimiento de OSM. Fundada por Steve Coast en 2004, el proyecto se inspiro en el éxito que tuvo Wikipedia así como el predominio de los datos cartográficos tanto en Reino Unido como en otros lugares. Desde entonces, OSM ha crecido muchísimo, ya son más de un millón de colaboradores en todo el mundo que almacenan datos con dispositivos GPS, fotografías aéreas y otras fuentes. El 6 de enero de 2013, OpenStreetMap alcanzó el millón de usuarios registrados, una minoría de los cuales contribuye a la mayoría de los contenidos. Alrededor del 30% de los usuarios ha contribuido al menos un punto a la base de datos de OpenStreetMap. Esta información colaborativa está disponible bajo la licencia abierta de la base de datos. El sitio Web está apoyado por la Fundación de OpenStreetMap, una organización sin ánimo de lucro cuya sede está en Inglaterra.

Los datos generados por el proyecto OpenStreetMap se considerados como su producción primaria o principal, incluso más que el propio mapa. Estos datos han sido favorables en comparación con fuentes de datos de propiedad, aunque la calidad de los datos varía en todo el mundo. El proyecto tiene una base de usuarios geográficamente diversa, debido a la importancia de los conocimientos locales y la realidad del terreno en el proceso de recolección de datos. Muchos de los participantes son expertos o profesionales en los sistemas de información geográfica, los cuales aportan sus datos con herramientas del Enviromental Systems Research Institute (ESRI).

Los datos del mapa inicial fueron recogidos desde cero por voluntarios que realizan estudios sistemáticos sobre el terreno utilizando una unidad de GPS de mano y una libreta, una cámara digital o una grabadora de voz. Estos datos fueron entonces introducidos en la base de datos de OpenStreetMap. Más recientemente, la disponibilidad de fotografías aéreas y otras fuentes de datos provenientes de fuentes tanto gubernamentales como comerciales ha incrementado enormemente la volecidad de este trabajo y ha permitido que los datos o información de usos del suelo sea almacenada con más exactitud por el proceso de digitalización. Cuando las grandes bases de datos espaciales están disponibles, un equipo técnico gestiona la conversión e importación de los datos.

Los estudios sobre el terreno son realizados por un cartógrafo, ya sea a pie, en bicicleta, coche o barco. Los datos del mapa se recogen por lo general mediante el uso de una unidad GPS, aunque esto no es estrictamente necesario si un área ha sido ya trazada a partir de imágenes provenientes de un satélite. Una vez que los datos han sido recogidos, éstos se introducen en la base de datos en el sitio Web del proyecto. En ese momento no existe información sobre el tipo disponible de trayectoria actualizada - podría ser, por ejemplo una autopista, un sendero, un río, etc. Así, en un segundo paso, toma lugar la edición haciendo uso de un editor de mapas construido expresamente como Java OpenStreetMap Editor (JOSM). Este mapa es realizado normalmente por el mismo cartógrafo, aunque a veces puede hacerlo otros contribuyentes registrados en OpenStreetMap. Como la recolección y carga de datos está separada de la edición de objetos, la contribución al proyecto es posible también sin necesidad de utilizar una unidad de GPS. La colocación y edición de objetos, tales como escuelas, hospitales, paradas de taxis, paradas de autobús, bares, farmacias,... se realiza basándose en los conocimientos locales de los editores contribuyentes. Algunos colaboradores comprometidos adoptan la tarea de cartografiar pueblos y ciudades enteras, incluso hay quienes organizan reuniones o fiestas de mapeo para obtener el apoyo de otros y completar así un área del mapa. Un gran número de usuarios menos activos contribuyen con correcciones y pequeñas adiciones al mapa. Algunas agencias gubernamentales han realizado datos oficiales sobre adecuadas licencias. Muchos de estos datos provienen de Estados Unidos donde los trabajos del gobierno federal están situados bajo dominio público.

Los datos de OpenStreetMap fueron publicados originariamente bajo un contenido de la licencia abierta de Creative Commons con la intención de promover el uso y redistribución gratuita de los datos. En September de 2012, se cambió la licencia por la licencia abierta de la base de datos (Open Database License) de Open Data Commons (ODC) para definir de forma más específica su relación con los datos en lugar de la representación. Toda la información añadida al proyecto necesita tener una licencia compatible con la licencia abierta de la base de datos. Esto puede incluir información fuera de los derechos de autor, dominio público u otras licencias.

El software utilizado en la producción y presentación de los datos de OpenStreetMap está disponible en muchos proyectos distintos y cada uno puede tener su propia licencia. La aplicación es mantenida por Ruby on Rails y además utiliza PostgreSQL tanto para el almacenamiento de datos de cada usuario como para la edición de metadatos. Mientras OpenStreetMap pretende ser una fuente de datos central, su mapa de representación así como su estética están destinados a ser una de las muchas opciones, algunas de las cuales destacan diversos elementos del mapa o enfatizan el diseño y el rendimiento.

Durante el terremoto de Haití, voluntarios de OpenStreetMap utilizaron imágenes de satélites disponibles para cartografiar las calles, edificios así como campos de refugiados de Puerto Príncipe en solamente dos días, creando "el mapa digital más completo de calles de Haití". Los mapas y datos resultantes han sido utilizados por varias

organizaciones, tales como el Banco Mundial, el Centro de Investigación Común de la Comisión Europea, la Oficina de Coordinación de Asuntos Humanitarios, UNOSAT entre otros, proporcionando así ayuda de emergencia. Junto con el trabajo posterior a los desastres provocados por el terremoto, el equipo humanitario de OpenStreetMap se ha esforzado mucho para crear mejores modelos de riesgo en otros países como Uganda, en colaboración con la Cruz Roja, el Banco Mundial y otras organizaciones humanitarias.

OpenStreetMap utiliza una estructura de datos topológicos constituida por cuatro elementos básicos:

- **Nodos**: Puntos con una posición geográfica, almacenados como coordenadas (pares con valor longitud y latitud) según el sistema de coordenadas geográficas mundial WGS84 que permite localizar cualquier punto sobre la Tierra (sin necesitar otro de referencia) por medio de tres unidades dadas [48]. Sin tener en cuenta el uso de los nodos en las trayectorias, estos se utilizan para representar características puntuales en el mapa, tales como puntos de interés o los picos de las montañas.
- **Trayectorias**: Listas ordenadas de nodos que representan una polilínea o incluso un polígono en caso de que la polilínea forme una trayectoria cerrada. Las trayectorias se utilizan para representar características lineales como pueden ser las calles y los ríos así como áreas o regiones como pudieran ser los bosques, lagos, parques, parkings, etc.
- **Relaciones**: Lista ordenada de nodos y trayectorias, donde cada miembro puede opcionalmente tener un rol. Las relaciones se utilizan para representra la relación existente entre los nodos y las trayectorias. Como ejemplos de relaciones se pueden incluir restricciones de giro en las carreteras, rutas que abarcan varias trayectorias existentes (por ejemplo una autopista de larga distancia) así como también las zonas con agujeros.
- **Etiquetas**: Pares clave-valor utilizados para almacenar metadatos sobre objetos en el mapa (tales como sus tipos, nombres y propiedades físicas). Las etiquetas no son independientes, sino que siempre están asociadas a un objeto, ya sea un nodo, una trayectoria, una relación o bien un miembro de una relación. Una ontología recomendada de las características del mapa (significado de las etiquetas) se mantiene en un wiki, es decir, en un sitio web cuyas páginas pueden ser editadas por múltiples voluntarios a través del navegador web.

DIFERENCIA CONCEPTUAL ENTRE USHAHIDI Y OPENSTREETMAP

Muchas instancias de Ushahidi usan OpenStreetMap como su mapa base, incluyendo la información que ya existe para dar contexto a nuevos informes en curso. La gente que utiliza Ushahidi puede hacer uso de OpenStreetMap para referenciar la información que están recolectando. Como ya se ha dicho anteriormente, Ushahidi es una plataforma que permite al usuario coleccionar y visualizar información sobre un mapa base, pero no permite generar el mapa base en sí mismo. Ushahidi se utiliza para informar sobre incidentes de todo tipo, desde informes sobre la escasez de agua hasta informes sobre

todo tipo de violencia, hasta llegar al terremoto de Haití. Todo esto se digitalizaba sobre un mapa que utilizaba OpenStreetMap como mapa base, debido a que no había buenos mapas antes del terremoto.

OpenStreetMap (OSM) es un proyecto global que necesitaba construir un mapa. Sin embargo, los mensajes de texto de Ushahidi no juegan ningún papel en la creación de dicho mapa. La mayoría de los voluntarios recolectan datos ya sea topografiando el suelo directamente como tomando datos desde un dispositivo GPS o bien trazando fotografías aéreas del terreno. Después del famoso terremoto de Haití, varios proveedores de fotografías aéreas autorizaron a la comunidad de OpenStreetMap a trazar sus imágenes aéreas de Haití. En este sentido los voluntarios construyeron el mapa. En algunas áreas, los nombres de las calles fueron añadidos utilizando viejos mapas militares de Estados Unidos en el dominio público o bien (con permiso) utilizando mapas de las Naciones Unidas. Este mapa base creado a partir de esta información fue entonces utilizado por Ushahidi.

2.2.3. WIKIMAPIA

WikiMapia es un proyecto colaborativo de cartografía de contenido abierto que tiene como objetivo marcar y describir todos los objetos geográficos del mundo. En él se combina una mapa interactivo Web con un sistema wiki georreferenciado. Más de 18 millones de objetos ya han sido marcados. En la actualidad la comunidad de Wikimapia consta de más de 1,5 millones de usuarios registrados. El contenido generado por el usuario está disponible a través de una aplicación web así como de una API bajo una licencia de Creative Commons. La principal preocupación de Wikimapia es recopilar y ordenar por categorías tanta información geográfica como sea posible en todo el mundo, y fproporcionar una oportunidad para explorar y utilizarla libremente. Wikimapia ha sido creado de forma colaborativa por voluntarios de internet.

Wikimapia se puso en marcha el 24 de mayo de 2006 por Alexandre Koriakine y Evgeniy Saveliev. Aunque el nombre del proyecto recuerda a Wikipedia adeás de compartir la filosofía "wiki", Wikimapia no forma parte de la Fundación Wikimedia, sino que está custodiada por una empresa comercial privada. Un pequeño equipo de administradores mantiene y continua desarrollando Wikimapia. Ellos introducen nuevas características y determinan con mayor precisión la evolución del entorno. Estas mejoras en general se ven influenciadas por los usuarios, así como también a través de informes, solicitudes en el sistema de seguimiento de problemas, discusiones en foros, ...

El sitio Web de Wikimapia proporciona un mapa web interactivo basado en la API de Google Maps que consiste en información generada por el usuario. El usuario puede seleccionar cualquier elemento u objeto marcado y visualizar su descripción. Además de esto y de a herramienta de búsqueda, el usuario puede también filtrar y destacar objetos por categorías. El usuario puede añadir también una nueva localización dibujando su

contorno, escribiendo la descripción del mismo, eligiendo la categoría a la cual pertenece y cargando fotografías de referencia. Solamente los usuarios registrados pueden editar las etiquetas existentes. Las características lineales como las calles, vías ferroviarias, ríos, ... pueden dibujarse y proporcionarse con una descripción así como con fotografías. Otra característica que ayuda a editar y explorar información sobre Wikimapia es un sistema de lista de observación. Cualquier usuario puede elegir y guardar una cierta área para supervisar los cambios en el mapa.

Los usuarios de Wikimapia pueden añadir información aparentemente valiosa. Esta es la razón por la que los artículos pueden contener la impresión personal de un lugar, consejos, notas de amonestación, recomendaciones o información más discutible. Así, los usuarios deben ser conscientes desde un principio de que no todos los artículos son de nivel enciclopédico. Aunque la comunidad de usuarios Wikimapia en gran medida se auto-organiza, hay un sistema de clasificación basado en puntos de experiencia que los usuarios obtienen por cada acción. El sistema está diseñado para animar a añadir nueva información y prevenir la desorganización o el vandalismo que son problemas comunes de un proyecto colaborativo. Si una persona se ha forjado una buena reputación como editor competente puede ser que consiga una propuesta para convertirse en moderador, en tal caso este recibiría una autoridad adicional para ayudar a los recién llegados, supervisar y controlar el mapa así como desabilitar a un usuario según corresponda. Wikimapia también tiene un foro desde donde se llevan a cabo la mayoría de las discusiones. En algunas zonas del mundo donde la cartografía es muy cara o no está actualizada, como es el caso de los países que están en desarrollo, el crecimiento de Wikimapia ha sido increíblemente rápido, sin embargo esto ha provocado algunos problemas. Las áreas urbanas estaban solapadas con el perímetro marcado por la posición de residencias privadas. Para evitar la saturación de los moderadores de Wikimapia, tales áreas se deben observar con atención y organizarlas de forma que sea fácil distinguir las residencias de los lugares de interés público. Además se recomienda editar o borrar las etiquetas cuando sea necesario.

En mayo de 2012 Wikimapia anunció que todo el contenido estaba disponible bajo la licencia de Creative Commons con la atribución de poder compartir la información. Esto significa que Wikimapia ofrece todos sus datos para compartir, difundir, transformar o adaptar en cualquier forma reconocible derivada de la original, para cualquier tipo de uso. Los licenciatarios deberán distribuir trabajos derivados sólo bajo una licencia idéntica a la licencia que regula la obra original.

2.2.4. GEOCOMMONS/GEOIQ

GeoCommons es una potente plataforma para compartir, visualizar y analizar datos geográficos. Los desarrolladores han trabajado mucho para poder proporcionar una interfaz fácil de usar a la hora de manejar grandes cantidades de datos y análisis complejos. Los Creadores pueden aprovechar GeoCommons como plataforma para

alojar, almacenar, recuperar y editar datos geoespaciales. Además, la API completa proporciona mapas temáticos para su construcción, incorporándolos en aplicaciones y la interacción a través de la interfaz de Javascript.

La plataforma GeoIQ alimenta la creciente comunidad de GeoCommons con más de 25 mil miembros creando y compartiendo activamente cientos de miles de datos y mapas en todo el mundo. Con GeoCommons, cualquiera puede contribuir y compartir datos, crear fácilmente mapas que pueden ser compartidos y colaborar con los demás. Ellos se dedican a asistir al usuario a través de cualquier tipo de problema que pueda surgir, así como ayudarle a aumentar su productividad y satisfacción.

Hay algunas restricciones de uso como, por ejemplo, que los usuarios tienen un número de descargas simultaneas y descargas totales limitado por día. En ese caso, los usuarios pueden contactar con GeoCommons (support@geoiq.com) para discutir el modo en el que se puede aumentar la facilidad de acceso a GeoCommons o aprender sobre la plataforma GeoIQ y cómo puede esta proporcionar acceso completo a un usuario.

La ayuda en GeoCommons está disponible a través de sus foros públicos donde los usuarios pueden hacer preguntas, sugerir nuevas ideas o conectarse con la comunidad. Una vez creada la cuenta de usuario, hacer un mapa puede ser muy fácil y rápido. Los usuarios pueden elegir cargar sus propios datos o buscar en los más de cien mil datos (entre puntos, líneas y polígonos) ya existentes en GeoCommons. Es posible subir datos en diferentes formatos (Shape, CSV, KML o GeoRSS). Si los usuarios disponen de datos básicos (nombres de países, direcciones, etc.) sin información geográfica es posible unir su propio fichero a uno ya existente en GeoCommons o crear las coordenadas longitud y latitud. Una vez que los usuarios han encontrado o bien han subido los datos que desean cartografiar, pueden elegir hacer un mapa o bien añadir a un mapa ya creado. Se permite elegir el mapa base, los estilos, las opciones de color, las etiquetas del mapa o solamente dejar estas opciones a GeoCommons. De cualquier manera, GeoCommons guiará a los usuarios a través de un proceso intuitivo y rápido. Algunas importantes herramientas de análisis espacial están disponibles para todos de forma limitada, como por ejemplo unir mapas, agregar puntos a polígonos, … Finalmente el usuario puede salvaguardar su mapa y compartirlo con otros, pertenezcan o no a la Comunidad.

2.2.5. USA-NPN

La Red de Fenología Nacional de Estados Unidos es una colaboración entre agencias federales, la comunidad académica y el público en general para establecer una ciencia nacional y controlar la iniciativa centrada en la fenología, el estudio de los ciclos de vida de las plantas y los animales, tales como la aparición de las hojas y la migración de las mariposas. Se trata de un consorcio de individuos y organizaciones que recolectan, comparten y utilizan datos fenológicos, modelos e información relacionada. La red se sirve de la ciencia y la sociedad, promoviendo un amplio conocimiento de la fenología de

las plantas y de los animales así como su relación con los cambios ambientales. A través del programa de la red (Nature's Notebook), gente de todas las edades y con todo tipo de formaciones observan y salvaguardan la actividad de los organismos como medio para descubrir y explorar la naturaleza, y el ritmo de nuestro mundo dinámico. La red fenológica crea datos, modelos e información relativa disponible libremente para capacitar a científicos, administradores de recursos y al público en general en la toma de decisiones y adaptarse a los climas variables y cambiantes. la adaptación a climas y ambientes variables y cambiantes. La Red Fenológica Nacional de Estados Unidos está constituida por una oficina de coordinación nacional, un comité de asesoramiento, y muchos accionistas (ciudadanos y profesionales científicos, administradores de recursos y educadores). Los asociados representan un rango de organizaciones que incluyen agencias públicas, organicaciones no gubernamentales, redes especializadas e instituciones académicas.

Los días 12 y 13 de julio de 2010 en Boulder, Colorado, la red fenológica organizó una revisión de su sistema de gestión de la información para examinar los problemas actuales y futuros en la tecnología de la información y garantizar que el sistema es útil y seguro hasta a la fecha. A este evento fueron invitados un grupo de expertos a modo de asesoramiento. La red busca trabajar de forma colaborativa y transparente con otras organizaciones en este campo, y mejorar las capacidades existentes con unas herramientas software de código abierto apropiadas. En este momento se está trabajando mucho para mejorar ciertos aspectos de la USA-NPN, como por ejemplo:

- Forma de asegurar la distribución de datos y su procedencia.
- Establecimiento de normas para archivar y distribuir datos fenológicos.
- Modo de verificar y validar los datos presentados por los observadores.
- Aumento de la participación de los observadores.
- Escalabilidad de los sistemas de la USA-NPN aumentando la participación de los usuarios así como los requisitos de almacenamiento de datos.

Los participantes no siempre entienden que el proceso de desarrollo de la USA-NPN se originó con necesidades claramente definidas alineadas con metas más amplias de la organización. Se recomendó que las mejoras y los nuevos proyectos se considerarán en un marco estratégico. Aunque se sugerieron muchas mejoras bien meditadas, no todas pueden ser abordadas en un futuro inmediato, dado los niveles actuales de financiación. Debe darse prioridad a aquellas mejoras que permitan a la organización alcanzar sus metas más grandes. Los datos de acceso, la calidda de los mismos, los servicios web, las herramientas de visualización así como las mejoras de seguridad específicas tienen probablemente una mayor importancia en esta etapa en el desarrollo de la organización.

Si desea más información sobre la Red Fenológica Nacional de los Estados Unidos puede vistar el sitio Web correspondiente:

https://www.usanpn.org/

¿QUÉ ES LA FENOLOGÍA?

La fenología es el calendario natural, por ejemplo cuando los cerezos florecen, cuando un petirrojo construye su nido y cuando las hojas cambian de color en otoño. La fenología es un componente clave de la vida en la Tierra.

Muchas aves calculan el tiempo de su nidificación para que los huevos eclosionen cuando haya insectos que puedan servir de alimento a los polluelos. Para muchas personas, la temporada de alergia se inicia cuando algunas flores en particular empiezan la floración; una floración temprana implica que las alergias también lo sean. Los agricultores y los jardineros tienen que saber cuándo plantar para evitar las heladas, y necesitan conocer el calendario del desarrollo de plantas e insectos para decidir cuándo aplicar fertilizantes y pesticidas. En la naturaleza muchas interacciones dependen de una temporización. De hecho, la fenología afecta a casi todos los aspectos del medio ambiente, incluyendo la abundancia, distribución y diversidad de organismos, servicios de los ecosistemas, cadenas alimentarias así como los ciclos globales del agua y del carbono.

Los cambios en los acontecimientos fenológicos como la floración y las migraciones de aves son algunas de las respuestas biológicas más sensibles al cambio climático. En todo el mundo, muchos procesos fenológicos primaverales se dan cada vez más pronto y los otoñales cada vez más tarde de lo que solía ser en el pasado. Sin embargo, no todas las especies y regiones están cambiando al mismo ritmo, dando lugar a muchos desajustes. El modo en que las plantas y los animales responden al cambio climático puede ayudar a predecir si sus poblaciones aumentarán o disminuirán, haciendo de la fenología un indicador principal de los impactos del cambio climático.

La fenología juega un papel importante en la cultura humana. Se organizan muchas festividades en todo el mundo para celebrar acontecimientos de tipo fenológico desde las migraciones de las ballenas hasta las floraciones de los cerezos.

Se sabe mucho sobre fenología, pero aún falta por conocer. Los voluntarios y personas interesadas pueden ayudar de varios modos:

- Recolectando datos directamente tomados en sus propios jardines, parques cercanos o como parte de un estudio.
- Organizando un esfuerzo fenológico de forma local para la recolección de datos, investigación y/o educación.
- Participando en concentraciones relacionadas con la fenología.

MODELO DE DATOS

Uno de los principales objetivos de la oficina de coordinación de la USA-NPN es proporcionar datos de buena calidad para reforzar e impulsar la investigación fenológica. Por esta razón, se diseñó un modelo de datos consultando a los científicos sobre sus

necesidades en cuanto a los datos fenológicos en términos de escala, cobertura, especies, fenofases, y fiabilidad. El modelo de datos también se ocupa de la necesidad de la integración de los datos existentes, la posibilidad de transferir el modelo de datos, y la transportabilidad y la integridad de los datos.

La figura 1 proporciona una versión simplificada del modelo de datos con los elementos clave que lo constituyen:

- **Persona**: Observador/Usuario.
- **Estación**: Localización donde las mediciones han sido realizadas.
- **Especies en la estación**: Estación en la que una especie de planta o animal ha sido localizada (para animales, este registro se encuentra a nivel de especie; para las plantas se encuentra a nivel de organismo individual, ya que el seguimiento de las plantas a través del tiempo así como su marcado es más factible.
- **Especies**: Especies de plantas y animales a observar. Se incluye el número taxonómico, distribuciones y otra información secundaria.
- **Red**: Afiliación para personas, especies y estaciones.
- **Protocolo**: Cconjunto de fenofases y sus definiciones.
- **Fenofase**: Una etapa definida del ciclo de vida (por ejemplo, la caída de las hojas de los árboles o el noviazgo en animales adultos).
- **Observación**: Valor del estado de la fenofase (Si/No/Incierto), además de la abundancia y la intensidad o porcentage de una especie de animal o planta, observada en una estación por una persona.

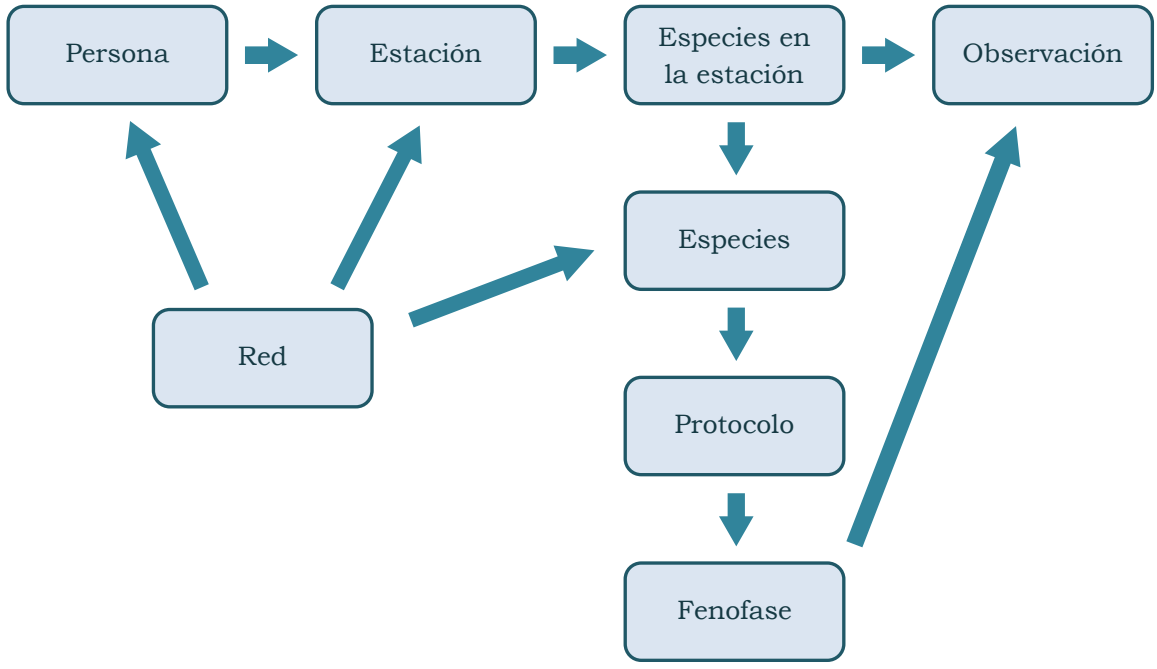
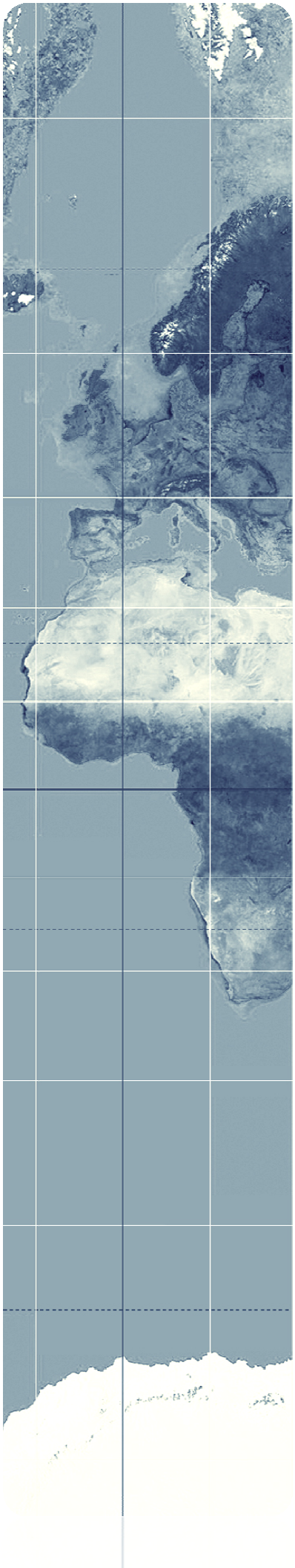


Figura 1 - Tabla de observación: Entidades clave y sus interrelaciones en el modelo de datos de la USA-NPN.

CAPÍTULO 3

SERVICIOS WEB BASADOS EN REST



3. SERVICIOS WEB BASADOS EN REST

Los Servicios Web son sistemas software diseñados para soportar una interacción interoperable máquina a máquina sobre una red. Dichos servicios suelen ser APIs Web que pueden ser accedidas dentro de una red (principalmente Internet) y ser ejecutados en el sistema donde están alojados. Gracias a su utilización e implementación en el prototipo de este trabajo, ha sido posible la visualización de la información geográfica voluntaria proveniente de las diferentes plataformas o infraestructuras existentes. El acrónimo REST (Representational State Transfer) fue definido por Roy Fielding en su tesis doctoral [16]. Roy es uno de los principales autores de las versiones 1.0 y 1.1 de la especificación del protocolo HTTP. REST es un estilo de arquitectura software para sistemas distribuidos como el World Wide Web (WWW) bastante popular en la actualidad y supone otra opción de estilo de uso de los Servicios Web, así como RPC y SOA. Por tanto debe quedar claro que comparar REST con los Servicios Web es erróneo ya que se trata de conceptos diferentes: REST es un estilo, mientras que los Servicios Web son sistemas software. Es posible diseñar Servicios Web basados en REST, es decir tomando REST como estilo de diseño. Estas arquitecturas consisten en clientes y servidores. Los clientes realizan peticiones a servidores; los servidores procesan dichas peticiones y devuelven las respuestas apropiadas a tales peticiones. Las peticiones y las respuestas se crean sobre la transferencia de las representaciones de los recursos web. Una representación de un recurso web es un documento que contiene el estado del recurso.

REST, además de controlar el buen funcionamiento de los participantes, se ha convertido en un modelo de diseño de servicio web predominante. A pesar de que REST no es un estándar, está basado en estándares: HTTP, URL, Representación de los recursos (HTML, XML, JPEG, ...), Tipos MIME (text/html, text/xml, ...). REST ha sido aplicado para describir la arquitectura Web deseada, para ayudar a identificar problemas existentes, para comparar soluciones alternativas, y para asegurar que las extensiones del protocolo no violan las restricciones básicas que hacen exitosa la Web.

Características	▪ Las operaciones se definen en los mensajes. ▪ Una dirección única para cada instancia del proceso. ▪ Cada objeto soporta las operaciones estándares definidas. ▪ Componentes débilmente acoplados.
Ventajas	▪ Bajo consumo de recursos. ▪ Las instancias del proceso se crean de forma explícita. ▪ El cliente no necesita información de enrutamiento a partir de la URI inicial. ▪ Los clientes pueden tener una interfaz genérica para las notificaciones. ▪ Fácil de construir y adoptar.
Desventajas	▪ Gran número de objetos. ▪ Manejar el espacio de nombres (URIs) puede ser complejo. ▪ La descripción sintáctica/semántica es muy informal (orientada al usuario). ▪ Pocas herramientas de desarrollo.

Tabla 2 - Características de REST

Un concepto importante en REST es la existencia de recursos (fuentes de información específica), cada uno de los cuales está referenciado a un identificador global (por ejemplo, un URI en HTTP). Para manipular estos recursos, los componentes de la red (agentes del usuario y servidores de origen) se comunican mediante una interfaz estandarizada (HTTP) y unas representaciones de intercambio de dichos recursos. Por ejemplo, un recurso que representa un círculo (un objeto lógico) puede aceptar y devolver una representación que especifica el centro de un punto y un radio, pero también puede aceptar y devolver una representación que especifica tres puntos diferentes sobre una curva (ya que esto también identifica de forma unívoca un círculo) como una lista separada por comas.

Una aplicación puede interactuar con un recurso sabiendo dos cosas: el identificador del recurso y la acción requerida. No es necesario saber si hay caches, proxys, pasarelas, cortafuegos (firewalls), túneles o alguna otra cosa entre medias con el servidor que actualmente soporta la información. Sin embargo, la aplicación necesita comprender el formato de la representación devuelto, que es típicamente un documento HTML, XML o JSON, aunque puede ser una imagen, texto plano o algún otro contenido.

Una API REST es una Interfaz de Programación de Aplicaciones (API), o librería de funciones, a la que se accede mediante el protocolo HTTP. Se accede a una REST API a través de direcciones web o URLs en las que se envían los datos de una consulta. Como respuesta a esa consulta sobre el REST API se obtienen datos en diferentes formatos (texto plano, HTML, XML, JSON, etc).

3.1. OBJETIVOS Y RESTRICCIONES

Los objetivos principales de REST son:

- La escalabilidad de la interacción de los componentes. Gran variedad de clientes (estaciones de trabajo, dispositivos móviles, …) pueden acceder a través de la Web.
- Generalidad de interfaces. Cualquier cliente puede interactuar con cualquier servidor HTTP sin ningún tipo de configuración especial, gracias al protocolo HTTP.
- Desarrollo de componentes independiente. Debido a que los clientes y los servidores pueden estar puestos en funcionamiento durante muchos años, éstos deben ser capaces de comunicarse correctamente con los clientes y los servidores que haya en la actualidad. Esto supone un desafío ya que diseñar un protocolo que permita este tipo de características resulta muy complicado.
- Compatibilidad con componentes intermedios. Los más populares intermediarios para la Web son los proxys (algunos de ellos, como las caches, mejoran el rendimiento, mientras que otros fortalecen las políticas de seguridad) y los gateways, que permiten encapsular sistemas que no son propiamente de la Web.

