Procesos web geoespaciales: dificultades y alternativas en el caso del visor SACOSTA.

G. Frontera Borrueco [(1)], P. Balaguer Huguet [(1)], D. March Morlà [(2)], J. P. Beltran Amengual [(1)]

[(1)] ICTS SOCIB - Sistema d'Observació i Predicció Costanera de les Illes Balears, Parc Bit, Edifici Naorte, Bloc A 2n-3a, 07021 Palma, bfrontera@socib.es, pbalaguer@socib.es, jbeltran@socib.es.
[(2)] IMEDEA - Institut Mediterrani d'Estudis Avançats (UIB-CSIC), Carrer Miquel Marquès 21, 07190 Esporles (Mallorca), david@imedea.uib-csic.es.

RESUMEN

La ICTS SOCIB (Sistema de Observación y Predicción Costero de las Illes Balears) dispone de un visor cartográfico de la sensibilidad ambiental de la línea de costa las Illes Balears (SACOSTA). El visor está concebido como una herramienta de apoyo a la toma de decisiones en la costa frente a un posible vertido de hidrocarburos en el mar. El visor SACOSTA está desarrollado sobre tecnologías de la OpenGeo Suite: datos espaciales almacenados en una base de datos PostGIS; capas servidas mediante WMS por un servidor GeoServer; visor y composición de las capas utilizando GeoExplorer.

El visor incorpora una nueva funcionalidad para analizar la sensibilidad ambiental de una región determinada. Presentamos las diferentes opciones valoradas para dar respuesta al siguiente requerimiento: dada una región representada por un polígono, ¿cuál es la longitud de la línea de costa según la clasificación geomorfológica?

Por una parte, apuntamos cómo se podrían utilizar servicios WPS de GeoServer, una solución que sigue los estándares OGC. Por otra, presentamos un desarrollo ad hoc que utiliza Flask, un miniframework de Python. Esta aplicación web realiza consultas de forma directa a la base de datos PostGIS para extraer la información agregada solicitada y la devuelve en formato JSON.

Finalmente, presentamos cómo se ha integrado este servicio en GeoExplorer, donde se ha añadido una nueva herramienta de dibujo y desplazamiento de polígonos. La aplicación se conecta al servicio web y muestra gráficos de barras sobre el visor mediante el uso de la librería d3.js. A partir de estos gráficos, además, podemos filtrar elementos de las capas según el tipo de costa. También incorpora un generador de informes de la sensibilidad ambiental de la región seleccionada.

Palabras clave: *WPS, GeoExplorer, Python, Flask, PostGIS.*

ABSTRACT

ICTS SOCIB (Coastal Ocean Observing and Forecasting System located in the Balearic Islands) provides a web-based map viewer which displays cartographic data related to the environmental sensitivity of the coastline of the Balearic Islands (SACOSTA). The viewer is designed as a decision-making tool to support responses at the coast to potential oil spills in the sea. SACOSTA is built using using OpenGeo Suite technologies: spatial data stored in a PostGIS database, WMS layers served by GeoServer, viewer and composition of the layers using GeoExplorer.

The viewer has a new functionality to analyse the environmental sensitivity of a given region. We present different options evaluated to achieve the following requirement: given a region represented by a polygon, what is the length of the coastline upon its geomorphological classification?

On the one hand, we will explain how GeoServer WPS services could be used, a solution that would follow OGC standards. On the other hand, we introduce an ad hoc development using Flask, microframework for Python. This web application makes queries directly to the PostGIS database to extract aggregated information and returns it in JSON format.

Finally, we present how this service has been integrated into GeoExplorer, where a new drawing and dragging tool has been added. The viewer use the new web service and displays some plots using d3.js library. Through this plots, the application allows to filter features of the layer depending on their type of coast. The application also includes a report generator of the environmental sensitivity of the selected region.

Key words: *WPS, GeoExplorer, Python, Flask, PostGIS.*

INTRODUCCIÓN

La ICTS SOCIB (Sistema de Observación y Predicción Costero de las Illes Balears) dispone de un visor cartográfico de la sensibilidad ambiental de la línea de costa de las Illes Balears (SACOSTA) [1], concebido como una herramienta de apoyo a la toma de decisiones frente a un posible vertido de hidrocarburos.

Recientemente, se ha añadido una funcionalidad para analizar la sensibilidad ambiental de una región determinada. En este artículo presentamos las diferentes opciones que se valoraron para dar respuesta al siguiente requerimiento: dada una región representada por un polígono, ¿cuál es la longitud de la línea de costa según la clasificación geomorfológica?