REST satisface tales objetivos mediante la aplicación de una serie de restricciones:

- C/S: Una interfaz uniforme separa clientes de servidores. Por un lado, a los clientes no les afecta el almacenamiento de datos que permanece internamente en cada servidor, así que la portabilidad del código cliente es mejorada. Y por otro lado, a los servidores no les influye el estado o la interfaz de usuario, así los servidores pueden ser más simples y escalables. Tantos los servidores como los clientes pueden ser reemplazados y desarrollados de forma independiente, siempre que la interfaz entre ellos permanezca inalterada.
- Stateless: Cada petición de cualquier cliente contiene toda la información necesaria para atender la solicitud, y cualquier estado de sesión se mantiene en el cliente.
- Cacheable: los clientes pueden almacenar en caché las respuestas de los servidores. Estas respuestas deben definirse, de manera implícita o explícita, como almacenable en caché o no, para evitar que los clientes reutilicen datos obsoletos o inadecuados en respuesta a otras peticiones.
- Sistema de capas: Un cliente no puede decir que está conectado directamente al servidor final, o a uno intermediario. Los servidores intermediarios pueden tanto mejorar la escalabilidad del sistema como hacer cumplir las políticas de seguridad.
- Demanda de código (opcional): Los servidores pueden extender o customizar temporalmente la funcionalidad de un cliente mediante la transferencia de código ejecutable. Ejemplos de esto pueden incluir componentes compilados (applets de Java) y scripts del lado del cliente (Javascript).
- Interfaz uniforme: la interfaz simplifica y desacopla la arquitectura, que permite a cada parte evolucionar de forma independiente.

Las aplicaciones pueden caracterizarse conforme a los constreñimientos REST como "RESTful" en caso de que ningún servicio viole alguna de las restricciones requeridas. El cumplimiento de estas restricciones, y el ajuste al estilo de arquitectura REST, permite a cualquier tipo de sistema de hipermedia distribuido tener propiedades emergentes desables, tales como el rendimiento, la escalabilidad, la simplicidad, la modificabilidad, la visibilidad, la portabilidad y la fiabilidad.

REST está destinado a recordar una imagen de cómo se comporta una aplicación Web bien diseñada: presentado con una red de páginas Web (un estado de la máquina virtual), el usuario avanza a través de una aplicación mediante la selección de enlaces (transiciones de estado), resultando en la página siguiente (que representa el siguiente estado de la aplicación) que se transfiere al usuario y se procesa para su uso. REST fue descrito inicialmente en el contexto de HTTP, pero no está limitado a dicho protocolo. Las arquitecturas RESTful pueden estar basadas en otros protocolos de capa de aplicación si ya proporcionan un vocabulario rico y uniforme para aplicaciones basadas en la transferencia del estado representacional significativo.

Un servicio Web RESTful es una colección de recursos con cuatro aspectos definidos:

- La base URI para el servicio web (por ejemplo, http://ejemplo.com/recursos/).
- El tipo de datos soportado por Internet (XML, JSON, HTML, …).
- El conjunto de operaciones soportado por el servicio Web mediante métodos HTTP (GET, PUT, POST o DELETE).
- La API debe ser conducida por hipertexto.

3.2. COMPARACIÓN ENTRE REST Y SOAP

A diferencia de SOAP (Simple Object Access Protocol), REST utiliza casi siempre HTTP como método de comunicación y XML o JSON para intercambiar datos. Cada URL representa un objeto sobre el que se pueden utilizar los métodos POST, GET, PUT y DELETE. Utiliza el idioma de la web. Por su parte SOAP es toda una infraestructura basada en el lenguaje XML donde cada objeto puede tener métodos definidos por el programador con los parámetros que sean necesarios. El principal beneficio de SOAP reside en que está fuertemente acoplado, permitiendo la posibilidad de ser testado y depurado antes de poner en marcha la aplicación. En cambio, la escalabilidad, el acceso con escaso consumo de recursos (número de operaciones limitado) y el esquema de direccionamiento unificado son las principales ventajas de la aproximación basada en REST. Por lo tanto, queda patente que REST es ligero, con poca configuración, fácil de leer y no es necesario nada especial para implementarlo. SOAP, por su parte, es mucho más ambicioso, es fácil de consumir y tiene un tipado mucho más fuerte (WSLD).

Muchos diseñadores de Servicios Web han llegado a la conclusión de que SOAP es demasiado complicado y prefieren utilizar Servicios Web basados en REST para manejar grandes cantidades de datos.

3.3. INTERFAZ BASADA EN REST

Los identificadores uniformes de recursos (URI) son cadenas de caracteres que identifican inequívocamente un recurso (servicio, página, documento, dirección de correo electrónico, ...). Normalmente estos recursos son accesibles en una red o sistema. Los URI pueden ser localizadores uniformes de recursos (URL), nombres uniformes de recursos, o ambos. Un URI está formado por las siguientes partes:

- **Esquema**: Nombre que hace referencia a una especificación para asignar los identificadores aunque a veces también identifica el protocolo de acceso al recurso (http:, mailto:, ftp:, ...).
- **Autoridad**: Elemento jerárquico que identifica la autoridad de nombres (por ejemplo //api.openstreetmap.org).
- **Ruta**: Información jerarquizada que identifica al recurso en el esquema URI y la autoridad de nombres (por ejemplo /api/0.6/trackpoints).

- **Consulta**: Información con estructura no jerárquica (usualmente pares "clave=valor") que identifica al recurso en el ámbito del esquema URI y la autoridad de nombres. El comienzo de este componente se indica mediante el carácter '?'.
- **Fragmento**: Permite identificar una parte del recurso principal, o vista de una representación del mismo.

A continuación se muestra un ejemplo de URI. En este ejemplo se van a recuperar puntos GPS dentro de los límites establecidos mediante coordenadas longitud y latitud.

```
http://api.openstreetmap.org/api/0.6/trackpoints?bbox=0,51.5,0.25,51.75&page=0
```

Popularmente se llama URL a todas las direcciones Web, sin embargo se recomienda usar el término URI en su lugar ya que es un identificador más completo. La diferencia principal entre URL y URI reside en que un URI permite incluir en la dirección una subdirección determinada por el fragmento.

No es posible acceder directamente al recurso, para ello es necesario obtener una representación (documento HTML, XML, JSON, una imagen, ...). Por tanto, hay que decidir cuál va a ser la representación de cada recurso.

Teniendo en cuenta la URI del ejemplo anterior, el formato de representación será XML, ya que es el principal formato utilizado para intercambiar información. se recuperan los primeros 5000 puntos dentro de los límites de un rectángulo definido en la URI.

```
<?xml version="1.0" encoding="UTF-8"?>
<gpx version="1.0" creator="OpenStreetMap.org"
      xmlns="http://www.topografix.com/GPX/1/0">
      <trk>
            <name>2013_04_20_14_39_Sat.gpx</name>
            <desc>car Excel to Bluewater</desc>
            <url>http://api.openstreetmap.org/user/Tallguy/traces/1443525</url>
            <trkseg>
                  <trkpt lat="51.5090058" lon="0.020787">
                        <time>2013-04-20T13:39:24Z</time>
                  </trkpt>
                  ...
                  ...
                  ...
                  <trkpt lat="51.6076188" lon="0.0202829">
                        <time>2013-03-18T17:32:15Z</time>
                  </trkpt>
            </trkseg>
      </trk>
</gpx>
```

El acceso a los recursos es posible gracias a los URIs y los métodos soportados. Los métodos HTTP más importantes son PUT, GET, POST y DELETE. A menudo se les

compara con las operaciones asociadas a la tecnología de base de datos u operaciones CRUD (CREATE, READ, UPDATE, DELETE). El acceso se puede hacer de varias formas, bien sea recibiendo una representación del recurso (GET o HEAD), añadiendo o modificando una representación (POST o PUT) o bien eliminando algunas o todas las representaciones (DELETE). La siguiente tabla muestra el modo en que se utilizan los métodos HTTP para implementar un servicio Web.

HTTP	CRUD	Descripción
POST	CREATE	Crea un nuevo recurso.
GET	READ	Obtiene la representación de un recurso.
PUT	UPDATE	Actualiza un recurso.
DELETE	DELETE	Elimina un recurso.

Tabla 3 - Descripción de los métodos HTTP soportados.

El método GET es un método seguro, esto quiere decir que esta llamada no produce efectos laterales.

A diferencia de los servicios Web basados en SOAP, no hay un estándar oficial para los servicios Web RESTful. Esto se debe a que REST es un estilo arquitectónico, a diferencia de SOAP, que es un protocolo. A pesar de que REST no es un estándar, una implementación RESTful como la Web puede utilizar estándares como HTTP, URI, XML, etc.

3.4. FORMATOS DE RESPUESTA

Al hacer peticiones a un servidor web se tiene la necesidad de disponer de un formato de datos de calidad para las respuestas. Los tres formatos de datos más comúnes o populares en la actualidad y que se utilizan para transmitir datos desde un servidor web a un cliente son CSV, XML y JSON. Es interesante entender las diferencias entre cada formato así como saber cuando usarlo.

- **CSV** (Comma-Separated Values): Este formato de datos es una lista de elementos separados por comas. La principal ventaja de este formato reside en que es el más compacto de los tres, permitiendo así la posibilidad de reducción del ancho de banda. Este formato ocupa aproximadamente la mitad del tamaño de los formatos XML y JSON. Sin embargo, CSV carece de versatilidad y no apoya en realidad a las jerarquías de datos.
- **XML** (Extensible Markup Language): Fue creado para representar mejor a los formatos de datos con una estructura jerárquica. Este formato es muy ventajoso, ya que además de ser totalmente compatible con las estructuras de datos jerárquicos, es muy legible. El mayor inconveniente de este formato es su tamaño, ya que ocupa aproximadamente tres veces el tamaño del formato CSV. Esto se debe a que cada elemento de datos tiene una etiqueta de apertura y cierre como parámetro asociado.

- **JSON** (JavaScript Object Notation): Popularizado por Yahoo y Google entre los años 2005 y 2006, fue creado como alternativa al formato XML. JSON representa los datos jerárquicos con el uso de comas, llaves y soportes. Un ejemplo de JSON podría ser: {"name": "José", "edad":"35"},{"name": "Margarita", "edad":"34"},{"name": "Joaquín", "edad":"36"}. La ventaja de este formato reside en que, además de ser compatible con los datos jerárquicos, es sencillo, compacto y su tamaño es más pequeño que XML. Como su nombre indica, fue creado para analizar con más facilidad los datos en objetos Javascript nativos, por lo que es muy útil para aplicaciones web. En la actualidad, este formato se está popularizando cada vez más gracias a las nuevas APIs y los plugins que están apoyando a JSON y XML.

Se puede concluir que JSON es el mejor formato de intercambio de datos que existe en la actualidad. Es ligero, compacto y versátil. CSV sólo debe utilizarse si se están enviando grandes cantidades de datos o bien el ancho de banda es insuficiente. Hoy en día, se recomienda utilizar XML únicamente para anotaciones de los documentos y no como un formato de intercambio de datos. Existen otros formatos de respuesta específicos de los sistemas de información geográfica que poco a poco van siendo más populares y que también merece la pena describir en este apartado.

- **SHP**: Un fichero shape completo es un conjunto de ficheros que contienen datos vectoriales geoespaciales creados mediante aplicaciones software de ESRI. Este tipo de fichero es comúnmente utilizado para software de sistemas de información geográfica para definir ficheros de puntos, líneas y/o polígonos georreferenciados. Un fichero shape está comprimido en 4 ficheros de diferentes extensiones:

*.shp	Fichero que almacena las entidades geométricas.
*.shx	Fichero que almacenael los índices de las entidades geométricas.
*.dbf	Fichero de la base de datos que almacena el atributo de información de las entidades.
*.prj	Fichero que almacena la proyección utilizada (requerida en caso de utilizar una proyección distinta a WGS84).

- **KML** (Keyhole Markup Language): Estándar del OGC basado en el lenguaje XML para gestionar la visualización de los datos geográficos en navegadores como Google Earth, Google Maps, Google Maps para móviles, y la NASA WorldWind.
- **GeoRSS** (Geo Really Simple Syndication): Conjunto de estándares para representar información geográfica mediante el uso de capas. Está construido dentro de la familia de estándares RSS.

3.5. SERVICIOS WEB IGV BASADOS EN REST

En este apartado se describen con todo lujo de detalles las API REST de las infraestructuras más populares en la actualidad orientadas o dedicadas a la información geográfica voluntaria: GeoIQ (GeoCommons), Wikimapia, OpenStreetMap, Ushahidi, USA-NPN (Red Fenológica Nacional de los Estados Unidos de América), etc.






Gracias a ello, es posible mostrar en un mapa aquella información geográfica que puede ser de interés para el usuario mediante llamadas a servidores remotos. Esta información podrá ser visualizada sobre el mapa base en múltiples capas (layers).

3.5.1. GEOIQ

El API REST GeoIQ tiene métodos para realizar todas las acciones disponibles en la interfaz gráfica de usuario GeoIQ, aunque con más granularidad. Crea mapas de cualquier esquema de color que se desee, actualiza conjuntos de datos (datasets) en tiempo real y establece permisos de usuario.

CONJUNTO DE DATOS (DATASETS)

- Punto final: http://geocommons/datasets
- Descripción: El punto final de los datasets permite la creación, modificación, desacarga y borrado de datos en GeoIQ. Los métodos para modificar y borrar datasets requieren de autenticación. Para la lectura de datos dicha autenticación no es necesaria.
- Autenticación: No requerida para conjuntos de datos de GeoCommons. Para los datasets GeoIQ ésta dependerá de los permisos de los datos.
- Parámetros requeridos: ninguno.
- Parámetros opcionales:

Parámetro	Descripción	Ejemplo
include_attributes	Incluye atributos de los datos en el output: 0 ó 1 (valor por defecto: 0).	include_attributes=1
include_features	Incluye características en el output.	include_features=1
include_geometry	Incluye geometrías.	include_geometry=1
hex_geometry	Utiliza Hex EWKB para geometrías.	hex_geometry=1

Tabla 4 - Descripción de los parámetros opcionales para los datasets en GeoIQ.

El valor por defecto es 0, es decir, falso.

Ejemplo de llamada REST:

```
http://geocommons.com/datasets/7294.json?include_attributes=1&hex_geometry=1
```

El dataset 7294 corresponde a una capa que contiene las coordenadas geográficas (longitud y latitud) de todos los edificios capitolios que hay en Estados Unidos con sus atributos (estado, ciudad, nombre del edificio, dirección, ...).

En la figura 2 se muestra el resultado gráfico sobre el mapa correspondiente a la respuesta devuelta a la llamada anterior por parte del servidor de GeoCommons.

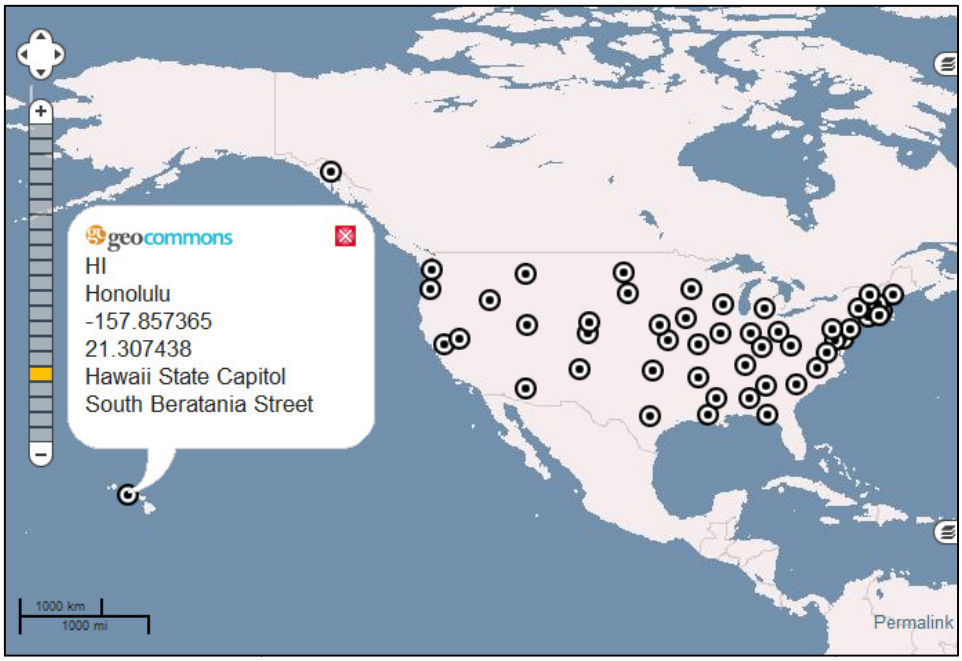


Figura 2 - Ejemplo de utilización de la función datasets
en la API REST de GeoIQ.

CARACTERÍSTICAS (FEATURES)

Los desarrolladores pueden obtener las características de los datasets, añadir nuevas características así como borrar otras existentes. En un futuro próximo habrá soporte para acceder y editar características individuales.

- Punto final: http://geocommons.com/datasets/:id/features.{format}
- Descripción: Las características de la API permiten al desarrollador recuperar todas las características o una porción de ellas en un dataset dado.
- Autenticación: Depende de los permisos. Se requiere para leer características filtradas, así como para crear y borrar características.
- Parámetros requeridos: ninguno.
- Parámetros opcionales:

Parámetro	Descripción	Ejemplo
limit	Número de características (features) a devolver (por defecto 30)	limit=2
order	Órden del atributo ascendente o descendente: 'ascending' o 'descending' (por defecto ascending)	order=descending

start	Valor índice por el que comenzar. En principio el valor inicial sería 1.	start=6
callback	Método de devolución de llamada para envolver la respuesta.	callback=filteredLayer
bbox	El orden de los límites coordenados para obtener las características contenidas dentro de la área descrita es Oeste/Sur/Este/Norte.	bbox=-79.5,20,-78,50
lon	Longitud (grados decimales).	lon=-77.9998
lat	Latitud (grados decimales).	lat=39.8282
units	Establece la unidad para el parámetro radio que por defecto es km (kilómetros). Las opciones posibles son: km,m,ft,miles,degrees	units=ft
radius	Define la distancia de búsqueda desde un polígono, línea, puntos o par longitud/latitud (unidad por defecto km).	radius=100
intersect	Especifica si los polígonos están contenidos completamente dentro del área de influencia o solamente una parte de ellos está contenida dentro del búffer: contained (valor por defecto) o full respectivamente.	intersect=contained
with_distance	Establece a “true” para obtener el cálculo de la distancia del centroide de la característica al borde del polígono devuelto con las características.	with_distance=1
filter	Filtra parámetros para atributos específicos dentro del conjunto de datos.	filter[dec_col][max]= 0.22
group_by	Array de columnas a agrupar “group by” en características.	group_by[]=state
aggregates	Array de valores de agregación. Incluye los atributos name y calc. Name es el atributo nombre y calc puede adoptar cualquiera de los siguientes valores: max, min, sum, average (máximo, mínimo, suma, media).	aggregates[][name]= timestamp&aggregates [][calc]=max
geojson	Devuelve la geometría en formato GeoJSON.	geojson=1
hex_geometry	Devuelve la geometría en formato HEX EWKB.	hex_geometry=1
encode	Devuelve la geometría en formato de polilínea codificado Google Maps.	encode=1

Tabla 5 - Descripción de los parámetros para las características (features) en GeoIQ.

Ejemplo de llamada REST:

```
http://geocommons.com/datasets/98696/features.json?bbox=-79.5,20,-78,50
```

MAPAS

- Punto final: http://geocommons.com/maps

- Descripción: El punto final permite la creación, estilo, descarga y eliminación o borrado de mapas. Los métodos de modificación y borrado de mapas requieren de una autenticación básica. Para la descarga de mapas depende de la configuración de la instancia GeoIQ para GeoCommons que no es requerida.
- Autenticación: La autenticación básica es requerida para mapas que no son compartidos de forma pública.
- Parámetros requeridos: ninguno.
- Parámetros opcionales: ninguno.

Ejemplo de llamada REST:

```
http://geocommons.com/maps/206016.kml
```

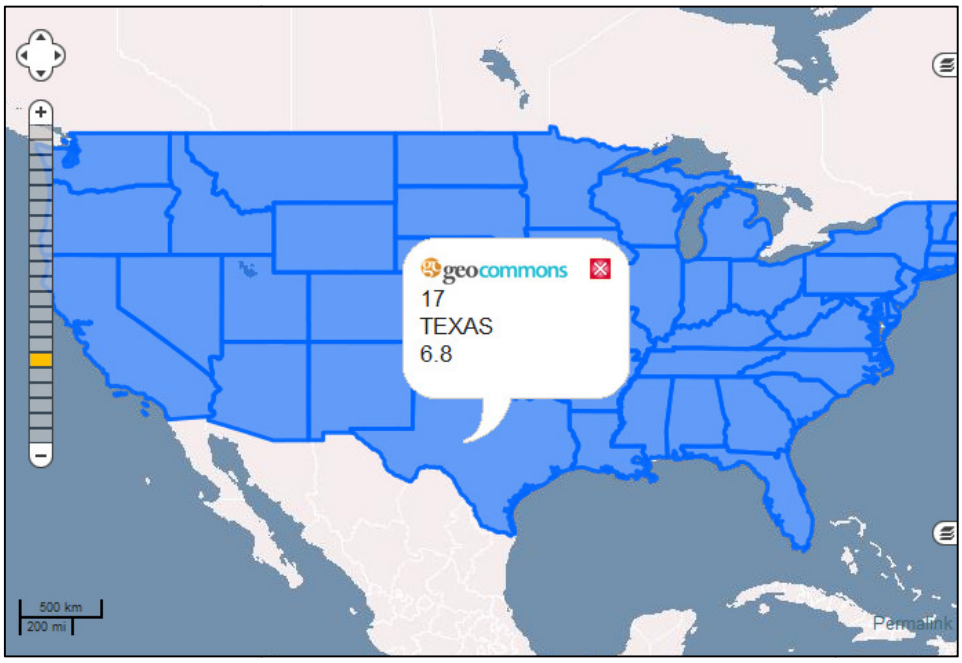


Figura 3 - Ejemplo de utilización de la función maps
en la API REST de GeoIQ.

En la figura 3 se muestra el resultado sobre el mapa de la llamada anterior en la cual se desea saber la tasa de desempleo que hay en el año 2012 en Estados Unidos. Si el usuario clica sobre cualquiera de dichos estados aparecerá el nombre del mismo junto con su tasa de desempleo (el estado de Texas tenía una tasa de desempleo del 6,8% en 2012).

BÚSQUEDA (SEARCH)

- Punto final: http://geocommons.com/search
- Descripción: GeoIQ proporciona una búsqueda común sobre la aplicación mediante el protocolo OpenSearch.
- Autenticación: No requerida para conjuntos de datos de GeoCommons. Para los datasets de GeoIQ la autenticación depende de los permisos de los datos.

- Métodos HTTP disponibles: GET
- Resumen URL:

Parámetro	URL	Formatos	MÉTODO
query	http://geocommons.com/search.{format}	atom,json,kml	GET

Tabla 6 - URL para la función de búsqueda en GeoIQ.

- Parámetros requeridos:

Parámetro	Descripción	Ejemplo
query	Información a buscar.	query=unemployment

Tabla 7 - Descripción de los parámetros requeridos para la función de búsqueda en GeoIQ.

- Parámetros opcionales:

Parámetro	Descripción	Ejemplo
bbox	Un cuadro que abarca la zona de búsqueda. El orden es Oeste, Sur, Este, Norte.	bbox=-79.5,20,-78,50
page	Establece la página de resultados a devolver.	page=3
limit	Número de resultados a devolver.	limit=15
order	Establece el tipo de ordenamiento del atributo (ascending / descending). El valor por defecto es ascending.	order=descending
sort	Nombre del atributo a ordenar.	sort=name

Tabla 8 - Descripción de los parámetros opcionales para la función de búsqueda en GeoIQ.

Un usuario puede buscar específicamente dentro de los campos prefijando las siguientes etiquetas antes de la petición (query): “title:”, “description:” o “tag:”.

Ejemplo de llamada REST:

http://geocommons.com/search.json?query=city&limit=3&page=2

El siguiente fragmento de texto contiene la respuesta a la petición anterior devuelta en formato json. En él se encuentran 3 resultados (limit=3) correspondientes a la página 2 de la petición efectuada 'city'.


```
{"entries": [{"icon_path": null, "feature_count": 262, "pk": 5628, "type": "Overlay",
"layer_size": 41138, "title": "Annual Number of Sunny Days for Cities, USA by City,
January 1st 2008", "short_classification": null, "analyzable": true, "data_type":
"Dataset", "is_temporal": null, "link": "http://geocommons.com/overlays/5628.json",
"name": "Annual Number of Sunny Days for Cities, USA by City, January 1st 2008",
"bbox": [13.4465, -170.277, 71.2901, 144.787], "description": "This dataset gives the
average number of sunny days in a year by city in the USA. The data were collected by
NOAA.", "permissions": {"download": true, "edit": false, "view": true}, "detail_link":
"http://geocommons.com/overlays/5628", "contributor": {"uri":
```

```
"http://geocommons.com/users/emily", "name": "emily"}, "published": "2008-01-
01T07:00:00-05:00", "created": "2008-10-23T19:18:32Z", "author": {"name": "",
"source": "National Oceanic and Atmospheric Administration (NOAA)"}, "id":
"overlay:5628", "geometry_types": ["point"], "tags": ["cities, sun, sunny, usa,
weather"], "unique_name": null}, {"icon_path": null, "feature_count": 50, "pk":
182460, "type": "Overlay", "layer_size": 8805, "title": "Afghanistan by City w seizure
sizes", "short_classification": null, "analyzable": true, "data_type": "Dataset",
"is_temporal": null, "link": "http://geocommons.com/overlays/182460.json", "name":
"Afghanistan by City w seizure sizes", "bbox": [30.1648, 62.086, 36.7106, 70.9794],
"description": "", "permissions": {"download": true, "edit": false, "view": true},
"detail_link": "http://geocommons.com/overlays/182460", "contributor": {"uri":
"http://geocommons.com/users/mfain19", "name": "mfain19"}, "published": "2013-06-
28T06:02:33-04:00", "created": "2011-11-30T01:31:37Z", "author": {"name": "",
"source": ""}, "id": "overlay:182460", "geometry_types": ["point"], "tags": [""],
"unique_name": null}, {"icon_path": null, "feature_count": 84, "pk": 32986, "type":
"Overlay", "layer_size": 174824, "title": "Drug Offense Locations 2.2010 and %
Population Below Poverty Level by Census Tract, 2000, Salt Lake City, Utah",
"short_classification": null, "analyzable": true, "data_type": "Dataset",
"is_temporal": null, "link": "http://geocommons.com/overlays/32986.json", "name":
"Drug Offense Locations 2.2010 and % Population Below Poverty Level by Census Tract,
2000, Salt Lake City, Utah", "bbox": [40.6416, -112.062, 40.8423, -111.795],
"description": "This dataset combines data from the 2000 US Census and Drug Offense
Location Data from crimereport.com. The location is census tracts around downtown Salt
Lake City. The crime reports are from 2.2010. The crime data was gathered by geocoding
the addresses of the offenses and doing a sum count per census tract.The census data
is the percentage of the population living in the census tract that is under the
poverty level.", "permissions": {"download": true, "edit": false, "view": true},
"detail_link": "http://geocommons.com/overlays/32986", "contributor": {"uri":
"http://geocommons.com/users/Burkey", "name": "Burkey"}, "published": "2010-03-
05T12:25:15-05:00", "created": "2010-03-05T17:15:42Z", "author": {"name": "",
"source": "CrimeReports \u0026 US Census"}, "id": "overlay:32986", "geometry_types":
["area"], "tags": ["census, crime, demographics, drugs, utah"], "unique_name": null}],
"next": "http://geocommons.com/search.json?limit=3\u0026page=3\u0026query=city",
"totalResults": 2882, "previous":
"http://geocommons.com/search.json?limit=3\u0026page=1\u0026query=city",
"itemsPerPage": 3}
```


Los nombres de las tres capas encontradas para esa petición son:

- Annual Number of Sunny Days for Cities, USA by City, January 1st 2008.
- Afghanistan by City w seizure sizes.
- Drug Offense Locations 2.2010 and % Population Below Poverty Level by Census Tract, 2000, Salt Lake City, Utah.

Si se desean obtener respuestas sobre capas más concretas, basta con añadir más información a la petición (por ejemplo: query=city+2013+park). Se recomienda usar el símbolo '+' o bien un espacio en blanco ' ' para separar cada palabra, nunca hacer la separación utilizando una coma ',' u otro símbolo.

3.5.2. WIKIMAPIA

La API de Wikimapia es un sistema que permite recibir datos de los mapas de Wikimapia. Es posible integrar estos datos geográficos en una aplicación externa o sitio web de forma gratuita para uso no comercial. Se proporciona acceso gratuito a la base

de datos de Wikimapia. La colección de esta base de datos se lleva a cabo mediante los contribuyentes o voluntarios, con el ánimo u objetivo de ayudar a la comunidad geográfica a usar estos datos no sólo en la Web de Wikimapia, sino también a través de otros sitios Web, incluyendo aplicaciones para móviles así como para sistemas de información geográfica. De este modo cualquiera puede usar esta información en cualquier parte.

Wikimapia está en contacto con los desarrolladores a través de un foro de discusión en la sección API, dando soporte con toda la documentación requerida y actualizaciones. Se espera que pronto sea publicada una guía de usuario de esta API, la cual aún se encuentra en estado de desarrollo (versión beta). En ella se incluirán más características como calles, utilización de gráficos, adición de información proveniente de dispositivos móviles a la base de datos de Wikimapia, etc.

Para implementar la API REST de Wikimapia hace falta obtener previamente una clave particular. Para ello es necesario registrarse o crear una nueva cuenta de usuario en la Web oficial de Wikimapia. A continuación se muestran las funciones de la API que están vigentes en la actualidad, aunque se espera que pronto se pongan más características así como una documentación más detallada a disposición de todos:

RECUADRO (BOX)

- Descripción: Devuelve el objeto contenido dentro de los límites de la zona requerida.
- Parámetros requeridos:

Parámetro	Descripción
http://api.wikimapia.org/?function=box	Esta es la base de la URL, que permite obtener información sobre los objetos contenidos dentro de los límites de un recuadro (box) dado.
&key=<key>	Clave de la API.
&bbox=<bbox>	Estas son las coordenadas del recuadro seleccionado, definidas en el siguiente orden: Oeste / Sur / Este / Norte. Puede usarse este parámetro o bien especificar las coordenadas separadamente como en el siguiente apartado.
&lon_min, &lat_min, &lon_max, &lat_max	Son las coordenadas del recuadro, donde lon significa longitud (λ) y lat hace referencia a la latitud (φ).
&x, &y, &z	Coordenadas 3D.

Tabla 9 - Descripción de los parámetros requeridos
para la función recuadro en Wikimapia.

- Parámetros adicionales:

Parámetro	Descripción
&options=<options>	Estas opciones se representan como valores separados por comas.
&disable=<disable>	Opción que deshabilita la salida de varios parámetros (location y/o polygon).
&page=<page>	Número de la página (por defecto 1).
&count=<count>	Variable que determina el número de resultados por página. El valor por defecto es 50.
&language=<language>	Lenguaje especificado en formato ISO 639-1.
&category=<category>	Código de las diferentes categorías en Wikimapia. Por ejemplo: category=17 - Shop, category=203 - School, etc. Se espera que en breve se publique una lista detallada en formato UTF-8 de las categorías disponibles en Wikimapia, tanto sus códigos respectivos como el nombre de la categoría.
&format=<format>	Formato de salida: xml (por defecto), kml, json, jsonp, binary.
&pack=<pack>	Es posible comprimir la información o datos de salida. Los valores disponibles son: none (por defecto) o gzip.

Tabla 10 - Descripción de los parámetros adicionales
para la función recuadro o box en Wikimapia.

OBJETO

- Descripción: Devuelve información sobre el objeto.
- Parámetros requeridos:

Parámetro	Descripción
http://api.wikimapia.org/?function=object	URL que permite obtener información del objeto.
&key=<key>	Clave de la API.
&id=<id>	Identificador del objeto del que se desea obtener información.

Tabla 11 - Descripción de los parámetros requeridos
para la función objeto en Wikimapia.

- Parámetros adicionales: Son los mismos parámetros adicionales que para la función recuadro anterior.

BÚSQUEDA (SEARCH)

- Descripción: Devuelve los resultados de la búsqueda de una petición o query dada.
- Parámetros requeridos:

Parámetro	Descripción
http://api.wikimapia.org/?function=search	URL que permite buscar información del objeto.
&key=<key>	Clave de la API.
&q=<query>	Petición a buscar en Wikimapia (UTF-8).
&lat=<lat>	Coordenada latitud (φ) del punto a buscar.
&lon=<lon>	Coordenada longitud (λ) del punto a buscar.

Tabla 12 - Descripción de los parámetros requeridos para la función búsqueda en Wikimapia.

- Parámetros adicionales: Son los mismos parámetros adicionales que para la función Box.

Ejemplo de llamada REST:

http://api.wikimapia.org/?function=search&q=hospital&count=100&page=1&lon=-0.3715776474975&lat=39.476898078791&key=xxxxxxxx-xxxxxxxx-xxxxxxxx-xxxxxxxx-xxxxxxxx-xxxxxxxx-xxxxxxxx-xxxxxxxx

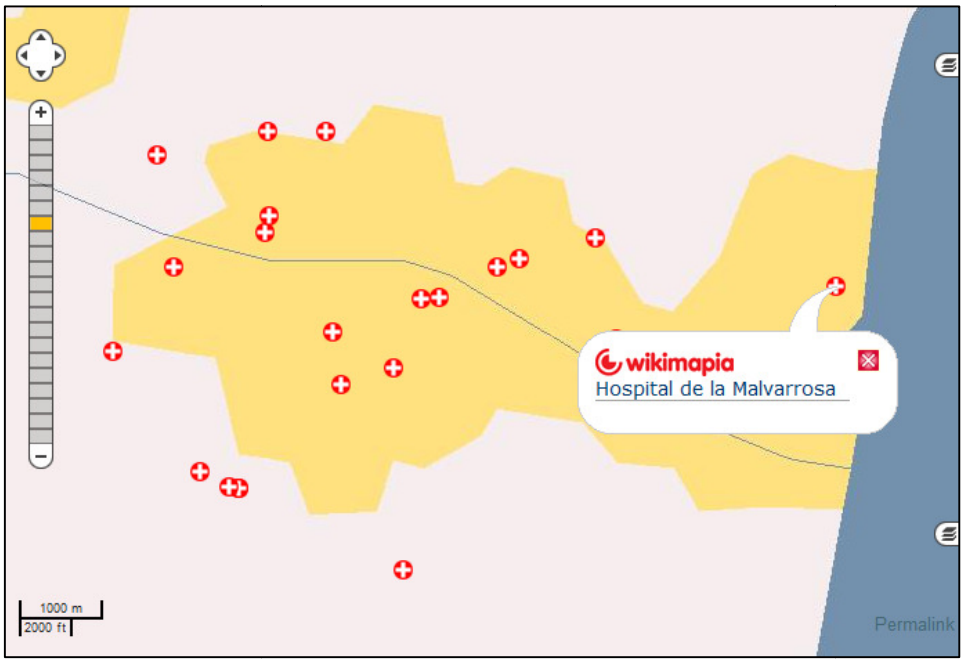


Figura 4 - Ejemplo de utilización de la función search en la API REST de Wikimapia.

En esta llamada URL se desea conocer la geolocalización de todos los hospitales existentes en torno a una zona determinada, mediante sus coordenadas longitud y latitud. La zona de estudio en este caso corresponde al área de la ciudad de Valencia, España. La figura anterior muestra el resultado plasmado sobre el mapa tras recibir la respuesta a la llamada URL efectuada al servidor de Wikimapia.

3.5.3. OPENSTREETMAP

Esta API está basada en las ideas de la API RESTful. La API es un componente servidor al cual se direccionan las peticiones REST. Las peticiones REST toman la forma de mensajes HTTP: GET, PUT, POST y DELETE. Cualquier carga útil está en formato XML, mediante el tipo MIME "texto/xml" y codificación de caracteres UTF-8, y puede ser comprimido sobre la capa HTTP si el cliente indica a través de la cabecera HTTP "Aceptar" que puede soportar mensajes comprimidos.

Las peticiones para modificar la base de datos se conceden mediante la autorización básica HTTP. Leer las peticiones no requiere autenticación (excepto detalles de usuario).

Todas las llamadas a la API que actualizan, crean o borran datos tienen que ser realizadas por un usuario autorizado que esté autenticado (nombre de usuario y contraseña). Existen llamadas API para crear, leer, actualizar y borrar los tres elementos básicos que conforman los datos del mapa para OpenStreetMap. Cada uno de ellos devuelven los datos para los elementos en formato XML. Cada modificación de uno o más elementos tiene que hacer referencia a un conjunto de cambios abierto.

IDENTIFICACIÓN DE USUARIOS

La versión previa de esta API (v0.5) devolvía solamente el nombre del usuario. El usuario puede actualizarlo en cualquier momento y no existe ningún histórico almacenado para mostrar los cambios en el nombre. Esto quiere decir que no había manera de identificar con fiabilidad que el usuario realizara un cambio concreto. La versión actual de esta API incluye el identificador de la cuenta del usuario además del nombre de usuario.

```
<node id="68" ... user="fred" uid="123"/>
```

Esto requiere aún que el usuario haya hecho sus ediciones públicas. El identificador de usuario para usuarios que previamente han hecho cambios anónimos no será visible. De acuerdo con una recomendación de la Junta de la Fundación de OpenStreetMap, ya no se admiten las ediciones anónimas.

VERSIONES DE LOS ELEMENTOS

Las llamadas a la API para elementos devuelven un atributo versión para cada nodo, camino o ruta y relación.

```
<node id="68" ... version="12"/>
```

Estas versiones se utilizan para bloqueos. Para actualizar la nueva versión de un objeto, el cliente necesitará proporcionar la versión del objeto que está siendo modificado. Si la versión proporcionada no es la misma que la versión del servidor actual, se devolverá un mensaje de error (HTTP status code 409: Conflict). Esto significa que cualquier cliente que actualiza información necesitará salvar el número de versión de los datos originales. Un elemento puede actualizarse múltiples veces durante el cambio de un conjunto de cambios y su número de versión se incrementa cada vez. De este modo pueden haber múltiples versiones de un simple elemento para un conjunto de cambios. Además, los clientes pueden ahora hacer peticiones sobre versiones específicas de un elemento.

Los números de versión empezarán siempre con el valor 1 y se incrementarán cada vez que un elemento sea modificado con una unidad.

FORMATO XML

Cada respuesta XML desde el servidor se incluye dentro de un elemento <osm> a menos que se especifique de otro modo.

```
<osm version="0.6" generator="OpenStreetMap server">
  ...
  ...
</osm>
```

Cada llamada a la API debe ser envuelta en un elemento <osm>, pero la versión y los atributos del generador pueden quedarse fuera.

CAPABILITIES

La llamada de esta API tiene por objeto proporcionar información sobre las apacidades y limitaciones de la API actual. Respuesta: Devuelve un documento XML (tipo de contenido text/xml).

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="OpenStreetMap server">
        <api>
                <version minimum="0.6" maximum="0.6"/>
                <area maximum="0.25"/>
                <tracepoints per_page="5000"/>
                <waynodes maximum="2000"/>
                <changesets maximum_elements="50000"/>
                <timeout seconds="300"/>
        </api>
</osm>
```

Nótese que los valores devueltos pueden cambiar en cualquier momento y que este documento XML solamente sirve como ejemplo.

- Las versiones mínima y máxima son las versiones de la API que serán aceptadas por el servidor.
- El área máxima, expresada en grados, que puede ser consultada mediante llamadas a la API.
- La traza de los puntos por página es el número máximo de puntos en una traza GPS simple.
- Puntos de ruta máximo es el número máximo de nodos que puede contener un camino o ruta.
- Máximo conjunto de cambios es el número máximo de nodos combinados, rutas y relaciones que pueden ser contenidas en un conjunto de cambios.

MAPA

Parámetro	URL	MÉTODO
query	http://api.openstreetmap.org/api/0.6/map	GET

Tabla 13 - URL para la función mapa en OpenStreetMap.

- Punto final: http://api.openstreetmap.org/api/0.6/map
- Métodos HTTP disponibles: GET
- Respuesta del comando:

    a. Todos los nodos incluidos dentro de los límites del recuadro dado así como las relaciones que les referencian.
    b. Todos los caminos que referencian al menos un nodo que está dentro de los límites del recuadro dado, las relaciones o rutas que les referencian así como aquellos nodos externos al área de influencia a la que los caminos pueden hacer referencia.
    c. Todas las relaciones que referencian a uno de los nodos, nos o relaciones incluidas.

- Parámetros requeridos:

Parámetro	Descripción
bbox	Estas son las coordenadas del recuadro seleccionado, definidas en el siguiente orden: Oeste/Sur/Este/Norte.

Tabla 14 - Descripción de los parámetros requeridos
para la función mapa en OpenStreetMap.

Ejemplo de llamada REST:

```
http://api.openstreetmap.org/api/0.6/map?
bbox=2.6891413413834, 39.54683483663, 2.6979037388424, 39.552803426204
```

En la figura 5 puede verse el resultado de la llamada REST anterior de forma gráfica implementada en el prototipo presentado en esta tésis. En la imagen se aprecian los nodos (representados en forma de marcadores a modo de chincheta) que indican puntos de interés; y líneas de trazo contínuo en color verde oscuro que representan caminos, perímetros de edificios, trazados eléctricos, transcurso de ríos, etc. Nótese que mientras este comando devuelve aquellas relaciones que hacen referencia a los nodos y rutas, a la inversa no es cierto, es decir, no se devuelven (necesariamente) todos los nodos y caminos o rutas referenciados por estas relaciones. Esto previene grandes conjuntos de resultados.

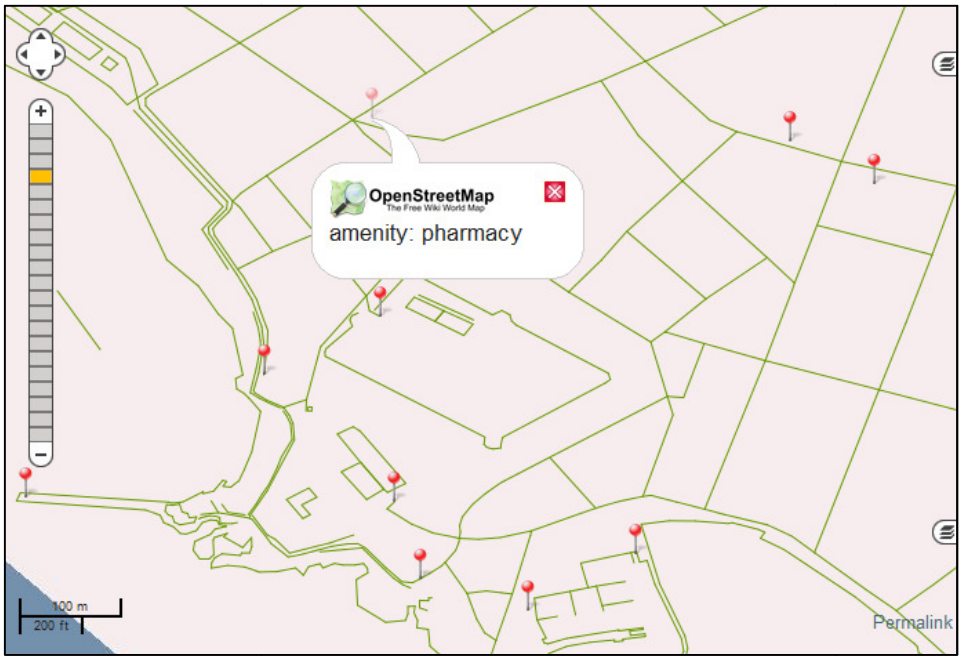


Figura 5 - Ejemplo de utilización de la función map
en la API REST de OpenStreetMap.

Imagine, por ejemplo, un caso donde hay una relación llamada "Spain" que referencia a cada nodo en España, y además se recuperan los nodos, caminos y relaciones dentro de los límites de un área considerada que cubre una pequeña porción de España. Mientras que el resultado incluiría los nodos, caminos y relaciones especificados por las reglas para este comando, incluyendo la relación 'Spain', no se incluiría cada nodo y ruta en España. Si se desea, los nodos y caminos referenciados por la relación 'Spain' podrían recuperarse por sus respectivos identificadores (IDs).

Nótese también que no se devuelven aquellas rutas que cruzan el área delimitada por el recuadro o caja en caso de que no tengan nodos dentro de la misma.

La API (versión 0.6) está actualmente disponible a través de la siguiente URL:

http://api.openstreetmap.org/

3.5.4. USA-NPN

En este apartado se describe cómo utilizar el servicio web de la Red Fenológica Nacional de los Estados Unidos (USANPN) para contribuir y extraer información de la base de datos fenológica nacional (NPDb). Esto incluye una serie de funciones que crean las construcciones necesarias para introducir observaciones – usuarios, estaciones, e individuales, así como un conjunto de funciones para recuperar datos utilizando diversas estructuras y filtros. Este servicio se basa en el protocolo SOAP, y soporta las versiones 1.1 y 1.2. El archivo WSDL del servicio se encuentra disponible en la siguiente URL:

https://www.usanpn.org/npn_portal/soap/wsdl/wsdl

De forma alternativa, las mismas funciones encontradas en el WSDL pueden ser utilizadas mediante una URL RESTful. Para la versión REST de las funciones, éstas pueden encontrarse en una ruta, y se llama a los parámetros GET después de que los atributos sean definidos para la función en el esquema WSDL.

Por ejemplo, la función getObservationComment tiene un correspondiente punto final WSDL de npn_portal/soap/observations/service y contiene un parámetro simple de entrada, un valor entero llamado observation_id. Por lo tanto, la llamada REST correspondiente sería:

```
https://www.usanpn.org/npn_portal/observations/getObservationComment.xml?
observation_id=X
```

Si se desea recibir la respuesta en formato JSON en lugar de en formato XML simplemente se debe cambiar la extensión en la URL con la que se está haciendo la petición:

```
https://www.usanpn.org/npn_portal/observations/getObservationComment.json?
observation_id=X
```

Las funciones disponibles en el WSDL se detallan a continuación. Cada entrada ofrece un ejemplo de cómo llamara a la función vía REST, y la información devuelta puede ser generalmente derivada del mensaje SOAP devuelto. Solamente han sido implementadas en el prototipo las funciones correspondientes al método HTTP de tipo GET.

Las funciones restantes (tipo POST) no han sido implementadas en el prototipo: createUser, createStation, createIndividual, enterObservation y enterObservationSet. Nótese que el acceso de estas funciones requiere que la comunicación se haga sobre el protocolo HTTPS y no sobre HTTP debido a cuestiones de seguridad.

GETOBSERVATIONS PORT

getAllObservationsForSpecies

Esta función devuelve todos los puntos de información positivos para cualquier número de especies dentro de un marco de tiempo especificado. Se toman los identificadores de las especies así como la fecha inicial y final como parámetros de entrada. Se devuelve una estructura de datos bastante compleja que incluye: un array que contiene todos los datos de la estación, un segundo array que contiene todos los datos de la fenofase, y un tercer array 4-D que se compone de todas las fechas en las que existen puntos de datos, todas las estaciones para las que hay datos en cada estación, todas las especies que hay en cada una de las estaciones y las fenofases para las cuales hay datos positivos de cada especie.

- Parámetros de entrada:

Campo	Tipo	Requerido	Cardinalidad	Descripción
species_id	int	SI	>=1	Identificadores de especies para los que se desea encontrar observaciones.
start_date	string	SI	1	Establece el intervalo de fechas para limitar los datos devueltos sobre las especies. El formato es YYYY-MM-DD.
end_date	string	SI	1	

Tabla 15 - Parámetros de entrada de la función getAllObservationsForSpecies en la API REST de USA-NPN.

- Parámetros de salida: station_list, station, station_id, station_name, longitude, latitude, phenophase_list, phenophase, phenophase_id, phenophase_name, color, observation_list, observation, date, stations, station_id, species_ids, species_id, phenophases y seq_num.

Ejemplo de llamada REST:

https://www.usanpn.org/npn_portal/observations/getAllObservationsForSpecies.xml?species_id[0]=52&species_id[1]=53&start_date=2008-01-01&end_date=2011-12-31

getObservationsForSpeciesIndividualAtLocation

Esta función se utiliza para devolver toda la información, positiva o negativa, para un individuo o grupo de individuos, que son de una especie común, en cualquier localización. Esta información se limita a un solo año. Como entrada, se toman tanto el identificador de la especie como el de un individuo, así como el año para el cual se desean obtener los datos y los identificadores de las diferentes estaciones consideradas.

Los identificadores de las estaciones se tienen en cuenta cuando la información que se pide pertenece a un grupo de individuos. Si ese individuo no está presente en esa estación, evidentemente no se devolverá información.

- Parámetros de entrada:

Campo	Tipo	Requerido	Cardinalidad	Descripción
species_id	int	NO	0 - 1	Identificador de la especie.
individual_id	int	NO	0 - 1	Identificador único del individual para el cual se desean encontrar observaciones.
year	string	NO	0 - 1	Parámetro opcional para limitar en un año concreto los datos devueltos. El formato es YYYY.
station_ids	int	SI	>=1	Identificadores de aquellas estaciones sobre las que buscar datos que conciernen a especies o a individuos concretos.

Tabla 16 - Parámetros de entrada de la función getObservationsForSpeciesIndividualAtLocation en la API REST de USA-NPN.

- Parámetros de salida: get_observations_for_species_individual_at_location_response_list, phenophase_id, phenophase_name, seq_num, color, dates, date, observations, id y observation.

Ejemplo de llamada REST:

https://www.usanpn.org/npn_portal/observations/getObservationsForSpeciesIndividualAtLocation.xml?year=2009&station_ids[0]=4881&station_ids[1]=4882&species_id=3

getObservationComment

Esta función devuelve el comentario asociado a una observación particular. Como entrada se toma el identificador de la observación asociado y como parámetro de salida el comentario. Los comentarios son valores opcionales de las observaciones, por lo que es frecuente que el resultado sea una cadena de texto vacía. Si el identificador de la observación no existe, el servicio devuelve nulo.

- Parámetros de entrada:

Campo	Tipo	Requerido	Cardinalidad	Descripción
observation_id	int	SI	1	Identificador de la observación para la que se desea recuperar el comentario.

Tabla 17 - Parámetros de entrada de la función getObservationComment en la API REST de USA-NPN.

- Parámetros de salida: observation_comment.

Ejemplo de llamada REST:

https://www.usanpn.org/npn_portal/observations/getObservationComment.xml?observation_id=1938

getObservationsForSpeciesByDay

Esta función devuelve una lista de especies que contienen todos los datos cuyas observaciones fueron hechas sobre esas especies, y un contador del número de las observaciones hechas en esa fecha. Como entrada se toman una fecha de inicio y una fecha final para especificar el intervalo de tiempo requerido, y una lista de identificadores de las especies. En respuesta a esta llamada se devuelve una lista de identificadores de especies así como una lista de todas las fechas y el número de observaciones correspondiente.

- Parámetros de entrada:

Campo	Tipo	Requerido	Cardinalidad	Descripción
species_id	int	SI	>=1	Identificadores de las especies.
start_date	string	SI	1	Establece el intervalo de fechas para limitar los datos devueltos sobre las especies. El formato es YYYY-MM-DD.
end_date	string	SI	1	

Tabla 18 - Parámetros de entrada de la función getObservationsForSpeciesByDay en la API REST de USA-NPN.

- Parámetros de salida: species, species_id, count_list, date y count.

Ejemplo de llamada REST:

https://www.usanpn.org/npn_portal/observations/getObservationsForSpeciesByDay.xml?start_date=2008-01-01&end_date=2011-12-31&species_id[0]=1

GETPERSON PORT

getPersonIDFromDrupalID

Esta función tomará un identificador de usuario drupal (gestor de contenidos) único para convertirlo en el identificador de persona. Este identificador es útil en otra función como getStationsForUser que requiere un identificador de persona. Si el usuario Drupal no tiene una cuenta asociada, el servicio devolverá el valor nulo.

- Parámetros de entrada:

Campo	Tipo	Requerido	Cardinalidad	Descripción
drupal_id	int	SI	1	Identificador drupal del usuario.

Tabla 19 - Parámetros de entrada de la función getPersonIDFromDrupalID en la API REST de USA-NPN.

- Parámetros de salida: person_id.

Ejemplo de llamada REST:

https://www.usanpn.org/npn_portal/person/getPersonIDFromDrupalID.xml?drupal_id=4326

GETIDIVIDUALS PORT

getIndividualsOfSpeciesAtStations

La función devuelve una lista de todos los individuos, que son miembros de una especie, en cualquier número de estaciones. Esta función dará información adicional perteneciente al número de observaciones hechas de cada individuo, aunque se limitará a los datos de un único año.

Como entrada se toma el identificador de la especie, el año por el que se cuentan el número de observaciones y una lista de todos los identificadores de las estaciones en las que buscar dichos individuos. Se devolverán los identificadores de todos los individuos encontrados, sus nombres y el número de observaciones hechas sobre cada una.

- Parámetros de entrada:

Campo	Tipo	Requerido	Cardinalidad	Descripción
species_id	int	SI	1	Identificador de la especie.
year	string	NO	0 - 1	Número de observaciones en un año concreto. El formato de este campo es YYYY.
station_ids	int	SI	>=1	Identificador de las estaciones consideradas.

Tabla 20 - Parámetros de entrada de la función getIndividualsOfSpeciesAtStations en la API REST de USA-NPN.

- Parámetros de salida: individual, individual_id, individual_name y number_observations.

Ejemplo de llamada REST:

https://www.usanpn.org/npn_portal/individuals/getIndividualsOfSpeciesAtStations.xml?station_ids[0]=60&station_ids[1]=259&species_id=35&year=2009

getIndividualsAtStations

Esta función devuelve todos los individuos en una serie de estaciones. Como entrada, se toma una serie de identificadores de estación. Como salida, se devuelve el identificador de cada individuo, el reino y el nombre.

- Parámetros de entrada:

Campo	Tipo	Requerido	Cardinalidad	Descripción
station_ids	int	SI	>=1	Este parámetro permite especificar cualquier número de estaciones (ids) para buscar por sus individuos.

Tabla 21 - Parámetros de entrada de la función getIndividualsAtStations en la API REST de USA-NPN.

- Parámetros de salida: individual, individual_id, individual_name, kingdom, species_id, active y seq_num.

Ejemplo de llamada REST:

https://www.usanpn.org/npn_portal/individuals/getIndividualsAtStations.xml?station_ids=507&station_ids=523

getIndividualById

Esta función devolverá un único individuo al introducir su identificador. Como entrada se toma el identificador del individuo y se devuelve, como salida, el nombre del individuo, el reino al que pertenece y el identificador de la especie.

- Parámetros de entrada:

Campo	Tipo	Requerido	Cardinalidad	Descripción
individual_id	int	SI	1	El identificador primario del individuo para el que se proporciona información adicional.

Tabla 22 - Parámetros de entrada de la función getIndividualById en la API REST de USA-NPN.

- Parámetros de salida: individual_name, kingdom y species_id.

Ejemplo de llamada REST:

```
https://www.usanpn.org/npn_portal/individuals/getIndividualById.xml?
individual_id=250
```

getShadeStatuses

Esta función devuelve una lista de valores válidos para el campo de estado 'sombra' útil para introducir plantas. No requiere parámetros de entrada.

- Parámetros de entrada: No tiene.
- Parámetros de salida: shade_status y status.

Ejemplo de llamada REST:

```
https://www.usanpn.org/npn_portal/individuals/getShadeStatuses.xml
```

GETSUBMISSION PORT

getLastSubmissionForPerson

Esta función se utiliza para descubrir la última fecha añadida a una observación en la base de datos. Se toman las credenciales válidas del usuario como entrada y se devuelve la última fecha, así como algún mensaje de respuesta y un código que informa de si se tuvo éxito o fracaso. Nótese que en este función se transmiten datos de seguridad del usuario, solamente puede ser accedido vía HTTPS y no vía HTTP, de lo contrario se devolvería un mensaje de error.

- Parámetros de entrada:

Campo	Tipo	Requerido	Cardinalidad	Descripción
user_id	int	NO	0 - 1	Clave primaria del usuario.
user_pw	string	NO	0 - 1	Contraseña del usuario.
access_token	string	NO	0 - 1	Token de acceso asociado con el usuario adquirido.
consumer_key	string	NO	0 - 1	Clave del consumidor del servicio solicitante. Debe utilizarse junto con el campo access_token.

Tabla 23 - Parámetros de entrada de la función getLastSubmissionForPerson en la API REST de USA-NPN.

- Parámetros de salida: date, response_messages y response_code.

Ejemplo de llamada REST:

```
https://www.usanpn.org/npn_portal/submissions/getLastSubmissionForPerson.xml?
user_id=2364&user_pw=XXXX
```

GETSPECIES PORT

getSpecies

Esta función se utiliza para obtener una lista de todas las especies en la base de datos. No se requieren parámetros de entrada, y la función devolverá el identificador de cada especie, el nombre común o vulgar, el género, el nombre y el número del Sistema de Información Taxonómico Integrado (ITIS).

- Parámetros de entrada: No tiene.
- Parámetros de salida: species, species_id, common_name, genus, species e itis_taxonomic_sn.

Ejemplo de llamada REST:

```
https://www.usanpn.org/npn_portal/species/getSpecies.xml
```

La siguiente imagen corresponde a la respuesta de la llamada anterior, devuelta por el servidor de la USA-NPN, tal cual se ve en el prototipo de la presente aplicación Web.

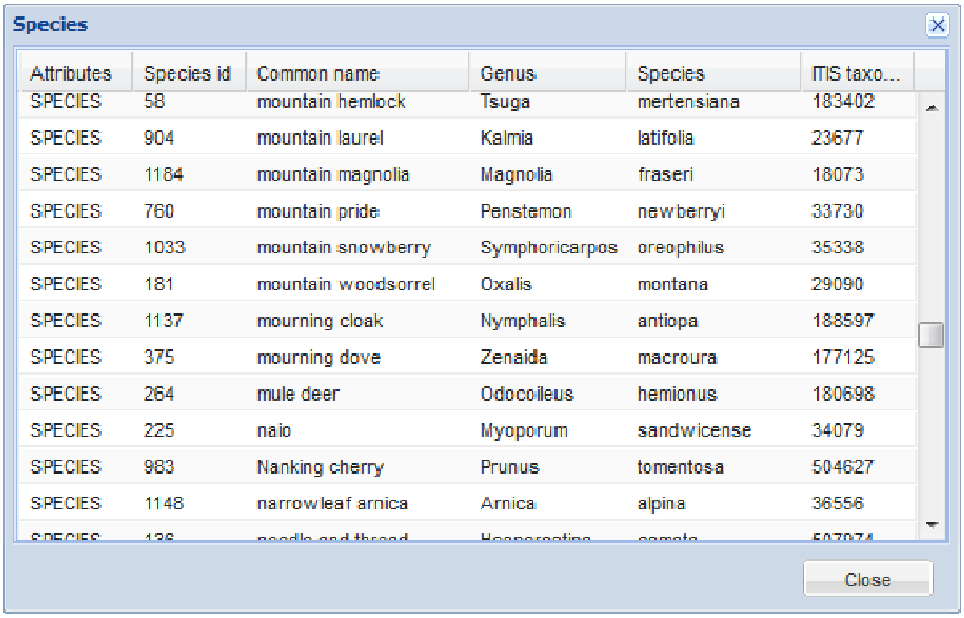


Figura 6 - Implementación de la función getSpecies
en la API REST de USA-NPN (visualización de todas las especies).

getSpeciesById

Esta función se utiliza para obtener información sobre una especie específica, basándose en el identificador de la misma. Como parámetro de entrada se introduce el identificador de la especie. Como resultado se obtiene el nombre común, el género, el nombre de la especie y el número ITIS.

- Parámetros de entrada:

Campo	Tipo	Requerido	Cardinalidad	Descripción
species_id	int	SI	1	Identificador de la especie.

Tabla 24 - Parámetros de entrada de la función getSpeciesById en la API REST de USA-NPN.

- Parámetros de salida: common_name, genus, species e itis_taxonomic_sn.

Ejemplo de llamada REST:

https://www.usanpn.org/npn_portal/species/getSpeciesById.xml?species_id=3

getSpeciesByState

Esta función devuelve una lista con todas las especies, filtradas por el estado y de forma opcional por el reino. Como entrada se toma la abreviación estándar del estado y opcionalmente el nombre del reino. El nombre del reino puede ser tanto 'Animalia' para animales como 'Plantae' para especies de tipo plantas. La salida devolverá el identificador de cada especie, el nombre común, el género, el nombre de la especie y el número ITIS.

- Parámetros de entrada:

Campo	Tipo	Requerido	Cardinalidad	Descripción
state	String	SI	1	Estado al que pertenece la especie. El formato debe cumplir el de la abreviación postal de EU estándar.
kingdom	string	NO	0 - 1	Campo opcional para filtrar las especies según el reino al que pertenezcan.

Tabla 25 - Parámetros de entrada de la función getSpeciesByState en la API REST de USA-NPN.

- Parámetros de salida: species, species_id, common_name y genus.

Ejemplo de llamada REST:

https://www.usanpn.org/npn_portal/species/getSpeciesByState.xml?state=HI&kingdom=Plantae

getSpeciesUpdateDate

Esta función devuelve la última fecha en la que la información de la tabla especie fue modificada. Esto permitiría al consumidor del servicio almacenar en caché localmente la información correspondiente a la especie, basándose en esta función, con una carga mucho más pequeña, que le informe sobre los posibles cambios. Esta función no requiere de parámetros de entrada y devuelve sólo un valor correspondiente a la fecha actualizada.

- Parámetros de entrada: No tiene.
- Parámetros de salida: update_date.

Ejemplo de llamada REST:

https://www.usanpn.org/npn_portal/species/getSpeciesUpdateDate.xml

getPlantTypes

Esta función devuelve una lista de todas las especies de tipo planta en la base de datos. Un tipo de especie podría ser 'conifer' para las plantas coníferas o 'deciduous' para las plantas de hoja caduca, o algún otro grupo de especie a la que pertenece la planta. Esta función no requiere parámetros de entrada, y devuelve una lista con los nombres de todos los tipos de plantas, sus identificadores y la cuenta del número de especies que pertenecen a cada grupo.

- Parámetros de entrada: No tiene.
- Parámetros de salida: type, species_type, species_type_id y species_count.

Ejemplo de llamada REST:

https://www.usanpn.org/npn_portal/species/getPlantTypes.xml

getAnimalTypes

Esta función devuelve una lista de todas las especies de tipo animal existentes en la base de datos. Un tipo de especie podría ser 'mammals' (mamífero) o 'insects' (insectos) o algún otro grupo de especie al que pertenece el animal.

Esta función no requiere parámetros de entrada, y devuelve una lista de los nombres de todos los tipos de animales, sus identificadores y la cuenta del número de especies que hay en cada grupo.

- Parámetros de entrada: No tiene.
- Parámetros de salida: type, species_type, species_type_id y species_count.

Ejemplo de llamada REST:

https://www.usanpn.org/npn_portal/species/getAnimalTypes.xml

La respuesta a la petición anterior puede verse en el siguiente texto devuelto en formato XML. En él se ven todas las especies animales existentes en la API de la Red Fenológica.