Por una parte, apuntamos cómo se podrían haber utilizado servicios WPS de GeoServer. Por otra, se presenta la solución ad hoc que se ha desarrollado, basada en el desarrollo de servicios web con Flask, un miniframework de Python, que realiza consultas de forma directa a la base de datos PostGIS para extraer la información agregada solicitada y la devuelve en formato JSON.

Visor SACOSTA

Desde finales de los 80 la cartografía sobre sensibilidad se realiza mediante la utilización de bases cartográficas digitales mediante el uso de Sistemas de Información Geográfica (SIG), permitiendo de este modo la generación automática

de mapas de sensibilidad. La NOAA (National Oceanic and Atmospheric Administration) ha liderado la estandarización, mediante la elaboración de directrices metodológicas [2], así como la producción automática de cartografía concerniente a la sensibilidad de la línea de costa frente a un posible vertido de hidrocarburos.

SACOSTA (Sensibilidad Ambiental de la línea de Costa) es un visor que consta de tres componentes principales a) Clasificación geomorfológica de la costa (tipos de costa); b) Recursos biológicos (áreas costeras protegidas); c) Uso humano (infraestructuras, equipamientos, servicios, patrimonio histórico de la línea de costa). La clasificación del litoral Balear de acuerdo con su Sensibilidad Ambiental (SA) se ha basado en el estándar internacional propuesto por la NOAA para la elaboración de los Índices de Sensibilidad Ambiental (ESI; Environmental Sensitivity Index) [3]

El desarrollo de SACOSTA se caracteriza por cumplir con diferentes estándares OGC de interoperabilidad, en consonancia con la directiva INSPIRE. La consulta de los datos puede realizarse desde el propio visor, así como desde Google Earth (pestaña 'Metadatados' del visor), o mediante un servicio WMS. La descripción de los datos (metadatos) se encuentra disponible en el catálogo de metadatos del SOCIB.

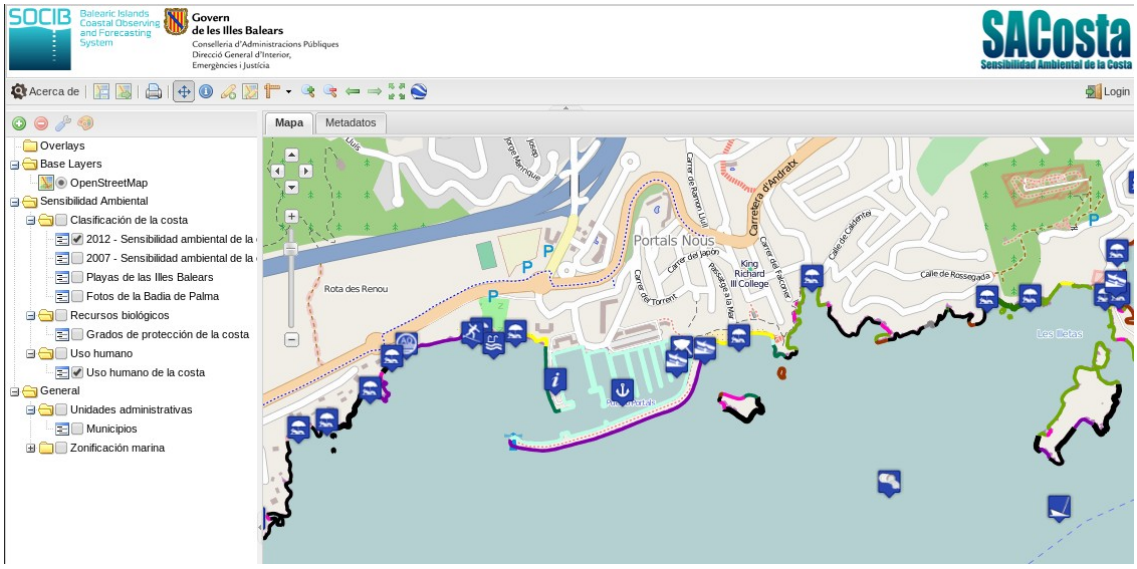

Figura 1: *Visor SACOSTA.*

El objetivo principal del establecimiento de la sensibilidad ambiental de la línea de costa es la determinación del comportamiento de los materiales que constituyen la línea de costa frente a una afección causada por hidrocarburos y/o sustancias contaminantes de naturaleza o comportamiento análogo. La determinación de la sensibilidad ambiental frente a estos tipos de contaminación constituye una herramienta de apoyo a la toma de decisiones, puesto que el conocimiento del comportamiento de los materiales frente a un episodio contaminante permitirá desarrollar protocolos de actuación adecuados para cada tipo de costa.