```
<types>
<type species_type="Amphibian" species_type_id="14" species_count="38"/>
<type species_type="Bird" species_type_id="15" species_count="96"/>
<type species_type="Fish" species_type_id="16" species_count="22"/>
<type species_type="Insect - Butterfly/Moth" species_type_id="17" species_count="23"/>
<type species_type="Mammal" species_type_id="18" species_count="25"/>
<type species_type="Dragonfly/Damselfly" species_type_id="19" species_count="1"/>
<type species_type="Reptile" species_type_id="20" species_count="22"/>
<type species_type="Insect-Tiger Beetle" species_type_id="21" species_count="2"/>
<type species_type="Insect-Bee" species_type_id="22" species_count="5"/>
<type species_type="Insect-Aquatic" species_type_id="23" species_count="13"/>
<type species_type="Insect-Grasshopper" species_type_id="24" species_count="1"/>
<type species_type="Insect - All" species_type_id="25" species_count="44"/>
<type species_type="Reptile and Amphibian" species_type_id="27" species_count="60"/>
<type species_type="Invasive Animals" species_type_id="28" species_count="6"/>
</types>
```

getSpeciesFilter

Esta función permite filtrar la lista de especies mediante un número de diferentes variables, devolviendo información sobre cada especie, incluyendo el número de observaciones hechas entre especies de todas las localizaciones durante un año.

Como parámetros de entrada, las especies pueden filtrarse combinando el identificador de la red (singular), el identificador del grupo (múltiple), el identificador de la estación (múltiple); y también opcionalmente un año para limitar el número de observaciones hechas. Si se omite el parámetro año, los resultados devueltos corresponderían a todo el tiempo. Si se excluyen todos los parámetros entonces esta función devolverá datos de todas las especies.

La función devuelve una lista de nombres de las especies, identificadores y el número de observaciones hechas de cada una.

- Parámetros de entrada:

Campo	Tipo	Requerido	Cardinalidad	Descripción
network_id	int	NO	0 - 1	Clave primaria de la red.
observation_year	string	NO	0 - 1	Campo opcional para delimitar el número de observaciones de cada especie asociadas a un año específico.
group_ids	int	NO	>=0	Una o más claves primarias asociadas a un tipo de especie.
station_ids	int	NO	>=0	Una o más claves primarias asociadas a una estación.

Tabla 26 - Parámetros de entrada de la función getSpeciesFilter en la API REST de USA-NPN.

- Parámetros de salida: species, species_id, common_name, genus, species y number_observations.

Ejemplo de llamada REST:

```
https://www.usanpn.org/npn_portal/species/getSpeciesFilter.xml?
group_ids[0]=9&group_ids[1]=3&observation_year=2010
```

getSpeciesByITIS

Esta función devuelve información sobre la especie asociada al número ITIS introducido. Como valor de entrada se considera el número ITIS único de cada especie. El resultado devuelve el identificador de cada especie, e nombre común, el género así como el nombre de la especie.

- Parámetros de entrada:

Campo	Tipo	Requerido	Cardinalidad	Descripción
itis_sn	int	SI	1	Número ITIS de las especies.

Tabla 27 - Parámetros de entrada de la función getSpeciesByITIS en la API REST de USA-NPN.

- Parámetros de salida: species_id, common_name, genus, species e itis_taxonomic_sn.

Ejemplo de llamada REST:

```
https://www.usanpn.org/npn_portal/species/getSpeciesByItis.xml?itis_sn=27806
```

getSpeciesByScientificName

Esta función devuelve información sobre las especies asociadas a un determinado género y nombre. Como parámetros de entrada se consideran el género y el nombre de la especie. Ambos parámetros son obligatorios. Los parámetros de salida devuelven el identificador de cada especie, el nombre común y el número ITIS.

- Parámetros de entrada:

Campo	Tipo	Requerido	Cardinalidad	Descripción
genus	string	SI	1	Género de la especie.
species	string	SI	1	Nombre de la especie.

Tabla 28 - Parámetros de entrada de la función getSpeciesByScientificName en la API REST de USA-NPN.

- Parámetros de salida: species_id, common_name e itis_taxonomic_sn.

Ejemplo de llamada REST:

https://www.usanpn.org/npn_portal/species/getSpeciesByScientificName.xml?genus=Clintonia&species=borealis

getSpeciesByCommonName

Esta función devuelve información sobre las especies asociadas a un determinado nombre común. El parámetro de entrada es el nombre común de la especie. Los parámetros de salida devuelven el ID de cada especie, el género, el nombre de la especie y el número ITIS. Se buscarán tanto nombres comunes primarios como secundarios.

- Parámetros de entrada:

Campo	Tipo	Requerido	Cardinalidad	Descripción
common_name	string	SI	1	Nombre común de la especie.

Tabla 29 - Parámetros de entrada de la función getSpeciesByCommonName en la API REST de USA-NPN.

- Parámetros de salida: species_id, common_name, itis_taxonomic_sn, genus y species.

Ejemplo de llamada REST:

https://www.usanpn.org/npn_portal/species/getSpeciesByCommonName.xml?common_name=thickleaved%20wild%20strawberry

GETSTATIONS PORT

getAllStations

Esta función devuelve una lista completa de todas las estaciones existentes en un estado. Como parámetro de entrada se considera la abreviación del código del estado al que pertenecen las estaciones que se requieren para una determinada consulta. Se devolverá una lista de todas las estaciones incluidas en la consulta, con sus respectivas coordenadas longitud y latitud, nombres e identificadores.

- Parámetros de entrada:

Campo	Tipo	Requerido	Cardinalidad	Descripción
state_code	string	SI	0 - 1	Código del estado.

Tabla 30 - Parámetros de entrada de la función getAllStations en la API REST de USA-NPN.

- Parámetros de salida: station, latitude, longitude, station_name, station_id, network_id y is_public.

Ejemplo de llamada REST:

https://www.usanpn.org/npn_portal/stations/getAllStations.xml?state_code=AL

En este caso se muestran sobre el mapa base las localizaciones de todas las estaciones fenológicas existentes en el estado AL (Alabama, Estados Unidos).

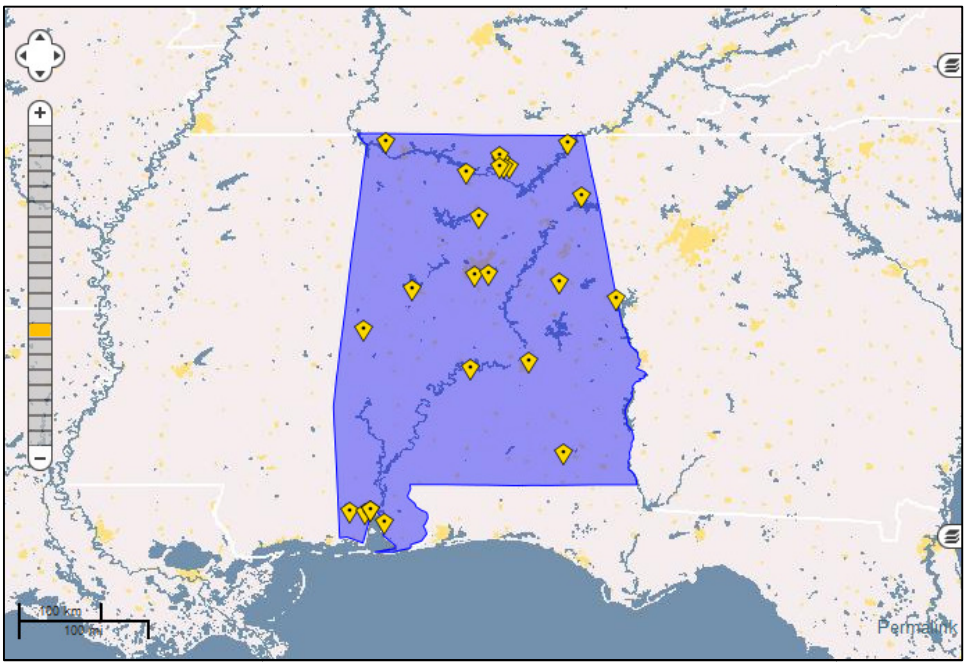


Figura 7 - Ejemplo de utilización de la función getAllStations en la API REST de USA-NPN (visualización de las estaciones).

Al clicar con el puntero del ratón sobre cualquiera de los iconos mostrados sobre el mapa se visualiza la información correspondiente a la estación fenológica en cuestión (nombre, identificador, coordenadas, ...).

Figura 8 - Ejemplo de utilización de la función getAllStations
en la API REST de USA-NPN (acceso a la información de una estación).

getStationsForUser

Esta función devolverá una lista con todas las estaciones que pertenecen a un usuario en concreto. Como parámetros de entrada se considera el identificador único de la persona. Los parámetros que se devuelven son la lista de estaciones con sus coordenadas latitud y longitud, nombres e identificadores.

- Parámetros de entrada:

Campo	Tipo	Requerido	Cardinalidad	Descripción
person_id	int	NO	0 - 1	Persona a la que pertenecen las estaciones devueltas.
access_token	string	NO	0 - 1	Token de acceso perteneciente al usuario al que pertenecen las estaciones devueltas.
consumer_key	string	NO	0 - 1	Clave del consumidor que solicitó el usuario desde el correspondiente token de acceso.

Tabla 31 - Parámetros de entrada de la función getStationsForUser
en la API REST de USA-NPN.

- Parámetros de salida: station, latitude, longitude, station_name, station_id, network_id e is_public.

Ejemplo de llamada REST:

```
https://www.usanpn.org/npn_portal/stations/getStationsForUser.xml?
person_id=2364
```

getStationsWithSpecies

Esta función devuelve una lista con todas las estaciones que tienen un individuo que es miembro de un conjunto de especies. Como entrada se consideran los identificadores de las especies y se devuelven las correspondientes estaciones, sus coordenadas latitud y longitud, nombres e identificadores.

- Parámetros de entrada:

Campo	Tipo	Requerido	Cardinalidad	Descripción
species_id	int	SI	>=1	Listado de identificadores de las especies.

Tabla 32 - Parámetros de entrada de la función getStationsWithSpecies en la API REST de USA-NPN.

- Parámetros de salida: station, latitude, longitude, station_name y station_id.

Ejemplo de llamada REST:

```
https://www.usanpn.org/npn_portal/stations/getStationsWithSpecies.xml?
species_id=3
```

getStationCountByState

Esta función devuelve una lista de todos los códigos de los estados en Estados Unidos así como la correspondiente cuenta de todas las estaciones localizadas en cada estado. Esta función no requiere parámetros de entrada, y devuelve una lista de cada estado y el correspondiente número de estaciones.

- Parámetros de entrada: No tiene.
- Parámetros de salida: state_count, state y number_stations.

Ejemplo de llamada REST:

```
https://www.usanpn.org/npn_portal/stations/getStationCountByState.xml
```

getStationsById

Esta función devuelve datos sobre cualquier número de estaciones basándose en sus identificadores. Como parámetros de entrada se considera los identificadores de las estaciones que se deseen. Los parámetros de salida devueltos tras la llamada son las estaciones consideradas, sus coordenadas latitud y longitud así como sus nombres e identificadores.

- Parámetros de entrada:

Campo	Tipo	Requerido	Cardinalidad	Descripción
station_id	int	SI	>=1	Listado de los identificadores de las estaciones cuya información es requerida.

Tabla 33 - Parámetros de entrada de la función getStationsById en la API REST de USA-NPN.

- Parámetros de salida: station, latitude, longitude, station_name y station_id.

Ejemplo de llamada REST:

https://www.usanpn.org/npn_portal/stations/getStationById.xml?station_id[0]=5122

getStates

Esta función devolverá la lista de todas las abreviaciones correspondientes a los estados de Estados Unidos, territorios estadounidenses, províncias canadienses y estados mexicanos, para utilizarse en el programa de la red fenológica nacional. Esta función no requiere parámetros de entrada y devuelve el código del estado, el nombre del estado y el identificador del estado para cada tipo de entidad.

- Parámetros de entrada: No tiene.
- Parámetros de salida: state, state_code, state_name y state_id.

Ejemplo de llamada REST:

https://www.usanpn.org/npn_portal/stations/getStates.xml

getStationsForNetwork

Esta función devuelve información sobre aquellas estaciones que pertenecen a una red particular. Como parámetro de entrada se toma el identificador de la red y se devuelven

las estaciones correspondientes, sus nombres, coordenadas geográficas, identificadores y un valor booleano que informa sobre si la estación en cuestión es pública (1) o no (0).

- Parámetros de entrada:

Campo	Tipo	Requerido	Cardinalidad	Descripción
network_id	int	SI	1	Clave primaria de la red para la cual se devuelve información sobre sus estaciones.

Tabla 34 - Parámetros de entrada de la función getStationsForNetwork en la API REST de USA-NPN.

- Parámetros de salida: station, latitude, longitude, station_name, station_id y is_public.

Ejemplo de llamada REST:

https://www.usanpn.org/npn_portal/stations/getStationsForNetwork.xml?network_id=69

getPublicStations

Esta función devuelve información sobre todas las estaciones marcadas como públicas. No se requieren parámetros de entrada. Como salida se proporcionan las coordenadas latitud y longitud de las diferentes estaciones, así como sus nombres e identificadores.

- Parámetros de entrada: No tiene.
- Parámetros de salida: station, latitude, longitude, station_name y station_id.

Ejemplo de llamada REST:

https://www.usanpn.org/npn_portal/stations/getPublicStations.xml

getPublicStationsForUser

Esta función devuelve datos sobre estaciones públicas asociadas a un usuario en particular. Como parámetro de entrada se toma el identificador de la persona. De forma alternativa, pueden introducirse el token de acceso y la clave del consumidor.

La respuesta devuelve las coordenadas longitud y latitud de las estaciones, nombres, identificadores, si la estación es pública o no y el identificador de la red a la que pertenecen las estaciones.

- Parámetros de entrada:

Campo	Tipo	Requerido	Cardinalidad	Descripción
person_id	int	SI	0 - 1	Persona a la que pertenecen las estaciones devueltas.
access_token	string	NO	0 - 1	Token de acceso que corresponde al usuario al que pertenecen las estaciones devueltas.
consumer_key	string	NO	0 - 1	Clave del consumidor que solicitó el usuario desde el su token de acceso.

Tabla 35 - Parámetros de entrada de la función getPublicStationsForUser en la API REST de USA-NPN.

- Parámetros de salida: station, latitude, longitude, station_name, station_id, is_public y network_id.

Ejemplo de llamada REST:

```
https://www.usanpn.org/npn_portal/stations/getPublicStationsForUser.xml?
person_id=2364
```

GETNETWORKS PORT

getPartnerNetworks

Esta función devuelve una lista con todas las redes, es decir, con todas las organizaciones, con las que la red fenológica está asociada. No se requieren parámetros de entrada y se devuelve una lista con los nombres e identificadores de cada una de las redes.

- Parámetros de entrada: No tiene.
- Parámetros de salida: network, network_id y network_name.

Ejemplo de llamada REST:

```
https://www.usanpn.org/npn_portal/networks/getPartnerNetworks.xml
```

getNetworksForUser

Esta función devuelve datos sobre redes a las que está asociado un usuario. Como parámetros de entrada se toma el identificador de usuario. De forma alternativa, pueden

introducirse el token de acceso al que pertenecen las estaciones devueltas y la clave del consumidor que solicitó el usuario desde el correspondiente token de acceso. La respuesta devuelve los nombres y los identificadores de las redes.

- Parámetros de entrada:

Campo	Tipo	Requerido	Cardinalidad	Descripción
person_id	int	NO	0 - 1	Persona a la que pertenecen las estaciones devueltas.
access_token	string	NO	0 - 1	Token de acceso.
consumer_key	string	NO	0 - 1	Clave del consumidor.

Tabla 36 - Parámetros de entrada de la función getNetworksForUser en la API REST de USA-NPN.

- Parámetros de salida: network, network_id y network_name.

Ejemplo de llamada REST:

```
https://www.usanpn.org/npn_portal/networks/getNetworksForUser.xml?
person_id=2364
```

getUserNetworkStatus

Esta función devuelve el estado de un usuario dentro de una red dada a la que pertenece. Esta información se utiliza para saber qué privilegios tiene el usuario dentro de la red. Como parámetros de entrada se toman el identificador del usuario así como el identificador de la red con la que probar los privilegios del usuario. Opcionalmente pueden introducirse el token de acceso (que corresponde al usuario al que pertenecen las estaciones) y la clave del consumidor. Como parámetro de salida se devuelve el estado del usuario, que es un valor entero:

- Parámetros de entrada:

Campo	Tipo	Requerido	Cardinalidad	Descripción
person_id	int	SI	0 - 1	Persona a la que pertenecen las estaciones devueltas.
access_token	string	NO	0 - 1	Token de acceso.
consumer_key	string	NO	0 - 1	Clave del consumidor que solicitó el usuario.
network_id	int	SI	1	Clave primaria de la red para la que probar los privilegios del usuario.

Tabla 37 - Parámetros de entrada de la función getUserNetworkStatus en la API REST de USA-NPN.

- Parámetros de salida: status.

Estado	Descripción
null	El usuario no pertenece a la red.
1	El usuario es el administrador de una red. Este usuario puede crear nuevas estaciones dentro de la red así como crear y borrar plantas y animales en estaciones dentro de la red.
2	El usuario es un usuario de una red. Este usuario puede crear y borrar plantas y animales en estaciones dentro de la red, pero no puede crear nuevas estaciones dentro de la misma.

Tabla 38 - Descripción del parámetro de salida de la función getUserNetworkStatus en la API REST de USA-NPN.

Ejemplo de llamada REST:

https://www.usanpn.org/npn_portal/networks/getUserNetworkStatus.xml?person_id=2364&network_id=62

GETPHENOPHASES PORT

getPhenophases

Esta función devuelve una lista con todas las fenofases utilizadas en la base de datos. No se requieren parámetros de entrada y se devuelve una lista con el identificador, nombre, categoría y color asociado a cada fenofase.

- Parámetros de entrada: No tiene.
- Parámetros de salida: phenophase, phenophase_id, phenophase_name, phenophase_category y color.

Ejemplo de llamada REST:

https://www.usanpn.org/npn_portal/phenophases/getPhenophases.xml

getPhenophasesForSpecies

Esta función devuelve una lista de todas las fenofases utilizadas por un conjunto de especies. Estas fenofases son susceptibles de cambiar con el tiempo, de manera que una fecha puede también facilitarse para proporcionar un intervalo de tiempo específico, o puede devolverse un catálogo completo de fenofases. Como parámetros de entrada se toman una lista de identificadores de especies y la fecha. La fecha especifica especifica qué fenofases deberían devolverse. Finalmente, el último parámetro de entrada 'return_all' es un parámetro booleano que si está establecido como cierto, entonces el

parámetro fecha será completamente ignorado, y se devolverán todas las fenofases de cada especie. Esta función devuelve un array bidimensional. El primer nivel del array contiene el identificador y el nombre de cada especie. El segundo nivel contiene toda la información asociada de las fenofases con cada especie, incluyendo sus identificadores, nombre, categoría, definición, definición adicional, color, número de secuencia y datos de abundancia.

Nótese que la definición adicional será diferente para especies diferentes, es decir, dos especies pueden tener la misma fenofase pero diferente definición adicional. Lo mismo es también cierto para el número de secuencia y los datos de abundancia.

- Parámetros de entrada:

Campo	Tipo	Requerido	Cardinalidad	Descripción
species_id	int	SI	>=1	Identificador(es) de las especies para las que se devuelve información de la fenofase.
date	string	SI	1	Fecha por la que encontrar la información de la fenofase aplicable para cada especie.
return_all	boolean	NO	0 - 1	Booleano que indica si debe o no devolver todas las fenofases para cada especie, independientemente de los datos suministrados. El valor por defecto es falso.

Tabla 39 - Parámetros de entrada de la función getPhenophasesForSpecies en la API REST de USA-NPN.

- Parámetros de salida: species_list, species_id, species_name, phenophases, phenophase_id, phenophase_name, phenophase_category, phenophase_definition, phenophase_additional_definition, color, seq_num, abundance_category y raw_abundance.

Ejemplo de llamada REST:

https://www.usanpn.org/npn_portal/phenophases/getPhenophasesForSpecies.xml?species_id[0]=7&species_id[1]=3&date=2011-02-25&return_all=false

getPhenophasesUpdateDate

Esta función devuelve la fecha en la que se modificó por última vez la información en tabla fenofase. Esto permitiría que alguien que consume el servicio pueda almacenar en caché localmente información sobre fenofases, basándose en esta función, con una

carga mucho más pequeña, que le informe sobre los posibles cambios. Esta función no requiere parámetros de entrada y devuelve sólo el valor de fecha de actualización.

- Parámetros de entrada: No tiene.
- Parámetros de salida: update_date.

Ejemplo de llamada REST:

```
https://www.usanpn.org/npn_portal/phenophases/getPhenophasesUpdateDate.xml
```

getAbundanceCategory

Esta función devuelve una descripción de una categoría de abundancia, junto con una descripción completa de todos los valores válidos que se pueden proporcionar para dicha categoría. Esta función requiere, como parámetro de entrada, el identificador de la categoría. Se devuelve el nombre de la categoría, la descripción y un array de cada valor válido y sus correspondientes identificadores, nombres y descripciones.

- Parámetros de entrada:

Campo	Tipo	Requerido	Cardinalidad	Descripción
category_id	int	SI	1	Identificador de la categoría de la que recuperar la información.

Tabla 40 - Parámetros de entrada de la función getAbundanceCategory en la API REST de USA-NPN.

- Parámetros de salida: category_name, category_description, category_values, value_id, value_name y value_description.

Ejemplo de llamada REST:

```
https://www.usanpn.org/npn_portal/phenophases/getAbundanceCategory.xml?
category_id=15
```

getAbundanceCategories

Esta función devuelve una descripción de todas las categorías de abundancia, junto con una descripción completa de todos los valores válidos que se pueden proporcionar para cada categoría. Esta función no requiere de ningún parámetro de entrada.

La llamada devuelve un array de cada categoría que contiene: el nombre de la categoría, la descripción y una lista de cada uno de los valores válidos para esa categoría, así como sus identificaciones posteriores, nombres y descripciones.

- Parámetros de entrada: No tiene.
- Parámetros de salida: category, category_name, category_description, category_values, value_id, value_name y value_description.

Ejemplo de llamada REST:

```
https://www.usanpn.org/npn_portal/phenophases/getAbundanceCategories.xml
```

3.5.5. USHAHIDI/CROWDMAP

Tanto en Ushahidi como en Crowdmap no existe un único mapa a editar por parte de los voluntarios, sino que existen muchísimos mapas distintos. Esto implica tener que implementar cada uno de estos mapas por separado, lo cual resulta inviable ya que es muy costoso y carente de sentido. Cada usuario de Ushahidi puede crear su propio mapa y compartirlo con los demás voluntarios a través de un plugin. En Ushahidi se han dado cuenta de que esto es un problema y en la actualidad se está trabajando con mucha celeridad para tener lista lo antes posible una API global que trabaje sobre un único mapa. Lo que se ha hecho es implementar tres funciones concretas de Ushahidi para acceder a la API pública de desarrollo: palabra clave, localización y categoría.

El propósito de esta API es listar todos los sitios que han optado por ser públicos. La razón de ser de esta API fue para que el sitio web de la Comunidad pudiera ofrecer implementaciones, mientras que proporciona a los administradores una manera de administrar la información públicamente sobre su implementación.

BÚSQUEDA POR PALABRA CLAVE

- Punto final: http://tracker.ushahidi.com/list/
- Descripción: Lista aquellos sitios públicos que coinciden con la palabra clave introducida.
- Métodos HTTP disponibles: GET
- Autenticación: No requerida.
- Parámetros requeridos:

Parámetro	Descripción	Ejemplo
q	Información a buscar.	q=haiti

Tabla 41 - Descripción de los parámetros requeridos
para la función de búsqueda por palabra clave en Ushahidi.

Ejemplo cURL:

```
http://tracker.ushahidi.com/list/?q=haiti
```

BÚSQUEDA POR LOCALIZACIÓN

- Punto final: http://tracker.ushahidi.com/list/
- Descripción: Lista aquellos sitios públicos que coinciden con una localización concreta, especificada mediante coordenadas geográficas (longitud y latitud) y un valor de distancia a modo de rango de influencia (millas o kilómetros).
- Métodos HTTP disponibles: GET
- Autenticación: No requerida.
- Parámetros requeridos:

Parámetro	Descripción	Ejemplo
Lat	Valor de la latitud de un punto.	lat=55.3
Lon	Valor de la longitud de un punto.	lon=-3.4
distance	Valor del radio de influencia en un punto.	distance=100
units	Unidad de medida: Kilómetros o Millas.	units=km

Tabla 42 - Descripción de los parámetros requeridos
para la función de búsqueda por localización en Ushahidi.

Ejemplo cURL:

```
http://tracker.ushahidi.com/list/?lat=55.3&lon=-3.4&distance=100&units=km
```

BÚSQUEDA POR CATEGORÍA

- Punto final: http://tracker.ushahidi.com/list/
- Descripción: Permite conocer los nombres de las categorías a las que pertenecen cada uno de los mapas desarrollados en Ushahidi/Crowdmap.
- Métodos HTTP disponibles: GET
- Autenticación: No requerida.
- Parámetros requeridos:

Parámetro	Descripción	Ejemplo
cat	Valor de una categoría concreta. El valor 'all' devuelve el nombre de todas las categorías.	cat=all

Tabla 43 - Descripción de los parámetros requeridos
para la función de búsqueda por categoría en Ushahidi.

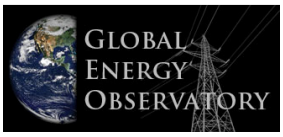
Ejemplo cURL:

```
http://tracker.ushahidi.com/list/?cat=all
```

En este momento, hay 13 categorías diferentes: Arts & Entertainment, Science & Technology, Academic, Recreation, Business & Finance, News, Computers, Just for fun, Politics, Government, Historical, Health y Other.

3.6. OTROS SERVICIOS WEB BASADOS EN REST

En este apartado se especifican de forma detallada otras APIs REST que han sido implementadas en el prototipo presentado. La información geográfica de alguna de ellas es ofrecida por organizaciones o empresas, y no por ciudadanos, sin embargo se considera que pueden ser de gran interés y ayuda en un momento determinado. Estas APIs pertenecen al Observatorio de Energía Global, el Servicio Geológico de los Estados Unidos, World Weather Online y Panoramio.



3.6.1. GLOBAL ENERGY OBSERVATORY

El Observatorio de Energía Global es un conjunto de bases de datos y herramientas interactivas libres construidas de forma colaborativa por usuarios. El objetivo principal es promover un entendimiento, a escala global, de las dinámicas de cambio en los sistemas energéticos, la cuantificación de emisiones y sus impactos, así como acelerar la transición a carbón neutral, sistemas de energía benévolos para el medioambiente mientras se suministra energía al alcance de todos. Al proporcionar datos, modelos y herramientas de análisis visuales y fáciles de utilizar se consigue involucrar tanto a expertos como al público en general. Los datos en GEO pueden editarse desde cualquier sitio del mundo.

Este proyecto fue concebido y desarrollado por el Dr. Rajan Gupta y está apoyado por un grupo considerable de colaboradores que tienen un profundo interés en el desarrollo mundial así como en el desarrollo energético, climatológico y medioambiental. Se cree que los sistemas complejos, que son impulsados en gran medida por la sociedad, la política, la económica, el medioambiente, la tecnología y los factores de disponibilidad de los recursos, requieren de la participación de un público bien informado para motivar al observatorio a apoyar este sitio web y las bases de datos.

El observatorio de energía global pretende ayudar a acelerar los tres aspectos del problema mundial de la energía:

- El acceso de la población mundial a una energía asequible.
- Trasladar los sistemas energéticos hacia los sistemas neutros de carbono.
- Lograr estos objetivos de una manera ambientalmente responsable y que económicamente sea viable.

El acceso a energía barata de carbono-neutral es esencial tanto para el desarrollo humano, como para la gestión del medio ambiente, así como para el mantenimiento de las sociedades industriales modernas. Junto con el agua potable, la atención médica y la educación, la necesidad de acceder por parte de todos a una energía de carbono-neutral que medioambientalmente sea amigable es urgente.

La infraestructura energética existente es enorme y muy compleja. Como un primer paso hacia una mejor comprensión de los desafíos que se tienen, el Observatorio de Energía Global está proporcionando información geo-espacial pública en el sistema actual. La idea es llevar a cabo un análisis de los posibles escenarios para el desarrollo mediante la participación de los expertos y el público al mismo tiempo. El primer paso hacia estos objetivos es apreciar lo vasta y compleja que es la infraestructura energética existente, y entender cómo funciona. Se cree que, en la actualidad, esta tarea es posible utilizando las herramientas software públicas (como Google, Google Earth, Web 2.0, Wikipedia, etc.) y un esfuerzo por parte de la comunidad.

Este sitio web representa grandes esfuerzos para recoger, recopilar y organizar la inmensa cantidad de información disponible y proporcionarlo a la población mundial como una plataforma abierta para fomentar el debate y el desarrollo. Si desea disponer de información más detallada sobre el observatorio, diríjase a la siguiente página Web:

http://globalenergyobservatory.org/

DESCRIPCIÓN DEL SERVICIO WEB BASADO EN REST

- Métodos HTTP disponibles: GET
- Resumen URL:

Parámetro	URL
query	http://globalenergyobservatory.org/constructNetworkUtility.php

Tabla 44 - URL para la función constructNetworkUtility en la API REST del Global Energy Observatory.

A continuación se muestran las categorías que comprende la API REST del Observatorio de Energía Global con sus correspondientes tipos:

- Plantas energéticas
  - Centrales eléctricas de carbón
  - Centrales eléctricas de gas
  - Centrales de energía geotérmica
  - Centrales hidroeléctricas
  - Centrales de energía nuclear
  - Centrales de gasoil
  - Centrales solares fotovoltaicas
  - Centrales solares termales
  - Centrales de residuos
  - Centrales eólicas

- Combustibles y recursos energéticos

  - Yacimientos de gas
  - Yacimientos de petróleo
  - Minas de carbón
  - Minas de uranio
  - Refinerías de petróleo crudo
  - Potencial solar
  - Potencial eólico
  - Potencial de Biomasa
  - Almacenamiento de CO2
  - Recursos hídricos

- Transmisión de energía

  - Gasoductos
  - Oleoductos
  - Puertos de carbón
  - Puertos de petróleo
  - Puertos de gas natural licuado
  - Enlaces ferroviarios
  - Enlaces por carretera
  - Rutas navegables
  - Red de suministro eléctrico

- Consumidores de energía

  - Industrias
  - Población
  - Actividad económica
  - Ciudades

PLANTAS ENERGÉTICAS

Campos o atributos de datos comúnes a las plantas energéticas:

Campo	Descripción
Name	Nombre oficial de la planta energética asignado por el gobierno/propietario.
Country-Plant-ID	Identificador oficial asignado a la planta energética por el gobierno del país.
OpenModel-PP-ID	Identificador único no editable asignado a la planta energética por el observatorio de enería global.
Latitude	Latitud de la planta energética. Valor comprendido entre -90 y +90 (grados sexagesimales).
Longitude	Longitud de la planta energética. Valor comprendido entre -180 y +180 (grados sexagesimales).
Design Capacity	Capacidad específica de la planta energética en el momento del diseño en megavatios (MW).
Realized Capacity	Capacidad oficial de realización basada en la experiencia de operación expresada en megavatios (MW).
Heat Supply Capacity	Capacidad para suministrar calor en megavatios térmicos.
Location	Dirección disponible de la planta energética que incluye la ciudad, el estado y el país.
Type of Plant	Tipo de central energética.
Source of fuel	Breve descripción del combustible primario de la fuente.
Source of Water	Breve descripción de la fuente de agua para diversas operaciones.
Capital cost of Plant	Coste total de la planta.

Tabla 45 - Lista de atributos de datos comúnes en las plantas energéticas.

- Parámetros de entrada:

Campo	Requerido	Tipo	Valores
op	SI	string	menu_latMapMarkers_from_file
lat	SI	int	0
latOffset	SI	int	89
lng	SI	int	0
lngOffset	SI	int	179
category	SI	string	PowerPlants
markerColor	SI	string	FA58AC
type	SI	string	Posibles valores: Coal, Gas, Geothermal, Hydro, Nuclear, Oil, Solar_PV, Solar_Thermal, Waste o Wind.
typeId	SI	int	Lista de posibles valores: 0 (Coal), 1 (Gas), 2 (Geothermal), 3 (Hydro), 4 (Nuclear), 5 (Oil), 6 (Solar_PV), 7 (Solar_Thermal), 8 (Waste) o 9 (Wind).

Tabla 46 - Parámetros de entrada de la función constructNetworkUtility en la API REST del Global Energy Observatory para las Plantas Energéticas.

Los parámetros lat y lng hacen referencia a las coordenadas latitud y longitud respectivamente del punto central de aquella extensión de superficie terrestre que se está visualizando en el mapa base. Por otro lado, los parámetros latOffset y lngOffset corresponden a la superficie (en grados decimales) que se desea observar. El valor máximo de latOffset nunca puede superar los 90º de latitud, del mismo modo que el valor máximo de lngOffset no puede ser mayor a 180º de latitud (ver Figura 8):

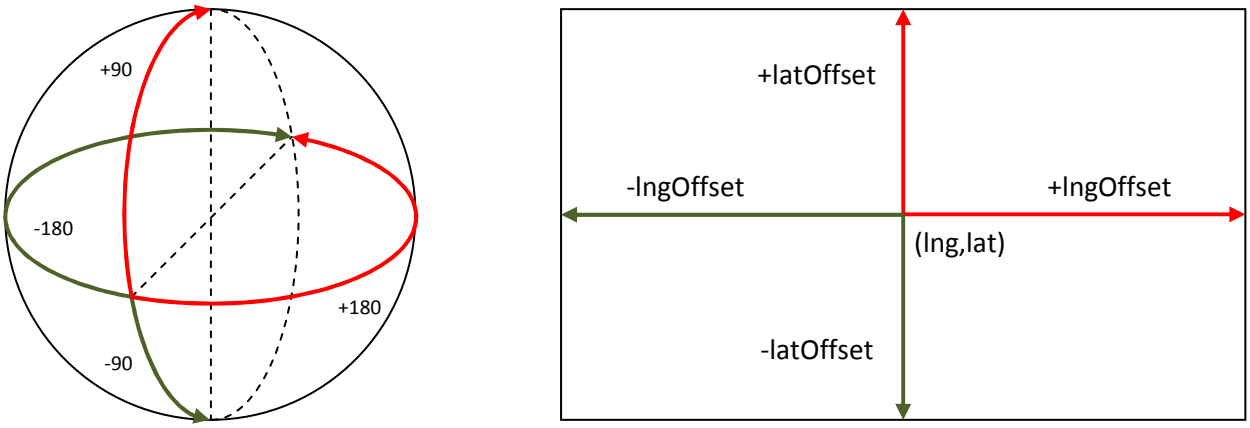


Figura 9 - Representación gráfica de los parámetros lat, lng, latOffset y lngOffset de la API REST del Global Energy Observatory:
(a) Superficie terrestre y (b) Proyección de la superficie terrestre sobre el plano

Ejemplo de llamada REST:

http://globalenergyobservatory.org/constructNetworkUtility.php?op=menu_latMapMarkers_from_file&lat=0&latOffset=89&category=PowerPlants&lng=0&lngOffset=179&markerColor=FA58AC&type=Coal&typeId=0

La figura 10 muestra las centrales eléctricas de carbón geolocalizadas dentro de los límites establecidos.

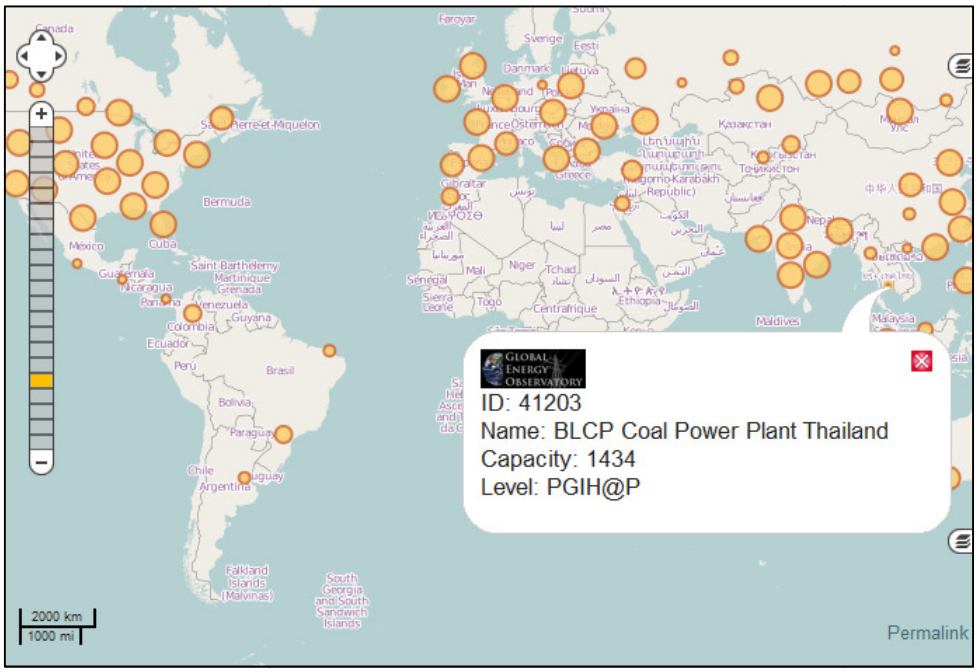


Figura 10 - Geolocalización de las centrales energéticas de carbón mediante el uso de la API REST del Global Energy Observatory.

Los diferentes tamaños para cada una de las entidades es debido a una estrategia de tolerancia cluster que se ha utilizado en la implementación del presente prototipo de aplicación Web. Gracias a esta estrategia aquellas entidades que aparecen muy próximas entre sí se agrupan o aglutinan en una única entidad de mayor tamaño para que la visualización sea más fácil, cómoda e intuitiva. A medida que el usuario se aproxima a las entidades agrupadas haciendo zoom, estas van a ir disgregándose en otras hasta conseguir visualizar cada elemento de forma individualizada.

COMBUSTIBLES Y RECURSOS ENERGÉTICOS

Campos o atributos de datos comúnes a las plantas energéticas:

Campo	Descripción
Name	Nombre oficial de la planta energética asignado por el gobierno/propietario.
Country-Plant-ID	Identificador oficial asignado a la planta energética por el gobierno del país.
OpenModel-PP-ID	Identificador único no editable asignado a la planta energética por el observatorio de enería global.
Latitude	Latitud de la planta energética. Valor comprendido entre -90 y +90 (grados sexagesimales).
Longitude	Longitud de la planta energética. Valor comprendido entre -180 y +180 (grados sexagesimales).
Design Capacity	Capacidad específica de la planta energética en el momento del diseño en megavatios (MW).

Location	Dirección disponible de la planta energética.
Distance to Nearest Road Head	Distancia al servicio de carreteras más próximo.
Distance to Nearest Rail Head	Distancia al servicio ferroviario más próximo.
Distance to Nearest Transmission	Distancia a la red de transmisión eléctrica más cercana.
Capital cost	Coste total de este servicio en el momento de la primera construcción.
Environmental Issues	Lista de cuestiones ambientales importantes (efluentes, emisiones, residuos generados, impactos del agua, el ruido, ...).

Tabla 47 - Lista de atributos de datos comúnes en los combustibles y los recursos energéticos.

- Parámetros de entrada:

Campo	Requerido	Tipo	Valores
op	SI	string	menu_latMapMarkers_from_file
lat	SI	int	0
latOffset	SI	int	89
lng	SI	int	0
lngOffset	SI	int	179
category	SI	string	Resources
markerColor	SI	string	81BEF7
type	SI	string	Posibles valores: Gas_Fields, Oil_Fields, Coal_Mines, Uranium_Mines, Crude_Oil_Refineries, Solar_Poten-tial, Wind_Potential, Biomass_Potential, CO2_Storage o Water_Resources.
typeId	SI	int	Lista de posibles valores: 0 (Gas_Fields), 1 (Oil_Fields), 2 (Coal_Mines), 3 (Uranium_Mines), 4 (Crude_Oil_Refineries), 5 (Solar_Potential), 6 (Wind_Potential), 7 (Biomass_Potential), 8 (CO2_Storage) o 9 (Water_Resources).

Tabla 48 - Parámetros de entrada de la función constructNetworkUtility en la API REST del Global Energy Observatory para los combustibles y los recursos energéticos.

Ejemplo de llamada REST:

http://globalenergyobservatory.org/constructNetworkUtility.php?op=menu_latMapMarkers_from_file&lat=0&latOffset=89&category=Resources&lng=0&lngOffset=179&markerColor=81BEF7&type=CO2_Storage&typeId=8

La figura 11 muestra la geolocalización de los recursos de almacenamiento de CO2 existentes dentro de unos límites establecidos. Al igual que en el caso anterior, no hay limitación alguna y se muestran los resultados para todo el mundo. En este caso también puede comprobarse que se ha utilizado una tolerancia cluster para representar cada una de las entidades existentes.

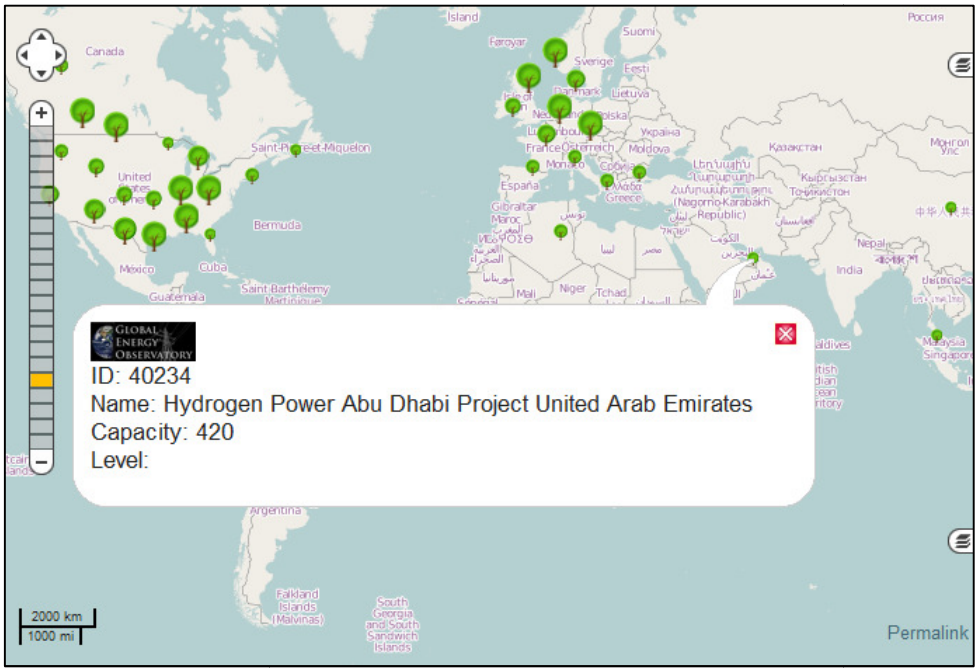


Figura 11 - Geolocalización de los recursos de almacenamiento de dióxido de carbono mediante el uso de la API REST del Global Energy Observatory.

TRANSMISIÓN DE ENERGÍA

- Parámetros de entrada:

Campo	Requerido	Tipo	Valores
op	SI	string	menu_latMapMarkers_from_file
lat	SI	int	0
latOffset	SI	int	89
lng	SI	int	0
lngOffset	SI	int	179
category	SI	string	Transmission
markerColor	SI	string	F2F5A9
type	SI	string	Posibles valores: Gas_Pipelines, Oil_Pipelines, Coal_Ports, Oil_Ports,LNG_Ports, Rail_Links, Road_Links, Shipping_Lanes o Electric_Power_Grid.
typeId	SI	int	Lista de posibles valores: 0 (Gas_Pipelines), 1 (Oil_Pipelines), 2 (Coal_Ports), 3 (Oil_Ports), 4 (LNG_Ports), 5 (Rail_Links), 6 (Road_Links), 7 (Shipping_Lanes) o 8 (Electric_Power_Grid).

Tabla 49 - Parámetros de entrada de la función constructNetworkUtility en la API REST del Global Energy Observatory para la transmisión de energía.

Ejemplo de llamada REST:

http://globalenergyobservatory.org/constructNetworkUtility.php?op=menu_latMapMarkers_from_file&lat=0&latOffset=89&category=Transmission&lng=0&lngOffset=179&markerColor=F2F5A9&type=Oil_Ports&typeId=3

La figura 12 muestra la transmisión de energía a través de puertos petrolíferos. Al igual que en los casos anteriores, no se ha restringido el espacio geográfico y se ha utilizado una tolerancia clúster para mostrar gráficamente cada una de las entidades obtenidas.

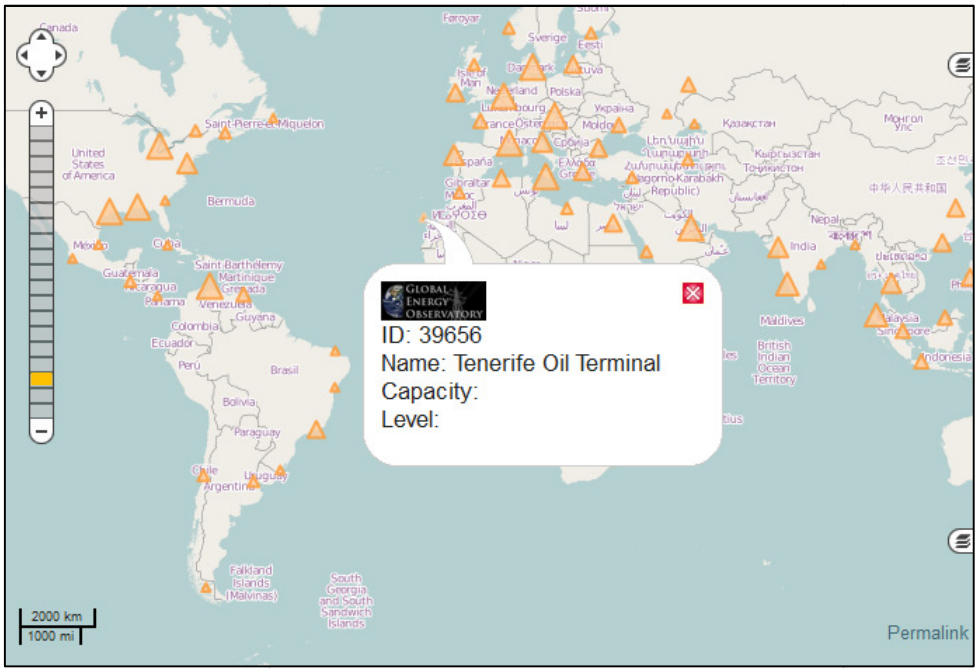


Figura 12 - Geolocalización de la transmisión de energía a través de puertos petrolíferos mediante el uso de la API REST del Global Energy Observatory.

CONSUMIDORES DE ENERGÍA

- Parámetros de entrada:

Campo	Requerido	Tipo	Valores
op	SI	string	menu_latMapMarkers_from_file
lat	SI	int	0
latOffset	SI	int	89
lng	SI	int	0
lngOffset	SI	int	179
category	SI	string	Consumers
markerColor	SI	string	F2F5A9
type	SI	string	Posibles valores: Industries, Population, Economic_Activity o Towns_and_Cities.
typeId	SI	int	Lista de posibles valores: 0 (Industries), 1 (Population), 2 (Economic_Activity) o 3 (Towns_and_Cities).

Tabla 50 - Parámetros de entrada de la función constructNetworkUtility en la API REST del Global Energy Observatory para los consumidores de energía.

La figura 13 muestra la geolocalización de los consumidores de energía de tipo industrial existentes en todo el mundo, es decir que no hay ningún tipo de limitación establecida. Al igual que en los casos anteriores, también se ha utilizado una tolerancia de agrupamiento para mostrar las entidades devueltas desde el servidor del observatorio.

Ejemplo de llamada REST:

```
http://globalenergyobservatory.org/constructNetworkUtility.php?
op=menu_latMapMarkers_from_file&lat=0&latOffset=89&category=Consumers&lng=0&
lngOffset=179&markerColor=F2F5A9&type=Industries&typeId=0
```

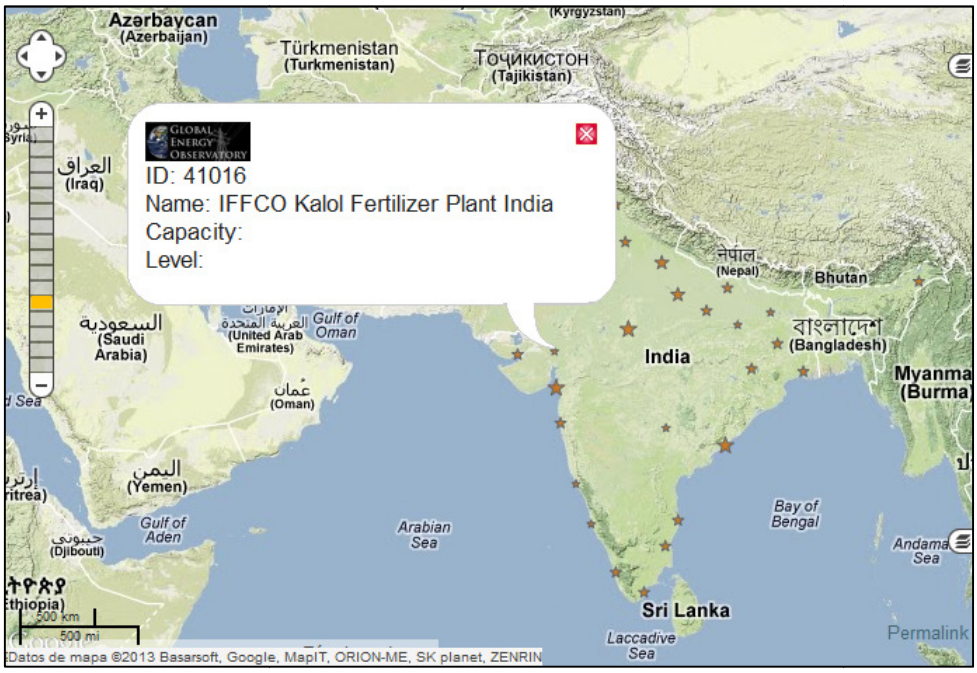


Figura 13 - Geolocalización de los consumidores de energía de tipo industrial mediante el uso de la API REST del Global Energy Observatory.

3.6.2. USGS

El Servicio Geológico de Estados Unidos (USGS) es una agencia científica del gobierno de los Estados Unidos de América fundada el 3 de marzo de 1879. Se trata de una organización investigadora sin responsabilidades reguladoras. Este servicio proporciona información relevante sobre el estado de los ecosistemas y del medio ambiente, los peligros naturales (principalmente terremotos), los recursos naturales, los impactos climatológicos y los cambios en los usos del suelo, entre otros.

El USGS dispone de más de 10000 expertos (científicos, técnicos y personal de apoyo) de una amplia gama de disciplinas. Ellos recogen, analizan y proporcionan los conocimientos científicos sobre las condiciones de los recursos naturales, problemas y otras cuestiones. El objetivo principal del USGS consiste en proporcionar información científica relevante para describir y comprender la Tierra, minimizar la pérdida de vidas debido a catástrofes naturales, gestionar recursos como el agua, la energía, los minerales, etc. resaltando y protegiendo la calidad de vida de los seres humanos.

El USGS se centra en algunos de los temas más importantes a los que se enfrenta la sociedad, en los que la ciencia natural puede contribuir sustancialmente al bienestar de todos. La Estrategia científica de USGS resume los principales problemas sociales a los que dicho servicio está a punto de enfrentarse. También se han creado estrategias específicas para cada una de esas áreas para ampliar y avanzar en las acciones que se pueden tomar en la próxima década:

- Sistemas científicos centrales
- Clima y cambio de usos del suelo
- Energía y Minerales
- Salud ambiental
- Ecosistemas
- Riesgos naturales
- Agua

El USGS tiene un programa de pelígro por terremotos que forma parte del Programa Nacional de Reducción de Peligro por Terremotos (NEHRP) establecido en un congreso en el año 1977. Se hace un seguimiento y un informe de los terremotos, se evalúan sus impactos y peligros, y se investigan las causas y efectos de dichos terremotos. Puede encontrar más información sobre dicho programa en la siguiente Web:

http://www.nehrp.gov/

Los científicos del USGS estudian y hacen un seguimiento exhaustivo de todos los terremotos que se producen en el mundo continuamente. Si desea más información sobre sobre este Departamento de Estudios Geológicos de Estados Unidos, consulte la siguiente página web:

http://www.usgs.gov/

Aprovechando esta información, se ha optado por implementar parte de la API REST de este servicio, de manera que el usuario pueda acceder a la información sísmica global cuando se desee. Así puede conocerse la geolocalización y magnitud (escala sismológica de Richter) de los diferentes terremotos registrados por el USGS (por hora, día o semana). A continuación se muestran las diferentes URLs necesarias para poder llamar al servidor remoto del USGS y que éste nos devuelva el fichero en formato XML con la información deseada. No es necesario introducir parámetros de entrada en las llamadas.

Ejemplos de llamada REST al servidor del USGS:

- Terremotos ocurridos en la última hora en todo el mundo:

```
http://earthquake.usgs.gov/earthquakes/catalogs/eqs1hour-M0.xml
```

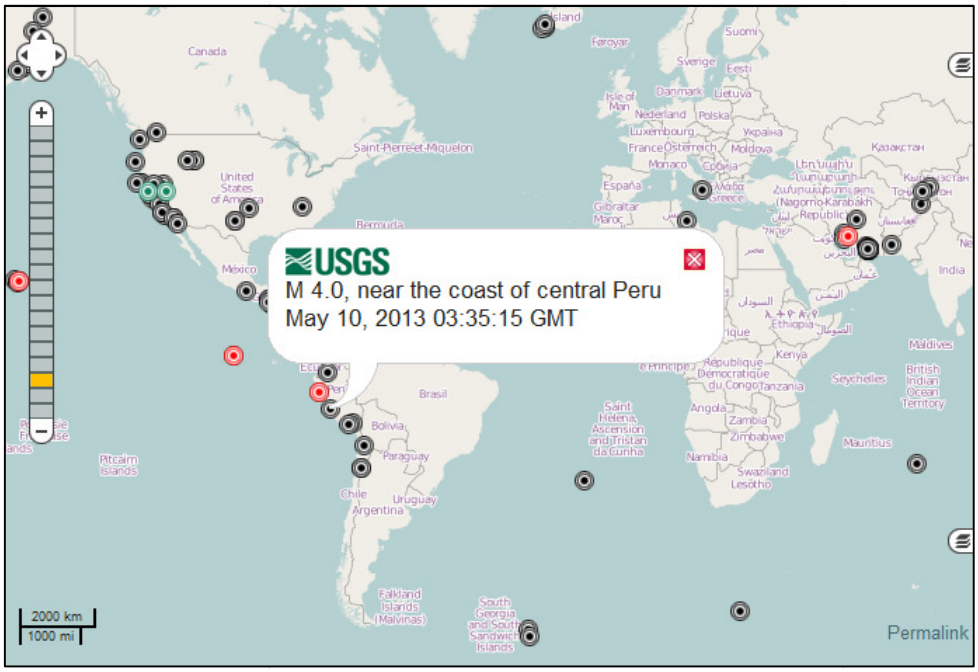
- Terremotos ocurridos en el último día con una magnitud mayor a 2,5 en la escala sismológica de Richter en todo el mundo:

```
http://earthquake.usgs.gov/earthquakes/catalogs/eqs1day-M2.5.xml
```

- Terremotos ocurridos en la última semana con una magnitud mayor a 2,5 en la escala sismológica de Richter en todo el mundo:

http://earthquake.usgs.gov/earthquakes/catalogs/eqs7day-M2.5.xml

A continuación puede verse el resultado visual sobre el mapa base tras efectuar dichas llamadas en el portal web desarrollado para este trabajo, con su correspondiente leyenda:



Leyenda: Terremotos ocurridos en la última hora.
Terremotos ocurridos durante el último día.
Terremotos ocurridos durante la última semana.

Figura 14 - Visualización de terremotos a nivel mundial (13 de mayo de 2013, 18:00h) mediante la utilización de la API REST del USGS.

El ususario puede conocer los detalles de cada terremoto visualizado en el mapa haciendo clic con el cursor sobre el icono correspondiente. En la Figura 14 se muestra información detallada (magnitud, nombre, fecha y hora) sobre un seismo acontecido días antes.

3.6.3. WORLD WEATHER ONLINE

World Weather Online tiene dos APIs (Free y Premium) que proporcionan una manera de obtener datos climatológicos en tiempo real, pasado e incluso previsiones futuras. Las APIs proporcionan información climatológica mediante peticiones hechas sobre el estándar HTTP, devolviendo datos en formatos fáciles de utilizar como XML, JSON, JSON-P, CSV y TSV. Estos datos son utilizados en todo el mundo por agencias de viajes,

puertos deportivos, líneas aéreas, compañías de telefonía móvil, marineros, propietarios de sitios web, etc.

A diferencia de otros proveedores de contenidos meteorológicos, esta web genera su propio pronóstico del tiempo, por lo que dependen de terceros para los datos meteorológicos. Los proveedores de World Weather Online obtienen sus datos a partir del modelo atmosférico ECMWF, que es una organización meteorológica mundial, además de otros modelos como el del Global Forecast System (GFS), la Agencia Meteorológica del norte de Asia, así como también consultando las imágenes climatológicas provenientes de satélites geoestacionarios. Estas fuentes de datos permiten proporcionar información climatológica en todo el mundo. Se utilizan varios modelos oceánicos y atmosféricos para proporcionar previsiones meteorológicas precisas. La web tiene meteorólogos experimentados que revisan los resultados del modelo climatológico para poder asegurar de forma fidedigna que la previsión sea la correcta.

La API Free proporciona tanto datos actuales, así como otras llamadas de utilidad; mientras que la API Premium proporciona más previsiones meteorológicas, sin ningún tipo de limitación, así como datos históricos. Es necesario aclarar que solamente ha sido implementada la API Free de World Weather Online, ya que la API Premium requiere pagar una cuota mensual. Para utilizar cualquiera de estas APIs es necesario previamente obtener una clave. Para ello es preciso acceder a la página de registro de la API y rellenar el formulario correspondiente. Una vez creada y verificada la cuenta de usuario, se podrá crear la clave accediendo a la cuenta.

Los datos climatológicos se proporciona para las siguientes categorías y actividades:

- Clima local de ciudades/poblaciones
- Clima marino
- Clima para deportes
- Promedios climatológicos mensuales

La API cubre aproximadamente la información climatológica de tres millones de ciudades y poblaciones. Además no hay ningún tipo de limitación mensual en el número de peticiones o llamadas al servidor, aunque se pide que no se hagan más de 500 peticiones por hora. Los datos climatológicos son actualizados cada 3-4 horas y los datos climatológicos marinos cada 6-8 horas. A continuación se describe las diferentes llamadas de la API Free que han sido implementadas en el prototipo.

- Clima local
- Clima marino
- Hora local
- Buscar localización

CLIMA LOCAL

Se accede a las condiciones climatológicas actuales así como a una previsión fiable para los cinco días siguientes. Esta función devuelve elementos climatológicos tales como la temperatura, la precipitación, la descripción climatológica, la imagen o el icono climatológico correspondiente y la velocidad del viento.

- Métodos HTTP disponibles: GET
- Resumen URL:

Parámetro	URL	Método
query	http://api.worldweatheronline.com/free/v1/weather.ashx	GET

Tabla 51 - URL para la función weather en la API REST Free de World Weather Online.

- Parámetros de entrada:

Campo	Requerido	Descripción
q	SI	El campo q hace referencia a la geolocalización (ver tabla 53).
extra	NO	Incluye información extra.
num_of_days	SI	Número de días de predicción.
date	NO	Especifica el clima para un día concreto. Los valores válidos son 'today', 'tomorrow' y 'YYYY-MM-DD'.
fx	NO	Indicador opcional para decidir si se tiene que devolver la predicción climatológica o no. Los posibles valores son 'yes' (valor por defecto) o 'no'.
cc	NO	Indicador opcional para decidir si se tiene que devolver la climatológica actual o no. Los posibles valores son 'yes' (valor por defecto) o 'no'.
includeLocation	NO	Indicador opcional para decidir si se tiene que devolver el punto climatológico más cercano al punto requerido: 'yes' o 'no' (valor por defecto).
format	NO	Formato de salida a devolver: 'xml' (valor por defecto), 'json' y 'csv'.
show_comments	NO	Indicador opcional para incluir comentarios o no. Los posibles valores son 'yes' (valor por defecto) o 'no'.
callback	NO	Nombre de la función para el callback de JSON.
key	SI	Clave de la API proporcionada al registrarse.

Tabla 52 - Parámetros de entrada de la función weather en la API REST Free de World Weather Online.

El parámetro q puede adoptar una de las siguientes 4 formas para describir la localización de un punto del cual se desea conocer su climatología:

Tipo de localización	Formato	Ejemplos
Nombre de la ciudad o población	Nombre ciudad Nombre ciudad, Estado Nombre ciudad, Estado, País Nombre ciudad, País	q=New+York q=New+York,ny q=Valencia,Spain
Dirección IP	XXX.XXX.XXX.XXX	q=101.25.32.325
Código postal UK, US o Canadá	Formato del código postal o código zip.	q=SW1 q=90201
Latitud y longitud	XX.XXX,XX.XXX (grados decimales)	q=48.834,2.394

Tabla 53 - Diferentes formatos que puede adoptar el parámetro q dentro de las funciones weather, tz y search en la API REST Free de World Weather Online.

- Parámetros de salida:

Campo	Descripción
type	Tipo de localización: 'city', 'latlon', 'postcode', 'zipcode', etc.
query	Petición efectuada.
observation_time	Momento de la observación. Ejemplo: 06:45 AM.
temp_C	Temperatura en grados centígrados.
windspeedMiles	Velocidad del viento en millas por hora.
windspeedKmph	Velocidad del viento en kilómetros por hora.
winddirDegree	Dirección del viento en grados.
winddir16Point	Dirección del viento en forma de Brújula.
weatherCode	Código de la condición climatológica.
weatherDesc	Descripción de la condición climatológica.
weatherIconUrl	URL hacia el icono climatológico.
precipMM	Valor de la precipitación en milímetros.
humidity	Porcentaje de humedad.
visibility	Visibilidad en kilómetros.
pressure	Presión atmosférica en milibares.
cloudcover	Porcentaje de nubosidad.
date	Fecha de previsión climatológica local.
tempMaxC	Temperatura máxima del día en grados centesimales.
tempMaxF	Temperatura máxima del día en grados Fahrenheit.
tempMinC	Temperatura mínima del día en grados centesimales.
tempMinF	Temperatura mínima del día en grados Fahrenheit.

Tabla 54 - Parámetros de salida de la función weather en la API REST Free de World Weather Online.

Ejemplo de llamada REST:

```
http://api.worldweatheronline.com/free/v1/weather.ashx?
key=xxxxxxxxxxxxxxxxx&q=valencia,spain&num_of_days=5&format=xml
```

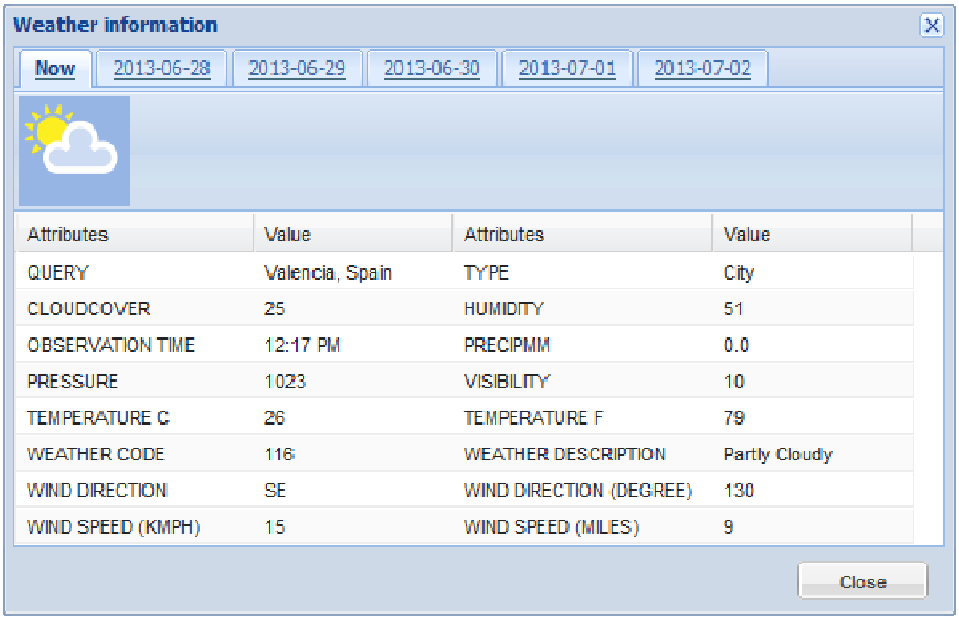


Figura 15 - Información climatológica implementada gracias a la función weather de la API REST de World Weather Online.

En la figura anterior se muestra la función weather implementada en el prototipo del geoportal realizado para este proyecto. En ella puede verse el valor de todos los parámetros de salida devueltos para la fecha actual (pestaña NOW), así como las predicciones correspondientes a los cinco siguientes días para esa misma localización (Valencia, España): Hora de la observación, temperatura, velocidad del viento, presión, precipitación, etc.

CLIMA MARINO

Se accede a la previsión climatológica marina/navegación en el día actual para una localización (longitud, latitud) dada. Esta API devuelve elementos climatológicos tales como la temperatura, la precipitación, la descripción climatológica, velocidad y dirección del viento, altura significante de las olas, altura, dirección y periodo del mar de fondo.

- Métodos HTTP disponibles: GET
- Resumen URL:

Parámetro	URL	Método
query	http://api.worldweatheronline.com/free/v1/marine.ashx	GET

Tabla 55 - URL para la función marine en
la API REST Free de World Weather Online.

- Parámetros de entrada:

Campo	Requerido	Descripción
q	SI	La localización debe especificarse únicamente mediante las coordenadas longitud y latitud (en grados decimales).
fx	NO	Indicador opcional para decidir si se tiene que devolver la predicción climatológica o no. Los posibles valores son 'yes' (valor por defecto) o 'no'.
format	NO	Formato de salida. Valores posibles: 'xml' (por defecto), 'json' y 'csv'.
callback	NO	Nombre de la función para el callback de JSON.
key	SI	Clave de la API proporcionada al registrarse.

Tabla 56 - Parámetros de entrada de la función marine en
la API REST Free de World Weather Online.

- Parámetros de salida:

Campo	Descripción
type	Tipo de petición en la localización. El valor admitido es 'latlon'.
query	Petición efectuada.
latitude	Latitud (grados decimales).
longitude	Longitud (grados decimales).
distance_miles	Distancia (millas) entre la localización introducida y la posición del área más próxima.

date	Fecha de previsión climatológica local.
maxtempC	Temperatura máxima del día en grados centesimales.
mintempC	Temperatura mínima del día en grados centesimales.
hourly	Información de la previsión meteorológica para cada hora.
tempMinF	Temperatura mínima del día en grados Fahrenheit.
tempMaxC	Temperatura máxima del día en grados centesimales.
tempMaxF	Temperatura máxima del día en grados Fahrenheit.
tempMinC	Temperatura mínima del día en grados centesimales.
tempMinF	Temperatura mínima del día en grados Fahrenheit.
windspeedMiles	Velocidad del viento en millas por hora.
windspeedKmph	Velocidad del viento en kilómetros por hora.
winddirDegree	Dirección del viento en grados.
winddirection	Dirección del viento en forma de Brújula.
weatherCode	Código de la condición climatológica.
weatherDesc	Descripción de la condición climatológica.
weatherIconUrl	URL hacia el icono climatológico.
precipMM	Valor de la precipitación en milímetros.
humidity	Porcentaje de humedad.
visibility	Visibilidad en kilómetros.
pressure	Presión atmosférica en milibares.
cloudcover	Porcentaje de nubosidad.
sigHeight_m	Altura (metros) de las olas significantes.
swellHeight_m	Alturas (metros) de las olas de mar de fondo.
swellDir	Dirección (grados decimales) de las olas de mar de fondo.
swellPeriod	Periodo (segundos) del mar de fondo.
waterTemp_C	Temperatura del agua en grados centesimales.
waterTemp_F	Temperatura del agua en grados Fahrenheit.

Tabla 57 - Parámetros de salida de la función marine en la API REST Free de World Weather Online.

Ejemplo de llamada REST:

```
http://api.worldweatheronline.com/free/v1/marine.ashx?
key=xxxxxxxxxxxxxxxxx&q=48.834,2.394&format=xml
```

HORA LOCAL

Devuelve la hora local actual con su offset UTC (Tiempo Universal Coordinado) en horas y minutos para una localización específica.

- Métodos HTTP disponibles: GET
- Resumen URL:

Parámetro	URL	Método
query	http://api.worldweatheronline.com/free/v1/tz.ashx	GET

Tabla 58 - URL para la función tz en la API REST Free de World Weather Online.

- Parámetros de entrada:

Campo	Requerido	Descripción
q	SI	La localización puede especificarse mediante el nombre de la ciudad o población, la dirección IP, las coordenadas longitud y latitud (expresada en grados decimales).
format	NO	Formato de salida: 'xml' (valor por defecto), 'json' y 'csv'.
callback	NO	Nombre de la función para el callback de JSON.
key	SI	Clave de la API proporcionada al registrarse.

Tabla 59 - Parámetros de entrada de la función tz en la API REST Free de World Weather Online.

El parámetro q puede adoptar una de las 4 formas explicadas en la Tabla 53 para describir la localización de un punto del cual se desea conocer su climatología.

- Parámetros de salida:

Campo	Descripción
type	Tipo de petición en la localización. Los posibles valores son 'city', 'latlon', 'postcode', 'zipcode', etc.
query	Petición efectuada.
localTime	Hora local actual.
utcOffset	Desfase u offset UTC respecto a GMT en horas y minutos.

Tabla 60 - Parámetros de salida de la función tz en la API REST Free de World Weather Online.

Ejemplo de llamada REST:

```
http://api.worldweatheronline.com/free/v1/tz.ashx?
key=xxxxxxxxxxxxxxxxx&q=valencia,spain&format=xml
```

BUSCAR LOCALIZACIÓN

Devuelve información sobre una localización, incluyendo el país, población, región, coordenadas, …

- Métodos HTTP disponibles: GET
- Resumen URL:

Parámetro	URL	Método
query	http://api.worldweatheronline.com/free/v1/search.ashx	GET

Tabla 61 - URL para la función search en la API REST Free de World Weather Online.

- Parámetros de entrada:

Campo	Requerido	Descripción
query	SI	La localización debe especificarse únicamente mediante las coordenadas longitud y latitud (en grados decimales).
num_of_results	NO	Número de resultados a devolver.
timezone	NO	Indicador opcional para decidir si se tiene que devolver el desfase en horas respecto a GMT o no. Los posibles valores son 'yes' o 'no' (valor por defecto).
popular	NO	Indicador opcional para decidir si buscar solamente aquellas poblaciones más populares o no. Los posibles valores son 'yes' o 'no' (valor por defecto).
format	NO	Formato de salida a devolver. Los posibles valores son 'xml' (valor por defecto), 'json' y 'csv'.
key	SI	Clave de la API proporcionada al registrarse.

Tabla 62 - Parámetros de entrada de la función search en la API REST Free de World Weather Online.

El parámetro query puede adoptar una de las 4 formas explicadas en la Tabla 53 para describir la localización de un punto del cual se desea conocer su climatología.

- Parámetros de salida:

Campo	Descripción
result	Contiene información sobre una localización.
areaName	Nombre de la localización.
country	Nombre del país al que pertenece la localización.
region	Nombre de la región a la que pertenece la localización.
latitude	Latitud (grados decimales) de la localización.
longitude	Longitud (grados decimales) de la localización.
population	Población de la localización.
weatherUrl	URL en la que se encuentra la información climatológica de la localización.
timezone	Hora local de la localización.
offset	Desfase horario de la localización con respecto a GMT.

Tabla 63 - Parámetros de salida de la función search en la API REST Free de World Weather Online.

Ejemplo de llamada REST:

```
http://api.worldweatheronline.com/free/v1/search.ashx?
key=xxxxxxxxxxxxxxxxx&query=London&num_of_results=3&format=xml
```

En la siguiente figura puede verse la información climatológica ofrecida por esta API en diferentes regiones del área mostrada por el mapa base. Dicha información se muestra para 25 puntos equidistantes entre sí repartidos de forma homogenea sobre el mapa. Esto implica que se hagan 25 llamadas REST al servidor remoto de World Weather Online.

La información mostrada no corresponde a ningún tipo de predicción, sino que se trata de datos climatológicos actuales.

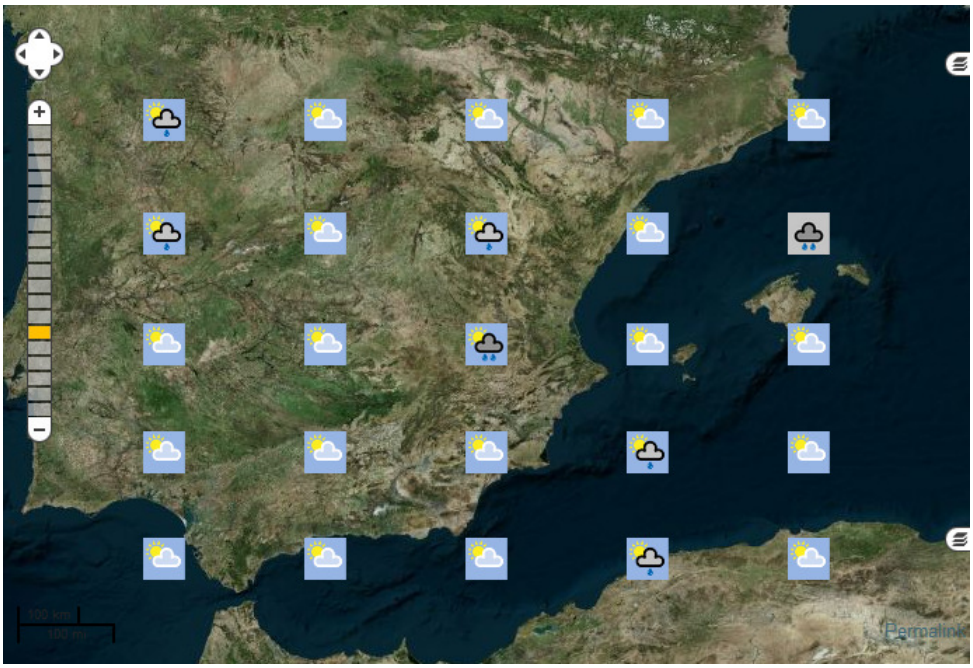


Figura 16 - Capa climatológica implementada de la API REST Free de World Weather Online.

Al navegar a través del mapa base mostrado en la figura, la información se refresca, repitiendo nuevamente las llamadas al servidor remoto pero para 25 puntos distintos. De este modo, se ofrece información climatológica actual de cuaquier sitio en cualquier momento de un modo rápido, fácil y fiable. Si desea más información sobre las diferentes APIs de World Weather Online, modo de uso, documentación, o información de contacto, consulte la siguiente página web:

http://developer.worldweatheronline.com/

3.6.4. PANORAMIO

Panoramio es un sitio web, disponible en varios idiomas, orientado a la geolocalización de fotografías. Su objetivo es permitir conocer mejor un área determinada mediante la visualización de fotografías de la zona tomadas por otros usuarios. Puede accederse a dichas fotografías mediante Google Earth y Google Maps. Panoramio fue creada oficialmente en octubre de 2005 por Joaquín Cuenca Abela y Eduardo Manchón Aguilar, dos emprendedores españoles. Un año y medio más tarde, la web de Panoramio contaba ya con más de un millón de usuarios y el número de fotografías había alcanzado los cinco millones. En julio de 2007, Google adquiere Panoramio. Joaquín Cuenca y Eduardo Manchón dejaron la compañía en 2010 para centrarse en un nuevo proyecto. La sede de Panoramio está actualmente situada en Zurich, en el edificio de oficinas de Google Suiza. Panoramio dispone de una API REST gratuita, tanto para fines

comerciales como para no comerciales, con la que es posible mostrar sus fotografías georreferenciadas en otro portal web ajeno. Dichas fotografías son estupendas para enriquecer los mapas y mostrar información relevante sobre lugares concretos (hospitales, cines, monumentos, inmobiliarias, hoteles, rutas, senderos, etc.). Por esta razón se ha considerado interesante implementar su API en el prototipo de estudio.

- Punto final: http://www.panoramio.com/map/get_panoramas
- Métodos HTTP disponibles: GET
- Resumen URL:

Parámetro	URL	Formato	Método
query	http://www.panoramio.com/map/get_panoramas	php	GET

Tabla 64 - URL para la función de búsqueda en Panoramio.

- Parámetros de entrada:

Campo	Tipo	Requerido	Cardinalidad	Descripción
order	string	SI	1	Órden en el que se van a mostrar las fotografías de Panoramio. Posibles valores: popularity (popularidad) o upload_date (fecha de actualización).
set	string	SI	1	Identificador que se utiliza para decidir las fotografías que se van a mostrar. Opciones: 'public' que muestra las fotos más populares, 'full' que muestra todas las fotos y 'user_id' que muestra únicamente las fotos de un usuario concreto.
from	int	SI	1	Parámetros que permiten establecer el número de fotografías a mostrar. El valor 0 representa la última fotografía actualizada en Panoramio. Por ejemplo, 'from=0 to=20' extraería el conjunto de las últimas 20 fotografías actualizadas. El número máximo de fotografías que pueden recuperarse en una llamada es 100.
to	int	SI	1	
minx	int	SI	1	Parámetros necesarios para definir el área en la que mostrar las fotografías.
miny	int	SI	1	
maxx	int	SI	1	
maxy	int	SI	1	
size	string	SI	1	Tamaño de las fotografías. Los posibles valores son 'original', 'medium' (valor por defecto), 'small', 'thumbnail' (miniatura), 'square' y 'mini_square'.

Tabla 65 - Parámetros de entrada de la función get_panoramas en la API REST de Panoramio.

- Parámetros de salida:

Campo	Descripción
height	Dimensión altura de la fotografía (píxeles).
width	Dimensión anchura de la fotografía (píxeles).
latitude	Coordenada latitud de la geolocalización de la fotografía.
longitude	Coordenada longitud de la geolocalización de la fotografía.
owner_id	Identificador del propietario de la fotografía.
owner_name	Nombre de usuario del propietario de la fotografía.
owner_url	URL del propietario de la fotografía.
photo_file_url	URL en la que se encuentra la fotografía.
photo_id	Identificador de la fotografía.
photo_title	Título de la fotografía.
photo_url	URL en la que se encuentra almacenada la fotografía.
upload_date	Fecha de la última actualización.

Tabla 66 - Parámetros de salida de la función get_panoramas en la API REST de Panoramio.

Ejemplo de llamada REST:

http://www.panoramio.com/map/get_panoramas.php?order=popularity&set=public&from=0&to=50&minx=-7&miny=35&maxx=8&maxy=43&size=thumbnail

Figura 17 - Ejemplo de utilización de la API REST de Panoramio.

La figura anterior muestra de forma detallada un ejemplo de llamada REST de la API de Panoramio. En la Figura 17 pueden verse el resultado final de dicha llamada con las diferentes geolocalizaciones de las fotografías de Panoramio encontradas en el área visualizada por el usuario. En este caso se ha limitado la búsqueda a 50 fotografías como puede apreciarse en la llamada efectuada.

El usuario puede visualizar y acceder a la información (título de la fotografía y nombre de usuario del autor) de cada fotografía clicando con el puntero del ratón sobre los diferentes iconos existentes sobre el mapa base. Si se hace clic sobre la fotografía se abrirá una nueva pestaña en el navegador en la que se mostrará el enlace de dicha fotografía en la web oficial de Panoramio.

Cualquier tipo de navegación sobre el mapa base implicará una nueva llamada a la API REST de Panoramio, ya que los parámetros de entrada minx, miny, maxx y maxy se verán alterados. Esto provocará de forma inmediata la actualización de la visualización de los resultados sobre el mapa.

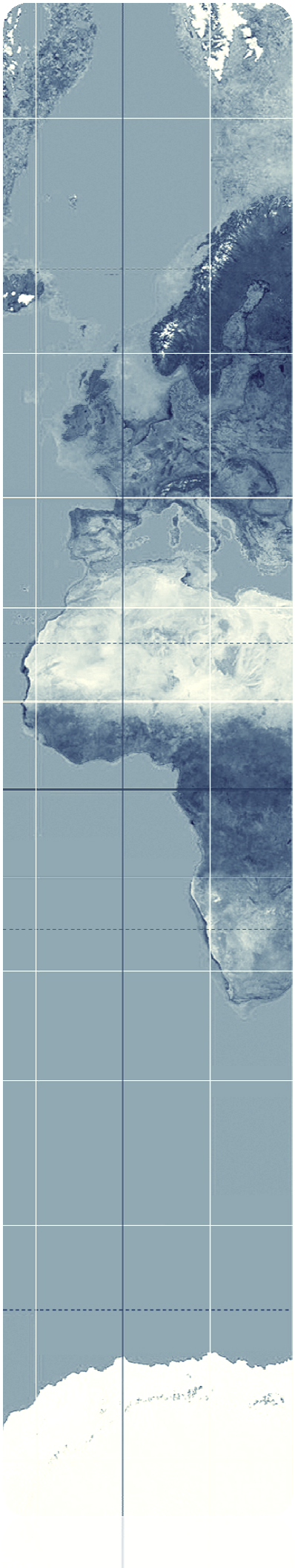
Si se desea conocer más información, visite la web de Panoramio:

http://www.panoramio.com

O bien, si se tiene alguna duda o sugerencia, puede contactar con Panoramio mediante el siguiente correo electrónico: questions@panoramio.com.

CAPÍTULO 4

PROTOTIPO DE LA APLICACIÓN WEB



4. PROTOTIPO DE LA APLICACIÓN WEB

El diseño de la aplicación Web es muy importante ya que modelará la interacción entre usuario y aplicación, y por tanto posibilitará o no la consecución de los objetivos perseguidos por el usuario. Se dice que una aplicación Web está bien diseñada cuando esta es comprensible, fácil de usar, amigable, clara, intuitiva y de fácil aprendizaje para el usuario. Para poder asegurar que un diseño cumple con estos requisitos se aconseja adoptar técnicas, procedimientos y métodos que aseguren empíricamente la adecuación del diseño a las necesidades, habilidades y objetivos del usuario. En este capítulo se describe con detalle todo aquello que tiene que ver con el prototipo de aplicación Web orientado a la información geográfica voluntaria creado para la defensa de este trabajo fin de máster.

4.1. APLICACIÓN WEB

En la actualidad se utilizan de forma predominante las aplicaciones Web en lugar de los sitios Web estáticos. Una aplicación Web es una aplicación software que se codifica en un lenguaje soportado por los navegadores Web. Estas aplicaciones son muy populares ya que estas aplicaciones son fáciles de actualizar y mantener, el navegador Web como cliente ligero es muy cómodo y el sistema operativo es independiente [17]. Las aplicaciones Web generan de forma dinámica un conjunto de páginas en un formato estándar (HTML o XHTML) que está soportado por los navegadores Web habituales.

4.1.1. ARQUITECTURA CLIENTE-SERVIDOR

La arquitectura cliente-servidor C/S es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios (servidores) y los demandantes (clientes). Un cliente realiza peticiones a otro programa, y es el servidor quien le retorna la respuesta.

La separación entre el cliente y el servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni se trata de un único programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, etc.

Una red cliente-servidor es aquella red de comunicaciones en la que todos los clientes están conectados a un mismo servidor, donde los recursos y aplicaciones con los que se cuenta están centralizados y puestos a disposición de los clientes. Los archivos de uso público así como los de uso restringido, los archivos que son de sólo lectura y los que pueden ser modificados, etc. se disponen en el servidor, que es la parte en la que se centran todas las gestiones que se realizan.

El cliente tiene un rol activo en la comunicación, ya que es quien inicia las solicitudes o peticiones y finalmente recibe las respuestas del servidor. El cliente puede conectarse a varios servidores simultáneamente. El cliente interactúa con los usuarios finales mediante la interfaz gráfica de usuario. Los servidores, por su parte desempeñan un rol pasivo en la comunicación, ya que al iniciarse, esperan a que les lleguen las solicitudes de los clientes para terminar procesándolas y reenviándolas nuevamente al cliente.

Por lo general, los servidores aceptan conexiones de un gran número de clientes aunque a veces el número máximo de peticiones sea limitado. Normalmente no interactúan de forma directa con los usuarios finales. En el lado del cliente se utilizan lenguajes interpretados (scripts) de forma directa o bien a través de plugins como por ejemplo JavaScript con la finalidad de añadir elementos dinámicos a la interfaz del usuario. Por lo general, cada página Web se envía al cliente como un documento estático, aunque la secuencia de páginas que constituyen la aplicación Web ofrece al usuario una experiencia interactiva. Durante la sesión, el navegador Web, que actúa como cliente en toda aplicación Web, interpreta y muestra en pantalla las páginas.

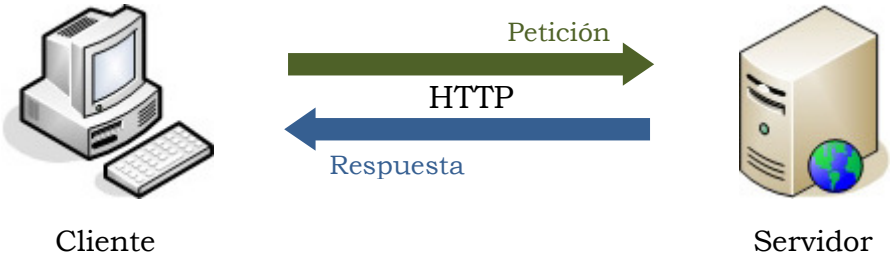


Figura 18 - Arquitectura Cliente-Servidor.

4.1.2. INTERFAZ

Las interfaces Web tienen una serie de limitaciones en la funcionalidades ofrecidas al usuario. En las aplicaciones de escritorio se tienen funcionalidades que no están soportadas por las tecnologías Web estándares como por ejemplo dibujar en la pantalla o arrastrar/soltar. Por esta razón, los desarrolladores Web suelen utilizan scripts en el lado del cliente para añadir más funcionalidades, especialmente para ofrecer un uso interactivo que no requiera recargar la página cada vez. La técnica de desarrollo Web conocida como AJAX ha sido creada para coordinar los lenguajes interpretados o scripts con las tecnologías existentes en el lado del servidor.

4.1.3. CONSIDERACIONES TÉCNICAS

Existen aplicaciones Web, escritas por ejemplo con HTML, CSS, DOM, …, que pueden causar problemas en el desarrollo y soporte de la propia aplicación Web. Esto suele ser debido a la falta de adhesión de los navegadores a dichos estándares, especialmente en versiones antiguas de Internet Explorer (navegador Web creado para Windows). Además,

la personalización de las características de la interfaz (como cambiar en las fuentes el tamaño, color, tipo, ...) por parte de los usuarios puede crear algunas inconsistencias en la propia aplicación Web.

4.1.4. ESTRUCTURA

Las aplicaciones Web generalmente se dividen en tres niveles o capas. En la primera capa, conocida como capa de la aplicación o del cliente, se encuentra el navegador Web. En la segunda capa, capa de negocio, se tiene el servidor Web de la aplicación, es decir, un motor capaz de usar alguna tecnología Web dinámica como puede ser PHP, JSP, ASP, Python, Ruby on Rails, ... Algunos de los servidores Web más importantes son Apache, IIS, Cherokee y Tomcat. Una base de datos constituye la tercera y última capa, conocida como capa de datos. El navegador Web envía peticiones a la capa intermedia que ofrece servicios valiéndose de consultas y actualizaciones a la base de datos y a su vez proporciona una interfaz de usuario.

Figura 19 - Estructura de una aplicación Web a 3 niveles.

Ventajas de las aplicaciones Web

- Se pueden realizar tareas sencillas sin necesidad de descargar ni instalar ningún programa.
- No ocupan espacio físico en el disco duro.
- Actualizaciones inmediatas: Como el software lo gestiona el propio desarrollador, cuando el usuario se conecta se está utilizando la última versión lanzada.
- Bajo consumo de recursos: Dado que la mayor parte de la aplicación no se encuentra en el ordenador del cliente, muchas de las tareas que realiza el software no consumen recursos propios debido a que estos se ejecutan desde otro ordenador.
- Son multiplataforma, es decir, que se pueden usar desde cualquier sistema operativo, lo único que hace falta es disponer de un navegador actualizado.
- Son portables, lo cual quiere decir que la aplicación es independiente del ordenador donde se utilice porque se accede a través de una página web (solamente es necesario disponer de acceso a Internet).
- La disponibilidad suele ser alta porque el servicio se ofrece desde múltiples localizaciones para asegurar la continuidad del mismo.

- Los virus no dañan los datos porque éstos están guardados en el servidor de la aplicación.
- El acceso al servicio y la compartición de datos por parte de varios usuarios es sencilla, ya que este acceso se realiza desde una única ubicación.
- Los navegadores ofrecen cada vez más y mejores funcionalidades para crear aplicaciones Web.

Inconvenientes

- Ofrecen menos funcionalidades que las aplicaciones de escritorio, aunque en la actualidad los navegadores están cada vez más preparados para mejorar este aspecto.
- La disponibilidad del servicio depende de un tercero, ya sea el proveedor de la conexión a Internet o bien el proveedor del enlace entre el servidor de la aplicación y el cliente.

4.1.5. LENGUAJES DE PROGRAMACIÓN

Existen muchos lenguajes de programación que pueden utilizarse para desarrollar aplicaciones Web: PHP, Java - con sus tecnologías Java Servlets y JSP, Javascript, Perl, Ruby, Python, C# y Visual Basic - con sus tecnologías ASP/ ASP.NET. Otros lenguajes o arquitecturas, que no son propiamente lenguajes de programación, como HTML o XML son muy utilizadas.

4.2. DISEÑO WEB

Para llevar a cabo el diseño de esta aplicación Web se han seguido ciertas pautas:

1. Planificación
2. Preparación de los contenidos
3. Realización de bocetos (mockups)
4. Maquetación

La planificación consiste en un período de análisis y reflexión en el que se decide cuál va a ser el objetivo del sitio Web, así como su propósito y el tipo de usuario al que va dirigido. ¿Es un sitio personal o comercial?, ¿su finalidad es entretener o proporcionar información a los usuarios?, ¿el objetivo del desarrollador es mostrar información detallada en forma de texto o impactar visualmente a los usuarios con imágenes y/o animaciones?. Una vez decididos los objetivos, se comenzó a planificar la estructura y los contenidos de la aplicación Web. Para ello es fundamental meditar todo a conciencia para conseguir minimizar posibles cambios que pudieran surgir durante el proceso de diseño y evitar así complicaciones.

Siempre es bueno consultar y estudiar sitios Web similares, obviamente nunca para copiar sino más bien para tomar buenas ideas y tratar de incorporar mejoras.

Una vez hecha la planificación se recomienda preparar un directorio concreto en el ordenador en el que se va a trabajar e introducir en él todo aquel material que va a ser necesario: imagen corporativa, animaciones, imágenes, fuentes de texto (tipografía), etc. En este caso se han incluido también las librerías JavaScript utilizadas, los estilos CSS, los iconos de los diferentes elementos a representar en el mapa (simbología), ... en diferentes carpetas dentro del directorio de trabajo. Una vez se tiene una idea clara de la Web a realizar se puede empezar a realizar bocetos sobre la apariencia gráfica de la misma, ya sean en papel o con cualquier programa de diseño. En este caso se realizaron varias pruebas con ideas diferentes.

Finalmente, una vez elegido el boceto definitivo se debe proceder a su maquetación, teniendo en mente los objetivos con los que se empezó el proceso de diseño Web.

4.3. ENTORNO DE TRABAJO

Para el desarrollo de esta aplicación web se ha utilizado la plataforma Eclipse, que es un programa compuesto por un conjunto abierto y extensible de herramientas útiles para un desarrollador de software; esto es lo que se conoce como Entorno de Desarrollo Integrado (IDE). Estos entornos cuentan con un editor de código, un compilador y un depurador. Se ha elegido Eclipse porque sirve como IDE de Java, posee muchas herramientas de desarrollo de software y es completamente gratuito. Si se desea información detallada sobre esta IDE viste la siguiente url:

http:// www.eclipse.org/

4.4. TECNOLOGÍA JSP

El lenguaje de programación Java se ha mejorado, ampliado y probado por una comunidad activa de muchísimos desarrolladores de software. Esta tecnología es extremadamente sencilla, independiente de la plataforma, de código abierto, eficaz y muy versátil. Por esta razón la tecnología elegida para crear esta aplicación Web ha sido JavaServer Pages (JSP). Una buena ventaja que se tiene al utilizar JSP es que se pueden usar todas las librerías de Java que se desee, aunque también ha sido clave en esta decisión el hecho de que todo el software necesario sea gratuito. De todas modos hay que tener en cuenta que todas las tecnologías que se utilizan en la actualidad como por ejemplo PHP, ASP o JSP mezclan la presentación con la lógica de negocios, lo cual hace que no sea una solución escalable desde el punto de vista de la mantenibilidad.

Recuerde que en la presentación se encuentran aquellos elementos con los que el usuario puede interactuar y que corresponden a la interfaz de usuario; mientras que en la componente lógica de negocios se establecen las distintas operaciones del sistema a realizar con los datos que provienen de la componente de datos (base de datos y sus mecanismos de acceso).

JSP es una tecnología que se utiliza para crear sitios Web dinámicos e interactivos basado en el lenguaje de programación Java. Sin JSP, la actualización de la apariencia del contenido de las páginas HTML estáticas debe hacerse a mano, aunque únicamente se desee cambiar una fecha o una imagen. Con JSP, el contenido puede hacerse dependiente de muchos factores como incluir la hora, la información proporcinada por el usuario, el historial del usuario que interactúa con el sitio Web e incluso el tipo de navegador Web del mismo. Esta capacidad es esencial para proporcionar servicios online en los que se puede ajustar cada respuesta al que hizo la petición, dependiendo de sus preferencias y requerimientos [18].

Un aspecto crucial de la prestación de servicios en línea significativos es que el sistema sea capaz de recordar los datos asociados con el servicio y sus usuarios. Por esa razón las bases de datos juegan un papel esencial en las páginas web dinámicas. Las JSP se construyen sobre una tecnología Java para crear contenido Web dinámico (servlets de Java) [19]. Los servlets de Java no son aplicaciones autónomas, se cargan en la memoria mediante un contenedor de servlet que funciona como servidor Web. Tomcat es el servidor Web de Java que, además de ejecutar el código, se encarga de recibir peticiones HTTP que provienen de navegadores Web y las pasa a los servlets [18]. Las páginas Web dinámicas, se generan en el servidor Web al ejecutar el script en el que se encuentran programadas. El resultado de esa ejecución es una salida HTML remitida al navegador.

Los siguientes pasos explican cómo el servidor Web crea la página Web:

1. El navegador envía una petición HTTP al servidor Web.
2. El servidor Web reconoce que la petición HTTP es para una página JSP y la reenvía a un motor JSP.
3. El motor JSP carga la página JSP y la convierte en un servlet de Java.
4. El motor JSP compila el servlet en una clase ejecutable y reenvía la solicitud original a otra parte del servidor Web llamada motor servlet. El motor JSP sólo transforma la página JSP a Java y recompila el servlet si detecta cambios en la página JSP desde la última petición.
5. El motor servlet carga la clase servlet y la ejecuta. Durante la ejecución el servlet crea un archivo HTML de salida que el motor envía al servidor Web dentro de una respuesta HTTP.
6. El servidor Web reenvía la respuesta HTTP al navegador.
7. El navegador actúa con la página HTML generada dinámicamente del mismo modo que si se tratara de una página estática.

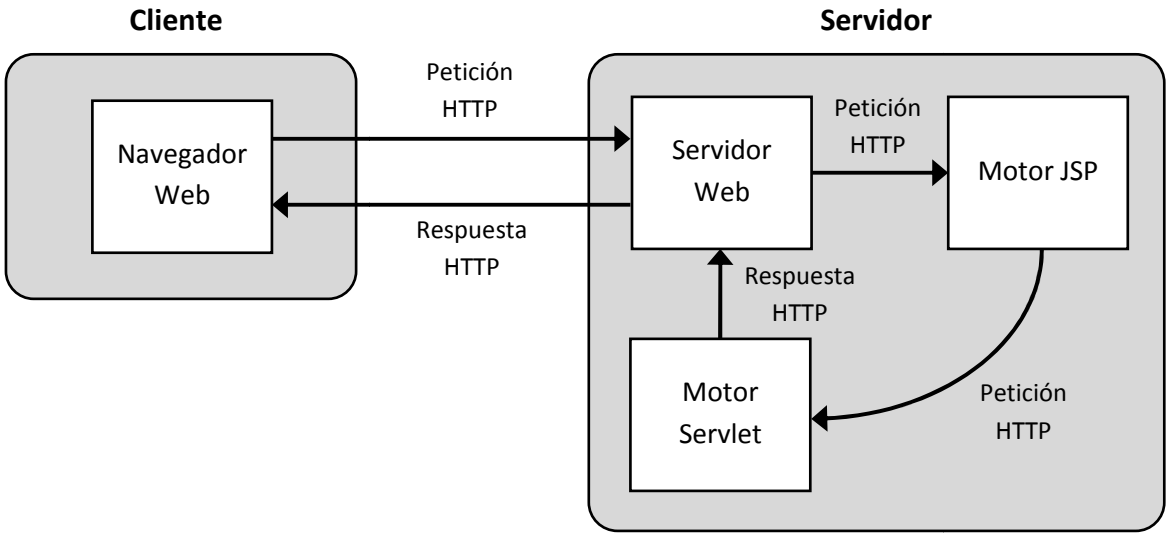


Figura 20 - Procesamiento JSP.

4.5. LENGUAJES DE PROGRAMACIÓN Y LIBRERÍAS UTILIZADAS

La Web era bastante aburrida en sus comienzos, ya que las páginas Web eran construídas en HTML y únicamente podían mostrar información. Lo más interactivo que había eran los enlaces entre diferentes páginas. Hoy en día, la mayoría de los sitios Web son casi tan sensibles como los programas existentes en un ordenador, reaccionando inmediatamente a cada clic del ratón. Todo esto se debe en gran medida a Javascript y a jQuery [20].

Para diseñar y desarrollar el prototipo de la presente aplicación Web se ha utilizado el lenguaje de programación JavaScript. El motivo principal por el que se ha usado este lenguaje se debe a que la librería más utilizada y avanzada en la creación de geoportales Web es OpenLayers (librería gratuita y de código abierto JavaScript). En este caso, la aplicación Web ha sido mayoritariamente construida en JavaScript, considerando además tres librerías propias de este lenguaje como son OpenLayers, ExtJS y GeoExt. El resto de la aplicación ha sido elaborada mediante el uso de hojas de estilo en cascada (CSS) y HTML.

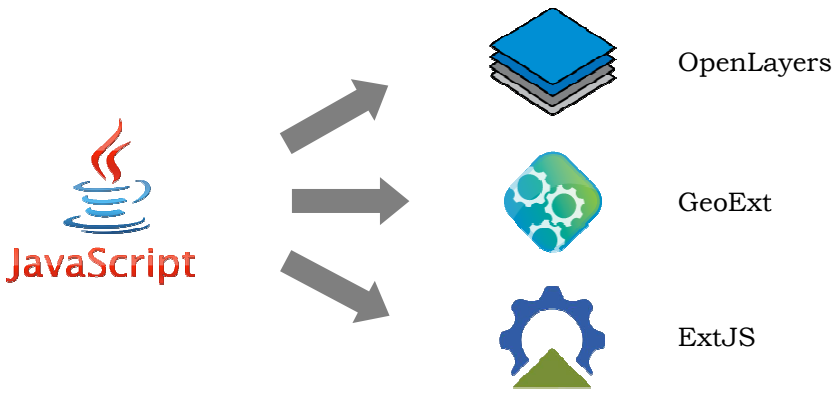


Figura 21 - Lenguaje de programación y librerías utilizadas.

- **OpenLayers**: Muestra mapas interactivos en los navegadores Web.
- **GeoExt**: Proporciona una base para enriquecer los geoportales Web.
- **ExtJS**: Útil para desarrollar aplicaciones Web interactivas usando tecnologías como AJAX, DHTML y DOM.

4.5.1. JAVASCRIPT

JavaScript es un lenguaje de programación, interpretado y ligero con capacidades rudimentarias orientadas a objetos, que permite al desarrollador potenciar el fichero HTML con animaciones, interactividad y efectos visuales dinámicos [20]. JavaScript, a pesar de lo que muchos pueden pensar, no es un lenguaje de programación Java simplificado, ambos lenguajes no están relacionados. La similitud en los nombres se debe únicamente a temas de marketing [21].

Este lenguaje puede hacer que las páginas Web sean más útiles suministrando información de forma inmediata. Por ejemplo, una página correspondiente al carrito de compras desarrollada en JavaScript puede mostrar inmediatamente el costo total, más impuestos y gastos de envío, en el momento en que un visitante selecciona un producto que desea comprar. JavaScript puede producir un mensaje de error inmediatamente después de que alguien trate de enviar un formulario Web con campos de información necesaria vacíos. JavaScript también permite crear interfaces dinámicas, atractivas e interactivas. Por ejemplo, con JavaScript es posible transformar una página estática de imágenes en miniatura en una presentación animada de diapositivas; o incluso añadir algo útil y atractivo, como información sobre herramientas emergentes que proporcionan datos adicionales para los diferentes detalles existentes en una página Web, etc.

Otra característica de JavaScript es su inmediatez. Permite que las páginas web respondan instantáneamente a acciones como hacer clic en un enlace, rellenar un formulario, o simplemente mover el ratón por la pantalla. Debido a que no depende de la constante de carga y recarga de las páginas Web, JavaScript permite crear páginas Web que se sienten y actúen más como programas de escritorio que como páginas Web.

Para comprender mejor la diferencia entre JavaScript y JSP hay que recordar dos cosas: en primer lugar, el código JavaScript es generado por el cliente Web o navegador después de que el servidor Web haya enviado la respuesta HTTP al navegador; y en segunda lugar, las JSP son ejecutadas por el servidor Web antes de que éste envíe la respuesta HTTP. Por todo esto, se dice que JavaScript es una tecnología aplicada al cliente mientras que las JSP son una tecnología aplicada al lado del servidor y su código es procesado por el éste antes de que llegue al cliente [19].

4.5.2. OPENLAYERS

En los últimos años, ha estallado la aceptación de los mapas web interactivos. Anteriormente, la creación de mapas interactivos estaba reservada a grandes empresas o profesionales adinerados. Pero ahora, con la llegada de servicios gratuitos como Yahoo! y Google Maps, la cartografía online está disponible a todos de forma fácil. Hoy en día, cualquiera puede producir un mapa Web con pocos e incluso nulos conocimientos en geografía, cartografía o programación [22]. Se supone que los mapas Web son rápidos, precisos y fáciles de usar. Existen pocas herramientas que son de utilidad para llevar a cabo todas estas expectativas. OpenLayers es un ejemplo de este tipo de herramienta. El mapeo o cartografía Web es el proceso correspondiente al diseño, implementación, generación y entrega de mapas en la World Wide Web y sus productos. OpenLayers proporciona un potente, pero fácil de usar, kit de herramientas puro JavaScript y HTML (sin necesidad de involucrar plugins de terceros) para crear rápidamente mapas web. Con Openlayers, los desarrolladores pueden implantar fácilmente su propio mapa web mashup utilizando WMS, Google Maps, entre otros [22].

OpenLayers fue realizado como software de código abierto por Metacarta Labs en 2006, y desde 2007 pasó a formar parte del proyecto Open Source Geospatial Foundation (OSGeo). OpenLayers es una librería de cartografía situada en la parte del cliente que proporciona una API para construir aplicaciones geográficas basadas en la Web ricas similares a Google Maps y Bing Maps. La librería estaba originalmente basada en el prototipo framework de Javascript.

La última versión estable de OpenLayers que se encuentra disponible es la 2.12, aunque está previsto que en breve se lance la versión 3. La nueva versión de esta librería se centrará en las mejoras de rendimiento, construcciones más ligeras, componentes visuales más bonitos, una API mejorada, etc.

Algunos de los aspectos más destacados de OpenLayers 3 son:

- **WebGL** parece que traerá capacidades 3D y un mayor rendimiento de todas las necesidades cartográficas para los navegadores Web más recientes.
- **Cesium**: La comunidad OpenLayers también integrará la nueva librería Cesium para permitir plenas capacidades de giro del globo en 3D.
- **Compilador de cierre**: Al utilizar el compilador de cierre, los desarrolladores de las aplicaciones podrán crear librerías más pequeñas y rápidas, facilitando el uso del amplio kit de herramientas de OpenLayers.
- **Un nuevo código base**: Esto ofrece la oportunidad de evitar algunas formas torpes de hacer las cosas en OpenLayers. El equipo también creará nuevos diseños de la API, que serán más accesible para todos.
- **Documentación de alta calidad**: En esta versión se incluirá documentación con nuevos y actuales ejemplos así como diseños predefinidos.

OpenLayers reside en la parte cliente. Una de las tareas principales que cumple el cliente One of the main tasks the client accomplishes es conseguir imágenes de mapa de un servidor de mapas. Básicamente, el cliente tiene que pedir a un servidor de mapas lo que se desea visualizar. El cliente tiene que hacer nuevas peticiones al servidor cada vez que el cliente navega o hace zoom en el mapa debido a que el cliente está pidiendo ver algo diferente. OpenLayers maneja todo esto para el usuario a través de llamadas AJAX a un servidor de mapas [22].

Un servidor de mapas proporciona el propio mapa. Hay muchos tipos diferentes de servidores de mapas. Todos estos servidores permiten a los clientes especificar la región del mapa que quieren ver (mediante el envío de una petición), y a continuación dichos servidores envían una respuesta que contiene la imagen del mapa. OpenLayers permite a los clientes utilizar tantos servidores de mapas diferentes en cualquier tipo de combinación como deseen. Esta librería no es un servidor de mapas web, sino que sólo consume datos de ellos. Hay numerosos servidores de mapas web gratuitos de código abierto disponibles que se encuentran alojados remotamente o que son fáciles de configurar, como Geoserver o Mapserver [22]. OpenLayers soporta GeoRSS (un estandar emergente para la codificación de la localización como parte de un agregado web), KML, GML y GeoJSON (formato abierto para la codificación de una variedad de estructuras de datos geográficos) y datos del mapa de cualquier fuente mediante la utilización de estándares del OGC como WMS o WFS:

Software:
- UMN MapServer
- MapGuide Open Source
- GeoServer
- ArcGIS Server
- ka-Map

Software as a Service:
- Google Maps
- OpenStreetMap
- Virtual Earth
- Yahoo! Maps
- World Wind servers

Con esta librería se puede interactuar con servicios SIG externos como Google Maps, Bing Maps, Yahoo Maps u OpenStreetMap a través de ArcGIS Server, GeoServer, MapServer, etc. Además, es posible crear mapas interactivos, visualizar información espacial/geográfica, incluir y superponer distintos tipos de capas de información así como editar información espacial.

Cuando un cliente trabaja en aplicaciones web cartográficas lo primero que debe hacerse es establecer el propio mapa. El mapa es el núcleo de la aplicación y es donde los clientes añadirán y visualizarán la información. Programar con OpenLayers está muy relacionado al código de escritura de Javascript así como también al de HTML. Por tanto, solamente se necesita un editor de texto para empezar a programar [23].

Inicialmente se crear un nuevo fichero html vacío para posteriormente insertar el siguiente código en él:

```
<!DOCTYPE html>
<html>
	<head>
		<title>Creating a simple map</title>
		<meta http-equiv="Content-Type" content="text/html;charset=UTF-8">

		<!-- Incluir la librería de OpenLayers -->
		<script type="text/javascript" src="http://openlayers.org/api/2.11/OpenLayers.js">
		</script>

		<style>
			html, body {width: 100%; height: 100%; margin: 0; padding: 0;}
		</style>

		<script type="text/javascript">
			function init() {
				// Creación del mapa
				var map = new OpenLayers.Map("mapa");
				// Creación de una capa ráster OSM y añadirla al mapa
				var osm = new OpenLayers.Layer.OSM();
				map.addLayer(osm);
				// Establecer el zoom a la máxima extensión posible
				map.zoomToMaxExtent();
			}
		</script>
	</head>

	<body onload="init()">
		<div id="mapa" style="width: 100%; height: 100%;"></div>
	</body>
</html>
```

Al ejecutar este código en cualquier navegador Web se muestra un mapa a pantalla completa con los controles de navegación básicos (ver figura 22).

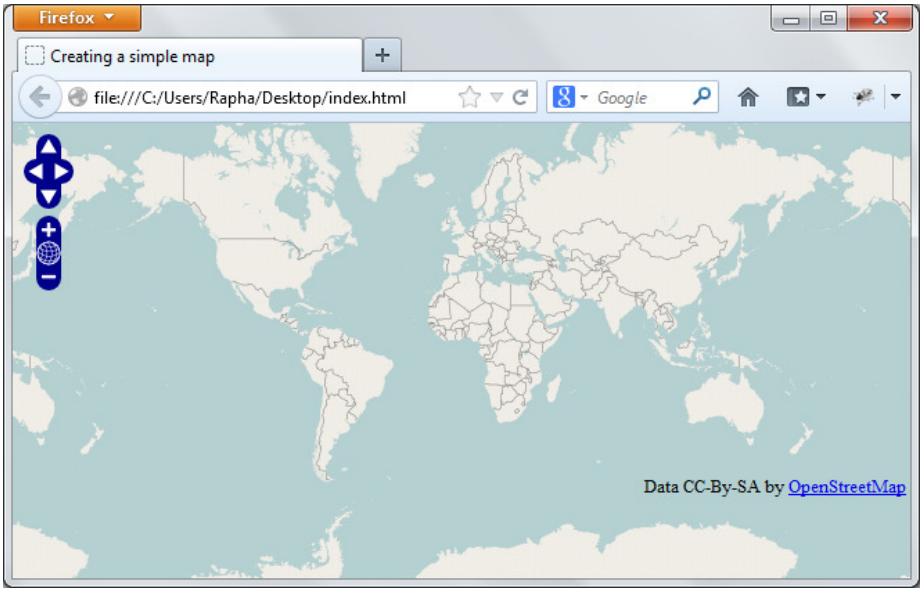
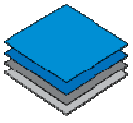


Figura 22 - Creación de un mapa simple con OpenLayers.

Si desea más información sobre esta librería visite el sitio Web oficial de OpenLayers:



http://www.openlayers.org/

4.5.3. EXTJS

OpenLayers carece de una interfaz de usuario rica. Para solucionar este hecho se ha contado con la librería ExtJS. La librería ExtJS comenzó como una extensión de la librería YUI (Yahoo User Interface) añadiendo todo aquello de lo que YUI carecía: una API fácil de usar y componentes (widgets) del mundo real [24]. ExtJS proporciona una interfaz de usuario rica y fácil de utilizar, al igual que sucede en una aplicación de escritorio. Esto permite que los desarrolladores Web se centren en la funcionalidad de las aplicaciones Web en vez de las advertencias técnicas. ExtJS puede funcionar junto a otras librerías de JavaScript mediante el uso de adaptadores. ExtJS hace que el desarrollo de aplicaciones Web sea sencillo gracias a la interacción entre el usuario y el navegador vía EventManager, respondiendo a los clics del ratón, eventos controlados en un navegador (como redimensionar la ventana o efectuar cambios en el tamaño de las fuentes). Además, ExtJS se comunica con el servidor en un segundo plano sin necesidad de refrescar la página. Esto permite pedir o subir datos desde o hacia nuestro propio servidor Web utilizando tecnologías como AJAX y procesar la respuesta en tiempo real.

Por lo general, la librería ExtJS debe usarse en un sitio Web que requiera de un alto nivel de interacción con el usuario, algo más complejo que el típico sitio Web. Por ejemplo, un sitio web que requiere de procesos y un flujo de trabajo. ExtJS dispone de una serie de componentes (widgets) que pueden ser incluidos en una aplicación Web. Estos componentes ya están preparados para manejar las complejidades de cada navegador existentes en el mercado, sin necesidad de efectuar cambio alguno. Por tanto, la apariencia de cada uno de estos componentes se muestra exactamente igual independientemente del navegador que se esté usando. Estos componentes son:

- Menús
- Barras de herramientas
- Árbol de datos
- Pestañas
- Paneles divisibles en secciones
- Sliders
- Gráficos
- Cuadros y áreas de texto
- Campos numéricos
- Campos para fechas
- Combos
- Radiobuttons y checkboxes
- Editor HTML
- Elementos de datos (modos de sólo lectura, datos ordenables, columnas bloqueables, etc.)

Además, gracias a esta librería es posible añadir interctividad a las páginas Web, como:

- Cuadros de diálogo
- quicktips

Los quicktips muestra mensajes de validación así como información sobre ciertos campos.

Todo lo necesario para usar esta librería puede descargarse libremente en:

http:// www.extjs.com/products/js/download.php

El kit de desarrollo Software (SDK) de ExtJS contiene un montón de ejemplos útiles, la API de referencia y los recursos que ExtJS necesita para funcionar correctamente.

Para incluir ExtJS en las páginas de un portal Web es necesario referenciar los ficheros de la librería. Para ello, se necesita incluir algunos ficheros proporcionados por el SDK en la parte de la cabecera (HEAD) de la página Web.

```
<!DOCTYPE html>
<html>
	<head>
		<title>Incluir la librería ExtJS</title>
		<link rel="stylesheet" type="text/css" href="lib/extjs/resources/css/ext-all.css"/>
		<script src="lib/extjs/adapter/ext/ext-base.js"></script>
		<script src="lib/extjs/ext-all-debug.js"></script>
	</head>
	<body>
		<!-- Nada en el cuerpo -->
	</body>
</html>
```

La ruta de los ficheros ExtJS debe ser la correcta y es relativa a la localización de cada fichero HTML. Estos ficheros deben ser incluidos en el orden especificado. Un fichero CSS puede incluirse después del fichero ext-all.css para customizar la apariencia de la interfaz de usuario. Como puede verse, se han incluido 3 ficheros que requiere la librería ExtJS en la página. Estos son:

- **ext-all.css**: Un fichero hoja de estilo que controla la apariencia y la alimentación de los componentes de ExtJS. Este fichero siempre debe incluirse sin ningún tipo de modificación. Cualquier cambio al fichero CSS impediría futuras mejoras. Si se decide efectuar algún tipo de ajuste sobre la apariencia debería incluirse otro fichero CSS después del fichero ext-all.css que sobreescriba todo aquello que se desee modificar.
- **ext-base.js**: Este fichero proporciona la funcionalidad básica de ExtJS. Es la base sobre la que ExtJS construye sus capacidades, y proporciona la interfaz para el entorno del navegador. Este es el fichero que se cambiaría si se quisiera utilizar otra librería, como jQuery, junto con ExtJS.
- **ext-all-debug.js**/**ext-all.js**: Todos los componentes (widgets) de la librería residen en este fichero. La versión de depuración se debe utilizar siempre durante el desarrollo, y luego cambiada por la versión de producción.

Una vez se ha añadido la librería a la página Web, el desarrollador puede empezar a programar el código que la utiliza.

En 2010 ExtJS se fusionó con JQTouch y Raphaël dando lugar a una nueva organización llamada **Sencha**. Para más información sobre la librería ExtJS se invita a consultar el sitio Web oficial:

http://www.sencha.com/

Para desarrollar el prototipo de la aplicación Web para el presente trabajo fin de máster, en lo concerniente a la librería ExtJS se han consultado principalmente dos libros: Learning Ext JS 3.2 [24] y ExtJS 4 Web Application Development Cookbook [25]; además de algunas páginas Webs con ejemplos prácticos como:

http://dev.sencha.com/deploy/ext-3.4.0/examples/

4.5.4. GEOEXT

GeoExt combina los controles geoespaciales de OpenLayers con los componentes de interfaz de usuario de ExtJS en un framework que permite crear aplicaciones SIG similares a las de escritorio, pero en un navegador. GeoExt es un framework (kit de herramientas), basado en las librerías OpenLayers y ExtJS, que incluye componentes estándares de interfaz de usuario para la construcción de aplicaciones geográficas Web con la apariencia y funcionalidad de las aplicaciones SIG de escritorio. GeoExt reúne la parte geoespacial de OpenLayers con parte correspondiente al aspecto de la interfaz de usuario de ExtJS para ayudar a construir potentes aplicaciones SIG de estilo escritorio en la web con JavaScript. GeoExt está disponible bajo licencia BSD y está soportado por una creciente comunidad de individuos, empresas y organizaciones. Por motivos de licencia, la librería ExtJS no puede ser incluida en la descarga de GeoExt, así que para utilizar la librería GeoExt en una aplicación Web requiere de un proceso de varios pasos:

1. Descargar la versión de desarrollo GeoExt desde la página de descargas.
2. Descargar la versión 2.10 o alguna otra posterior de OpenLayers desde http://www.openlayers.org/.
3. Descargar la última versión de Ext 3.x desde el sitio web ExtJS.
4. Situar ambas librerías descomprimidas en un directorio habilitado por el servidor Web que se utilice, en este caso Tomcat.

El equipo GeoExt recomienda utilizar, en los entornos de producción, versiones comprimidas y reducidas de GeoExt y ExtJS para optimizar el tamaño de descarga de la página. Todo mapa cartográfico debe incluir una leyenda, la cual proporciona la clave para poder interpretar los símbolos utilizados en él. Esta ha sido la razón principal por la que se ha utilizado esta librería, ya que permite la inclusión de un panel a modo de leyenda que facilita la visualización de las capas activas en el mapa con su correspondiente simbología.

EJEMPLO BÁSICO

El siguiente fragmento de código corresponde a la construcción de una página Web simple que sólo incorpora un mapa de navegación interactivo.

1. Lo primero que hay que hacer es incluir en la cabecera (HEAD) de una página HTML en blanco las librerías de OpenLayers, ExtJS y GeoExt en el orden correspondiente.
2. Es necesario crear un elemento en la página Web con su atributo identificador id que en este caso ha sido establecido como gxmap. Se utilizará este atributo para vincular un componente GeoExt al elemeno div.
3. Adjuntar un objeto MapPanel al elemento div con algo de código JavaScript.

```
<!DOCTYPE html>
<html>
      <head>
            <!--Paso 1-->
            <title>Página GeoExt básica</title>
            <script src="ext-base.js" type="text/javascript"></script>
            <script src="ext-all.js"  type="text/javascript"></script>
            <link rel="stylesheet" type="text/css" href="ext-all.css"></link>
            <script src="OpenLayers.js" type="text/javascript"></script>
            <script src="GeoExt.js" type="text/javascript"></script>
            <link rel="stylesheet" type="text/css" href="geoext-all-debug.css"></link>

            <!--Paso 3-->
            <script type="text/javascript">
                Ext.onReady(function() {
                    var map = new OpenLayers.Map();
                    var layer = new OpenLayers.Layer.WMS(
                        "Global Imagery",
                        "http://maps.opengeo.org/geowebcache/service/wms",
                        {layers: "bluemarble"}
                    );
                    map.addLayer(layer);
                    new GeoExt.MapPanel({
                        renderTo: 'gxmap',
                        height: 400,
                        width: 600,
                        map: map,
                        title: 'Página GeoExt básica'
                    });
                });
            </script>
      </head>
      <body>
            <!--Paso 2-->
            <div id="gxmap"></div>
      </body>
</html>
```

El resultado visual tras ejecutar el código anterior en un navegador se muestra en la siguiente figura. A partir de aquí, hay una amplia variedad de opciones disponibles para hacer aplicaciones de mapas personalizadas e interactivas con GeoExt.

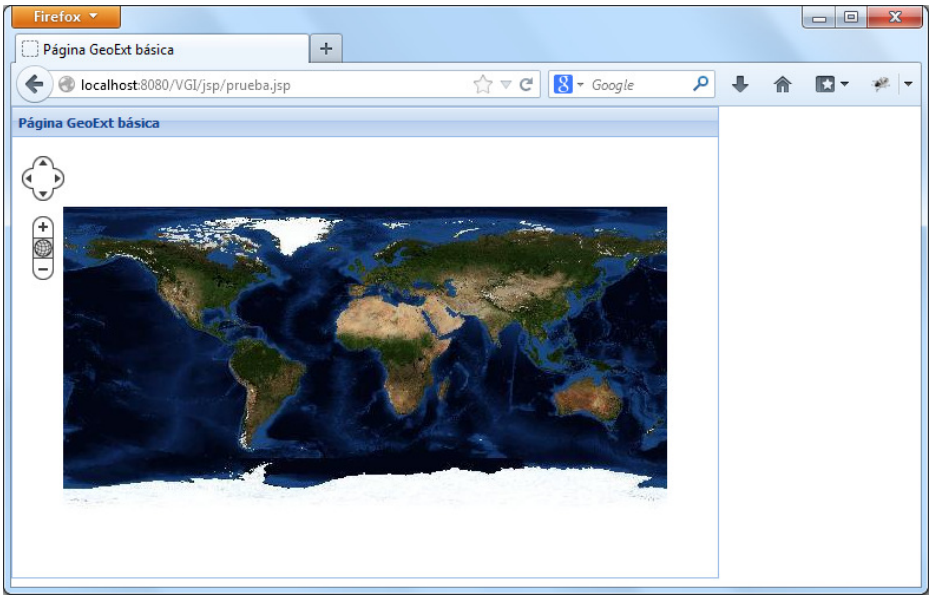


Figura 23 - Ejemplo sencillo de un panel de mapas básico con GeoExt.

Para obtener más información se recomienda consultar la API de referencia, los tutoriales y los ejemplos.

GeoExt	*http://www.geoext.org/*
Tutoriales	*http://www.geoext.org/tutorials/index.html*
Ejemplos	*http://www.geoext.org/examples.html#examples*
API de referencia	*http://www.geoext.org/lib/index.html*

4.6. PROXY

Antes de describir el geoportal Web creado en sí, es importante explicar la necesidad de incluir un proxy en la aplicación. En una red informática, un proxy es un programa que lleva a cabo una acción en representación de otro, de ahí que su nombre signifique intermediario en inglés. La finalidad más común de un proxy es la de servidor proxy, la cual consiste en interceptar las conexiones de red que un cliente hace a un servidor de destino, por motivos de seguridad, rendimiento, anonimato, etc. En este caso, debido a las normas de seguridad de JavaScript, no es posible recibir información sobre dominios remotos a través de un XMLHttpRequest (XHR). Incluso si dos servidores están corriendo sobre la misma máquina, pero en diferentes puertos se tiene esta limitación. XHR es una interfaz empleada para realizar peticiones HTTP y HTTPS a servidores Web. Para solventar esto, lo que hay que hacer es alojar un proxy en el servidor Web (Tomcat) y utilizar el OpenLayers.ProxyHost para direccionarlo. Hay algunas clases como WFS y GeoRSS que utilizan XHR para recibir sus datos. Si se está consultando a un servidor remoto, es necesaria la instalación de un proxy en alguna parte haciendo accesible esa máquina. Si la variable OpenLayers.ProxyHost no está establecida en una localización válida, las consultas serán enviadas directamente a los servidores remotos. En la mayoría de los casos, el resultado será una excepción de seguridad, aunque a veces esta

excepción sucede silenciosamente. En la configuración estándar de Apache Tomcat se recomienda situar el script en el directorio /usr/lib/cgi-bin/. Instalado el proxy, hay que editar la variable OpenLayers.ProxyHost ya mecionada haciéndola coincidir con la URL:

OpenLayers.ProxyHost = "/cgi-bin/proxy.jsp?url=";

Si la instalación no es correcta, al ejecutar la página Web se mostrará un mensaje de error 404, tanto si el proxy no está en la localización correcta como si el servidor Web no está configurado de forma adecuada. Una vez copiado el proxy en el servidor, no hay que olvidar editar el array con los hosts (anfitriones) permitidos.

4.7. PROTOTIPO WEB IGV

El prototipo de aplicación Web orientado a la información geográfica voluntaria creado, como todo geoportal Web de calidad que pueda haber en la actualidad, está formado por tres paneles que son necesarios para manejar y visualizar información espacial. Estos paneles son el mapa, selector de datos y leyenda. Además de estos paneles, la página muestra un menú desplegable donde el usuario podrá llevar a cabo una serie de acciones como geolocalizarse, calcular distancias y áreas sobre el mapa, manejar los controles del mapa, seleccionar la capa base, acceder a diferentes bases de datos, etc. También aparece en la parte inferior del mapa una pequeña barra con importante información referente al mapa visualizado como es la escala, el sistema de proyección, las coordenadas del puntero y las unidades.

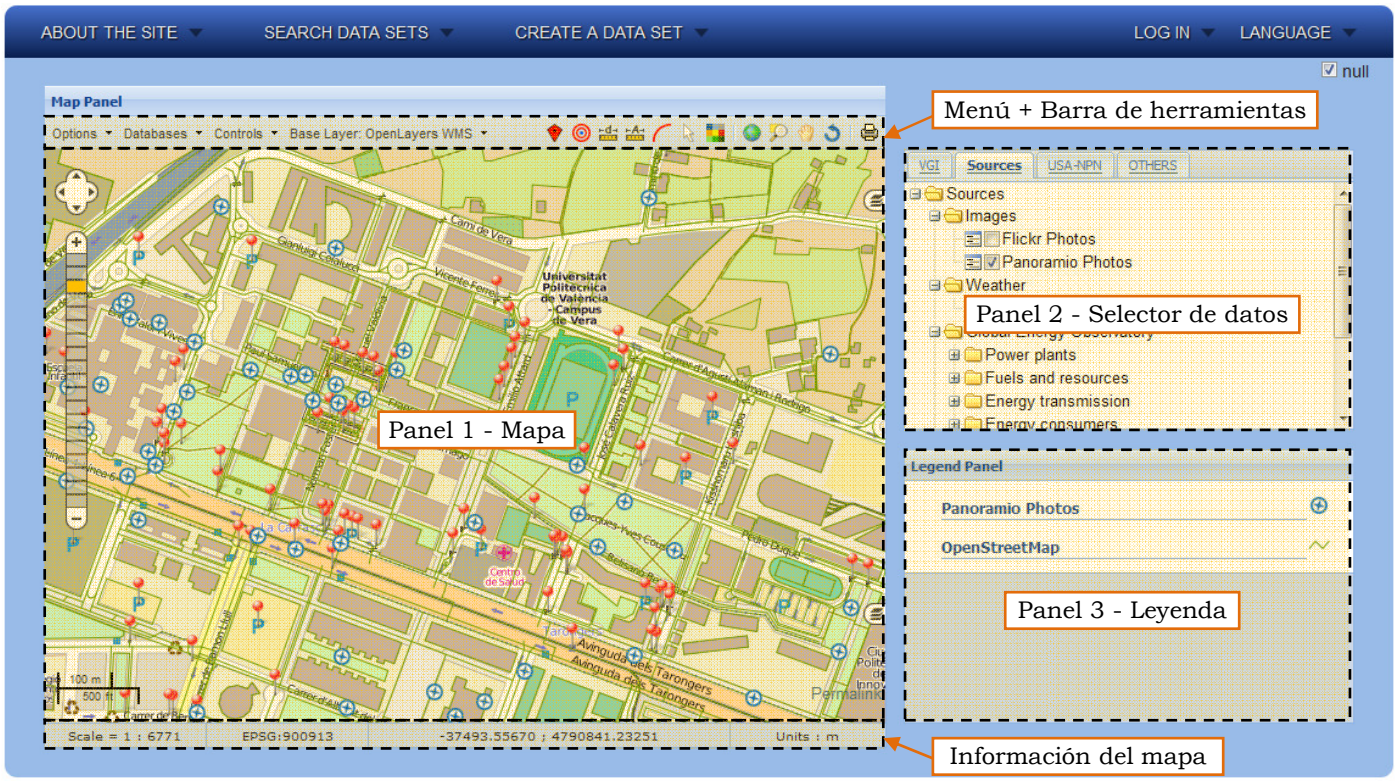


Figura 24 - Diseño del prototipo Web IGV.

4.7.1. DESCRIPCIÓN DE LOS COMPONENTES

MAPA

Un mapa es una representación geométrica plana a escala, simplificada y convencional, total o parcial de la superficie terrestre. Esta representación es una imagen incompleta, clara, legible y representativa de una realidad.

Un mapa de OpenLayers almacena información sobre la proyección por defecto, la extensión, unidades y demás propiedades del mapa. Dentro del mapa, los datos se muestran a través de las Capas. Una Capa es una fuente de datos – información sobre cómo OpenLayers debe pedir los datos y representarlos.

A continuación se describen los principales controles ofrecidos por OpenLayers, útiles para navegar e interactuar a través del mapa:

- PanPanel: Dibuja unos controles dentro del visor para desplazar el mapa.
- PanZoomBar: Dibuja una barra a modo de zoom que permite acercar o alejar el mapa.
- ScaleLine: Este control muestra una pequeña línea que representa la escala en el mapa actual. Por defecto se dibuja en la esquina inferior izquierda del mapa.
- LayerSwitcher: Permite ver la leyenda del mapa, mostrando las capas disponibles y permitiendo elegir las que se desean visualizar.
- OverviewMap: Mapa de referencia.
- Permalink: Crea un enlace a la visualización actual del mapa.
- Graticule: Muestra las líneas de longitud y latitud (cuadrícula) en el mapa.

Figura 25 - Controles del mapa facilitados por la librería OpenLayers.

SELECTOR DE DATOS

El selector de datos está formado por una serie de pestañas y en cada una de ellas existe un árbol de datos. Se ha utilizado cada una de las pestañas para diferenciar la información existente en ellas. Por su parte, cada árbol de datos contiene la información espacial correspondiente a las diferentes plataformas de información geográfica voluntaria estudiadas anteriormente. La información espacial seleccionada se mostrará tanto en el mapa como en la leyenda en forma de capa.

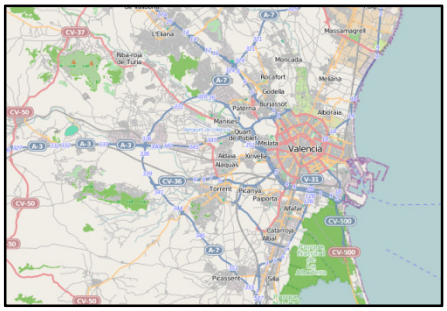
LEYENDA

La leyenda es muy útil ya que resume toda la información espacial activa en el mapa. En ella se refleja la simbología utilizada en el mapa, y proporciona la clave para su interpretación.

MENÚ Y BARRA DE HERRAMIENTAS

La barra de herramientas incluye a modo de menú los siguientes contenidos:

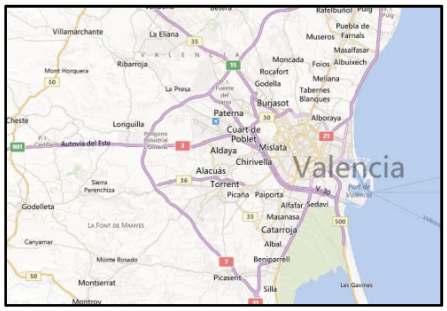
- Opciones (Acciones): geolocalización, cálculo de distancias y áreas, trazado de la línea o curva geodésica, impresión del mapa, ...
- Bases de datos: USA-NPN, GeoCommons, World Weather Online y Ushahidi.
- Controles del mapa: panZoomBar, scaleLine, OverviewMap, Permalink, layerSwitcher, cuadrícula, ...
- Listado de mapas base: OSGeo, OpenStreetMap, Google (mapa físico, calles, híbrido y satélite), Bing (calles, vista áerea con y sin etiquetas) y ESRI.


OpenStreetMap


Google Physical


Google Streets


Google Hybrid


Bing Road


Bing Aerial

Figura 26 - Algunos mapas base disponibles.

INFORMACIÓN DEL MAPA

Aquí se muestra información dinámica del mapa como son la escala, el sistema de proyección, las coordenadas del puntero sobre el mapa y las unidades de medición (metros / grados decimales).

4.7.2. ACCIONES IMPLEMENTADAS

En este prototipo de aplicación Web geográfica se han implementado una serie de acciones o funcionalidades típicas de este tipo de geoportales como son la geo-localización del cliente, el cálculo tanto de distancias como de áreas, ...

Otras acciones que son susceptibles de ser incluidas en todo visualizador cartográfico que se precie son aquellas destinadas a labores de edición como la selección, inserción, modificación y borrado de elementos (puntos, líneas, polígonos).

GEOLOCALIZACIÓN

En este momento existe un creciente mercado de aplicaciones relacionadas con la geolocalización o georreferenciación. Este mercado es perfectamente viable ya que los usuarios disponen de la tecnología necesaria para acceder a estos servicios. La geolocalización o georreferenciación es un vocablo que hace referencia al posicionamiento con el que se define la localización de un objeto espacial en un sistema de coordenadas y datum determinado.

Para obtener la posición de un usuario basándose únicamente en la dirección IP de un dispositivo conectado a internet es necesario usar algún tipo de mecanismo existente. Normalmente dichos mecanismos están basados en grandes bases de datos que relacionan direcciones IP y localizaciones. Las principales fuentes que suministran la información a estas bases de datos son las entidades de asignación de direcciones de cada área geográfica: American Registry for Internet Numbers (ARIN), RIPE Network Coordination Centre (Europe, RIPE NCC), Asia-Pacic Network Information Centre (APNIC), Latin American and Caribbean Internet Address Registry (LACNIC) y African Network Information Centre (AfriNIC). Algunas de estas bases de datos se han enriquecido mediante técnicas de minería de datos y técnicas estadísticas mejorando así los resultados. En la actualidad la geolocalización IP esta integrada dentro de HTML5, de este modo los navegadores son capaces de geoposicionar a sus usuarios.

La librería OpenLayers incluye un control que facilita mucho la faena al desarrollador a la hora de implementar esta acción. La figura 27 muestra el resultado tras ejecutar esta acción, así el usuario queda georreferenciado en el centro del mapa identificador por un icono que indica su posición. Si se desea saber con más detalle cómo se ha hecho, consulte las páginas 147-152 del libro "OpenLayers CookBook" [23].

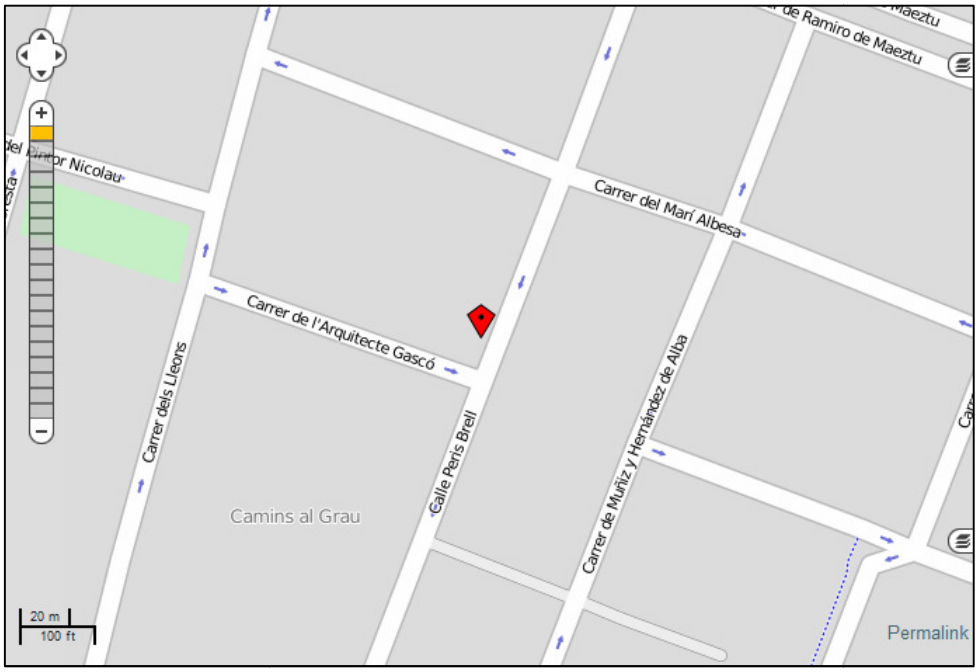


Figura 27 - Geolocalización del usuario.

CÁLCULO DE DISTANCIAS Y ÁREAS

Otro tipo de herramientas que son fundamentales en un proyecto SIG como este es la posibilidad por parte del usuario de efectuar mediciones sobre el mapa para obtener valores tanto de longitud como de superficie.

Figura 28 - Medición de la distancia (m) del puente de l'Assut de l'Or (Valencia).

La figura 28 muestra la medición de una distancia de forma clara. Obviamente para determinar el valor de una distancia son necesarios al menos dos puntos. Si el usuario

mide más de dos puntos consecutivos mediante simples clics con el ratón sobre el mapa, lógicamente el valor de la longitud irá incrementado. En caso de trazar una polilínea donde el punto inicial coincida con el punto final de la misma, el valor de la longitud total será lo que se conoce como perímetro del área dibujada. Para finalizar la medición es necesario efectuar doble clic sobre el último punto considerado. El valor de la distancia entre dos puntos se obtiene a partir de la siguiente fórmula:

$$d_{AB} = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$$

Por su parte la figura 29 muestra la medición efectuada de un área cualquiera. Recuerde que un polígono estará formado al menos por tres puntos. En ambos casos, para terminar la acción y obtener el valor de la medición correspondiente es necesario hacer doble clic con el ratón sobre el último punto considerado. El valor correspondiente al área de un polígono irregular se obtiene aplicando la siguiente fórmula:

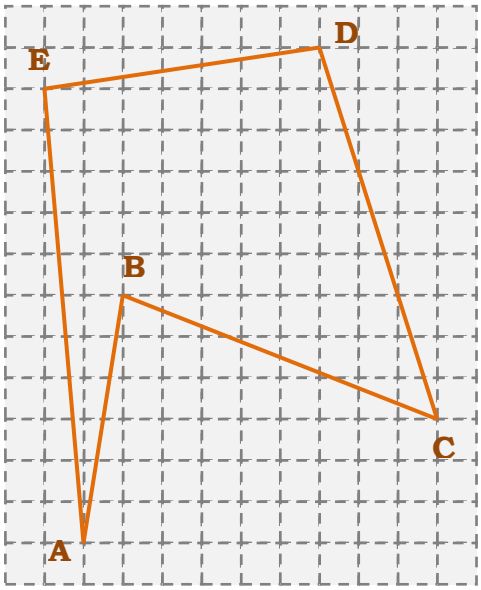
$$A(P) = \frac{1}{2} \cdot \left| \left(\sum_{i=1}^{i=n-1} (x_i \cdot y_{i+1} - y_i \cdot x_{i+1}) \right) + (x_n \cdot y_1 - y_n \cdot x_1) \right|$$

Figura 29 - Medición de la superficie (m²) de los jardines de Ayora (Valencia).

EJEMPLO DE CÁLCULO: A continuación se muestra un sencillo ejemplo donde queda demostrado el cálculo de la superficie de un polígono irregular cualquiera aplicando la formula anterior:

Las coordenadas de los vértices del polígono son:

A(-3,-2), B(-1,4), C(6,1), D(3,10) y E(-4,9).



El procedimiento consiste en crear una lista ordenada con las coordenadas X e Y de cada punto.

	X	Y
A	-3	-2
B	-1	4
C	6	1
D	3	10
E	-4	9
A	-3	-2

El segundo paso consiste en multiplicar la coordenada X de cada vértice por la coordenada Y del siguiente y sumar los resultados:

	X	Y	
A	-3	-2	-12
B	-1	4	-1
C	6	1	60
D	3	10	27
E	-4	9	8
A	-3	-2	

- 12 - 1 + 60 + 27 + 8 = 82

El tercer paso consiste en multiplicar la coordenada Y de cada vértice por la coordenada X del siguiente y sumar los resultados:

	X	Y	
A	-3	-2	2
B	-1	4	24
C	6	1	3
D	3	10	-40
E	-4	9	-27
A	-3	-2	

2 + 24 + 3 - 40 - 27 = -38

Una vez obtenidos los sumatorios de ambos productos cruzados se efectuará la resta de ambos y se dividirá por dos el valor adquirido, obteniéndose así el área o superficie del polígono irregular en unidades cuadradas.

$$A(P) = \frac{1}{2} \cdot |(82) - (-38)| = 60\, u^2$$

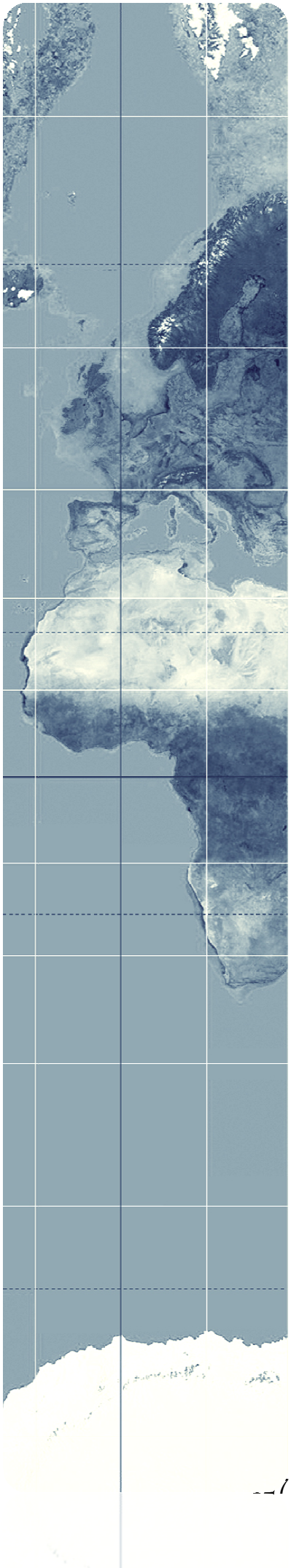
TRAZADO DE LA LÍNEA GEODÉSICA

El trazado de una línea recta entre dos puntos en un mapa representa el camino más corto ambos puntos en el plano proyectado, pero esa línea únicamente coincide con el camino más corto entre dichos puntos sobre la superficie de la Tierra si es menor a 500

kilómetros. La línea o curva geodésica es la distancia más corta entre dos de sus puntos medida sobre una superficie dada (elipsoide). Al aumentar la latitud y disminuir el radio del paralelo, la línea geodésica se irá curvando hasta llegar al punto más alto (polo norte) o más bajo (polo sur). Un ejemplo claro de esto son los trazados de las rutas de vuelo a varios destinos en todo el mundo. Estas rutas no son líneas rectas, sino que aparecen curvadas, porque representan la trayectoria de la línea geodésica sobre la superficie de la Tierra proyectada sobre un plano (mapa).

Figura 30 - Trazado de líneas geodésicas.

CONCLUSIONES, MEJORAS Y LÍNEAS DE INVESTIGACIÓN



CONCLUSIONES

Una vez realizada la aplicación Web y haber implementado las APIs de las diferentes infraestructuras de datos espaciales IGV estudiadas, se ha comprobado que aún queda mucho trabajo por hacer. Es necesario que se invierta más tiempo y dinero en este tipo de proyectos. Además es fundamental que se publiciten más, ya que muchas veces no sabemos ni que existen. Lo que queda claro es que OpenStreetMap es la plataforma más avanzada y popular de todas. La principal característica de OSM es que toda modificación realizada sobre el mapa se actualiza en tiempo real. Esto no sucede con las plataformas IGV restantes, ya que se actualizan con un desfase temporal variable que puede durar varios días. Este hecho es muy importante, sobre todo en situaciones catastróficas o de emergencia, en las que es primordial poder actuar rápidamente.

No hay que olvidar tampoco que el portal Web realizado es tan sólo un prototipo. Podría llegar a comercializarse pero no a escala global (mundial) como aparece en este documento, sino más bien a una escala mayor, es decir, enfocándo la aplicación a nivel estatal o mejor aún, a nivel regional (comunidades autónomas, provincias, ayuntamientos, ...), pudiendo sacar un mayor rendimiento a la aplicación.

POSIBLES MEJORAS

En la actualidad cuando se piensa en una aplicación Web no hay que olvidar o dejar de lado los sistemas móviles, ya que no son sólo el futuro sino que también el presente. En el último año 2012 se ha generado una auténtica fiebre por los terminales móviles 3G con pantalla táctil que ha revolucionado el mercado, tanto de la venta de teléfonos como de un suculento negocio de las aplicaciones para móviles que están convirtiendo a estos dispositivos en el ordenador del futuro gracias a varios factores: disponibilidad total, comodidad y portabilidad. Por tanto, habría que pensar también en realizar una aplicación móvil enfocada a la información geográfica voluntaria, que englobe las principales plataformas estudiadas, en los dos sistemas operativos principales (iPhone y Android).

La aplicación Web realizada se comporta como un visor cartográfico donde el usuario únicamente puede visualizar la información espacial que desea. Sin embargo, algunas de las APIs de las infraestructuras analizadas aquí permiten poder realizar tareas de edición sobre el mapa, siendo necesaria la autenticación del usuario.

Otra posible mejora que lamentablemente ha quedado pendiente ha sido la implementación de la API de Ushahidi, la cual habría enriquecido aún más el geoportal. Se espera que en un futuro cercano dicha API esté finalmente disponible y pueda ser incluida tanto en aplicaciones Web como esta como en aplicaciones para teléfonos de última generación.

Para tratar de evitar duplicidades se propone una estrategia basada en mapas de estados. Estos mapas indican al voluntario aquellas regiones en las que es más necesaria su ayuda, bien sea porque dichas regiones carecen de mediciones u observaciones, o bien porque ha transcurrido mucho tiempo desde la última observación efectuada, etc. Supóngase que se dispone de dos capas: una de polígonos y la otra de puntos. La primera hace referencia a aquellas zonas o regiones de un área que necesitan la ayuda de los voluntarios por algún motivo. La capa de puntos es la que contiene todas las observaciones realizadas en esa área. Por tanto, de lo que se trata aquí es de ayudar al voluntario a que elija la región sobre la que va a trabajar en base a unos supuestos.

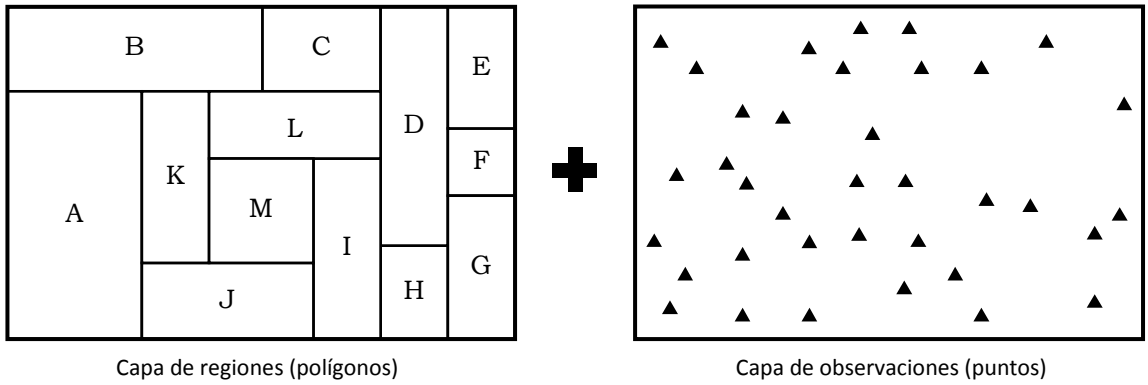


Figura 31 - Capas utilizadas para crear un mapa de estados.

La idea es que se visualice el mapa de regiones mediante simbología construyéndose así un mapa de estados, es decir, mostrando los polígonos en diferentes colores conforme a unos supuestos:

- Mapear el nº de observaciones (0, 23, 100, ...) en una escala continua o en clases. Dependiendo de ese valor, cada region aparecería representada de un color u otro según una clasificación (por ejemplo la siguiente: 0-10, 10-20, 20-30, ... >30).

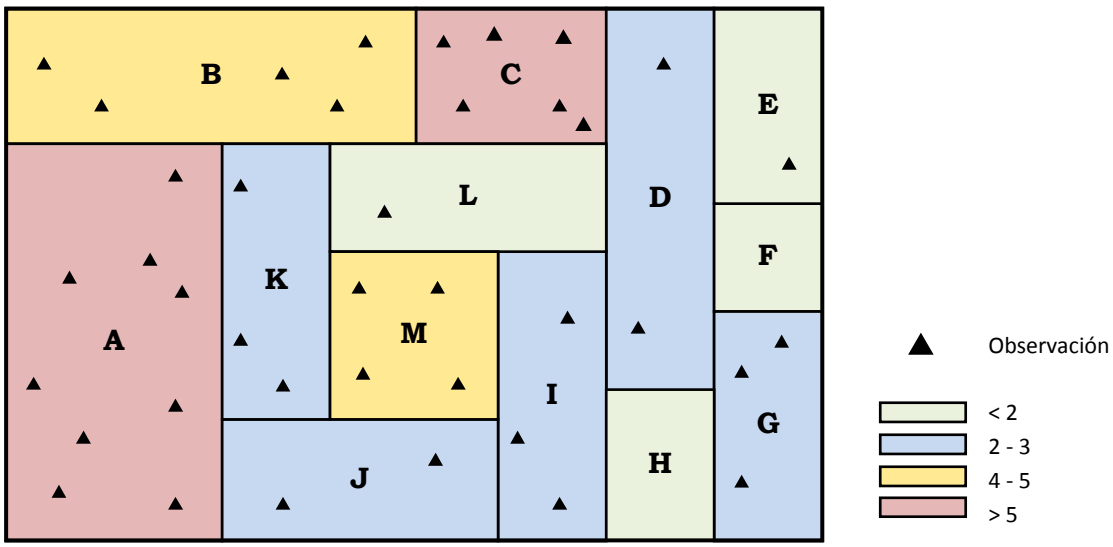


Figura 32 - Mapa de estados en función del número de observaciones.

Según la figura 32, las regiones donde sería más necesaria la colaboración del voluntario serían aquellas zonas en las que hay menos observaciones tomadas, en este caso los polígonos E, F, H y L.

- Comprobar el tiempo transcurrido desde la última observación. En este caso la clasificación podría ser: menos de una semana, menos de un mes, menos de 3 meses, menos de 6 meses, menos de un año y más de un año.

- Forecast: En el caso de la red fenológica podría considerarse la posibilidad de incluir el día de ocurrencia (predicción) de una fenofase. Para ello se calcularía la media de las observaciones del año anterior.

- La desviación típica o variabilidad de las mediciones puede ser calculada considerando los años anteriores. A mayor variabilidad o desviación en un polígono más "urgente" sería acudir a esa zona.

- Priorizar en términos de localización (geolocalización del usuario). De ese modo tendrán prioridad aquellas zonas que se encuentren más próximas al usuario. Esta opción no requiere la utilización de la capa que contiene las observaciones.

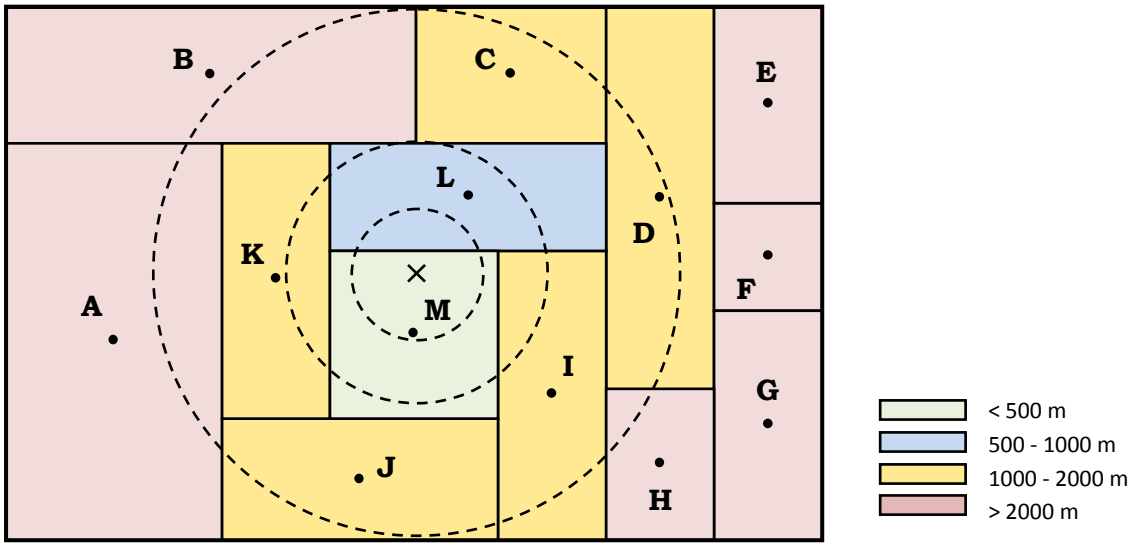


Figura 33 - Mapa de estados en función de la geolocalización del usuario.

Según la figura 33, las regiones donde sería más rápida la colaboración del voluntario serían aquellas zonas que se encuentran más cercanas a él, en este caso los polígonos M y L. Para realizar la clasificación se ha considerado el centro geométrico de cada polígono, aunque pueden considerarse otras posibilidades.

Para crear estos mapas de estados es necesario realizar tareas de geoprocesamiento. Uno de los principales propósitos del geoprocesamiento es permitir automatizar tareas SIG así como realizar tareas de análisis y modelado espacial. Para poder implementar labores de geoprocesamiento en este prototipo, sería necesaria la utilización de un servicio estándar del Open Geospatial Consortium (OGC) [1] cuya finalidad consiste en

describir y ejecutar una operación de geoprocesamiento a tarvés de un servicio estandarizando las peticiones y las respuestas para un servicio de procesamiento geoespacial en particular. Este servcicio se conoce como Web Processing Service (WPS) [26]. Dentro de la categoría de operaciones o procesos geoespaciales se incluye cualquier algoritmo, cálculo o modelo que opere sobre datos con referencia espacial, desde operaciones simples como por ejemplo un buffer hasta modelos complejos como el cálculo de zonas de inundabilidad.

El servicio WPS puede utilizar, como datos de entrada, tanto datos vectoriales como ráster, los cuales pueden estar ubicados en local o en la red. El cliente suministra estos datos al servicio y posteriormente podrá ejecutar la operación que desee sin necesidad de conocer detalles de implementación.

La iniciativa relativa a la implementación de servicios WPS que más destaca es la 52North. Esta iniciativa anunció hace algunos años el lanzamiento de la primera versión del cliente OpenLayers para servicios Web de geoprocesamiento. De este modo, al implementar este cliente en un sitio Web es posible realizar operaciones de geoprocesamiento utilizando un servicio Web estándar como WPS, es decir que gracias a este cliente sería posible procesar los supuestos anteriores a la hora de visualizar el mapa de estados. Este mapa vendría a ser una capa de polígonos en formato shape (*.shp), almacenado en un repositorio o base de datos local, que sería mostrado al usuario de la aplicación Web mediante el servicio WFS. Gracias a las operaciones de geoprocesamiento, llevadas a cabo por el estándar WPS, cada uno de los polígonos que constituyen ese mapa se visualizarían respecto a una simbología dada (que podría ser editable también por el usuario) en función al supuesto considerado.

Nota: El cliente WPS para OpenLayers de la iniciativa 52North no es capaz de realizar todas las operaciones de geoprocesamiento posibles, ni mucho menos. Para poder considerar los 5 supuestos anteriores sería necesario implementarlas en el código del cliente.

Si desea más información sobre el cliente WPS para OpenLayers visite la página oficial de la iniciativa 52North:

http://52north.org/

REFERENCIAS BIBLIOGRÁFICAS

1. Consortium, O.G. *OGC Web Site*. 2013; Available from: http://www.opengeospatial.org/.
2. Goodchild, M.F., *Citizens as sensors: the world of volunteered geography.* GeoJournal, 2007. 69(4): p. 211-221.
3. Zook, M., et al., *Volunteered geographic information and crowdsourcing disaster relief: a case study of the Haitian earthquake.* World Medical & Health Policy, 2012. 2(2): p. 7-33.
4. Luiz A. Manfré, E.H., Janaína B. Silva, Eduardo J. Shinohara, Mariana A. Giannotti, Ana Paula C. Larocca and José A. Quintanilha, *An Analysis of Geospatial Technologies for Risk and Natural Disaster Management.* ISPRS International Journal of Geo-Information, 2012: p. 20.
5. Kobiyama, M., et al., *Prevenção de desastres naturais: Conceitos básicos*2006: Organic Trading.
6. Nogueira, C.W., M.B. Gonçalves, and D. Oliveira. *O Enfoque da Logística Humanitária no Desenvolvimento de uma Rede Dinâmica para Situações Emergenciais: o Caso do Vale do Itajaí em Santa Catarina*. in *Anais do XXII Congresso de Pesquisa e Ensino em Transportes*. 2009.
7. Sorensen, M.B., Spada, M., Babeyko, A., Wiemer, S., Grünthal, G., *Probabilistic tsunami hazard in the Mediterranean Sea.* Journal of Geophysical Research, 2012. 117(B1): p. B01305.
8. Rodríguez, H., Quarantelli, E.L., Dynes, R.R., Andersson, W.A., Kennedy, P.A., Ressler, E., *Handbook of disaster research. The role of Geographic Information System / Remote Sensing in Disaster Management*. Vol. 643. 2007: Springer New York. 13.
9. McCann, D.G.C., Cordi, H.P., *Developing International Standards for Disaster Preparedness and Response: How Do We Get There?* World Medical & Health Policy, 2011. 3(1): p. 48-51.
10. Infrastructure, N.R.C.C.o.P.f.C.a.B.f.I.G.D., *Successful Response Starts with a Map: Improving Geospatial Support for Disaster Management*2007: National Academy Press.
11. Goodchild, M.F., *GIS and disasters: Planning for catastrophe.* Computers, Environment and Urban Systems, 2006. 30(3): p. 227-229.
12. Goodchild, M.F., *Citizens as Voluntary Sensors: Spatial Data Infrastructure in the world of Web 2.0.* Int. J. Spat. Data Infrastr. Res., 2007. 2: p. 24-32.
13. Sui, D.Z., *The wikification of GIS and its consequences: Or Angelina Jolie's new tattoo and the future of GIS.* Computers, Environment and Urban Systems, 2008. 32(1): p. 1-5.
14. Elwood, S., Goodchild, M.F., Sui, D.Z., *Researching volunteered geographic information: Spatial data, geographic research, and new social practice.* Ann. of the Association of American Geographers, 2012. 102(3): p. 571-590.
15. Roche, S., Propeck-Zimmermann, E., Mericskay, B., *GeoWeb and crisis management: issues and perspectives of volunteered geographic information.* GeoJournal, 2011: p. 1-20.
16. Fielding, R.T., *Architectural Styles and the Design of Network-based Software Architectures*, 2000, University of California. p. 180.
17. Mora, S.L., *Programación de aplicaciones web: historia, principios básicos y clientes web*2002, Universidad de Alicante. Departamento de Lenguajes y Sistemas Informáticos: Club Universitario.
18. Zambon, G., *Beginning JSP, JSF and Tomcat*. 2nd ed2012: Beginning Apress. 426.
19. Falkner, J., et al., *Desarrollo Web con JSP - Fundamentos*2002: Anaya Multimedia, S.A.
20. McFarland, D.S., *Javascript & jQuery - The missing manual*. 2nd Edicion ed2012, USA: O'Reilly Media. 538.
21. Flannagan, D., *JavaScript: The Definitive Guide*. 2nd Edition ed1997: O'Reilly.
22. Hazzard, E., *OpenLayers 2.10: Beginner's guide*. Create, optimize, and deploy stunning cross-browser web maps with the OpenLayers JavaScript web-mapping library2011: Packt Publishing.
23. Pérez, A.S., *OpenLayers Cookbook*. 60 recipes to create GIS web applications with the open source JavaScript library2012: Packt Publishing.
24. Shea Frederick, C.R., Steve 'Cutter' Blades, Nigel White, *Learning Ext JS 3.2*2010, Birmingham, UK: Packt Publishing Ltd.

25. Ashworth, S. and A. Duncan, *Ext JS 4 Web Application Development Cookbook*2012, Birmingham, UK: Packt Publishing. 488.
26. Peter Schut and A. Whiteside, *OpenGIS® Web Processing Service*, 2005. p. 86.