Las tareas de caracterización de la línea de costa se han realizado en un entorno de SIG, que permite un manejo ágil de la información beneficiando las tareas de gestión en situaciones de emergencia por contaminación marina en cuanto a tiempo de respuesta. La incorporación de información en un entorno SIG permite la consulta rápida de las bases de datos asociadas a la cartografía. Las ventajas de elaborar los datos correspondientes a la sensibilidad ambiental mediante la utilización de SIG son: 1) conocimiento, en tiempo real, de los tipos de costa afectados o potencialmente afectados por una afección contaminante, 2) longitud de línea de

costa afectada, 3) conocimiento del tipo de protección medioambiental de la línea de costa en caso de que exista, 4) conocimiento ágil de las principales características de las infraestructuras y equipamientos adecuados de cara la mitigación de la afección contaminante (p.e. tonelaje máximo de las grúas de los puertos, existencia de puesto de primeros auxilios, espacio de varadero disponible, etc.), 5) posibilidad de simplificar los tipos de costa, de manera rápida y ágil, en caso de que sea necesario (entre otros) y 6) facilitar el intercambio de información y su exposición en la web (internet: Web Map Services, SACOSTA).

SACOSTA está desarrollado sobre tecnologías de la OpenGeo Suite: datos espaciales almacenados en una base de datos PostGIS; capas servidas mediante WMS por un servidor GeoServer; visor y composición de las capas utilizando GeoExplorer [4].

NUEVA FUNCIONALIDAD

Recientemente el SOCIB realizó una revisión del visor SACOSTA en la que se decidió añadir una nueva herramienta con la que se pudiera seleccionar un área que pudiera ser potencialmente afectada por un vertido para obtener un informe sobre su sensibilidad ambiental.

Esta nueva funcionalidad responde a la experiencia previa del personal del SOCIB en la respuesta a catástrofes como la del hundimiento del buque Prestige [5] [6] o el del buque Don Pedro en aguas de Eivissa [7]. Entre las lecciones aprendidas, destacan la necesidad de disponer de un sistema operacional que ofrezca una respuesta rápida para minimizar el impacto de los vertidos sobre el medio ambiente marino [8]. Por una parte es necesario un buen estudio del clima marino y predicción de corrientes y oleajes. Por otra, se debe disponer de información acerca de la sensibilidad ambiental de la costa afectada según las predicciones de la deriva del vertido. De esta manera se pueden optimizar las tareas de lucha contra la contaminación. [9]

Así, la herramienta debía permitir lo siguiente:

a) Selección un área arbitraria en el visor, que potencialmente sería la afectada por un vertido.
b) Obtención de un análisis gráfico del tipo de costa.
c) Obtención de imágenes oblicuas de cada tipo de costa.
d) Generación de un informe de la sensibilidad ambiental de la región seleccionada.

Problema geoespacial planteado

A partir de los requerimientos de la nueva funcionalidad, surgió la necesidad de encontrar una solución técnica al siguiente problema geoespacial: una vez seleccionada una región en el visor, ¿de qué manera obtenemos la longitud de cada tipo de costa?

La región seleccionada estaría representada por la lista de vértices de un polígono obtenido a partir de la herramienta de dibujo disponible en la librería OpenLayers.

El tipo de costa se encuentra en una capa almacenada en una base de datos PostGIS y servida por un servidor GeoServer. La capa contiene elementos de tipo LineString y de tipo MultiLineString. Cada elemento tiene un atributo que identifica el tipo de costa.

APROXIMACIÓN A LA SOLUCIÓN CON WPS DE GEOSERVER

Web Processing Service (WPS) es una interfaz propuesta por la OGC para estandarizar las peticiones y respuestas de procesos geoespaciales, de manera que un cliente pueda lanzar la ejecución de un proceso y recoger la respuesta. También especifica cómo publicar los procesos geoespaciales para que los clientes puedan descubrir los servicios proporcionados. Así, cada servidor puede implementar un conjunto de procesos diferente y publicar la lista en un documento de WPS *capabilites*.

GeoServer ofrece dos categorías de procesos:

- Procesos de JTS Topology Suite [10], una biblioteca Java de procesado de geometrías en dos dimensiones que implementa las operaciones propuestas por la OGC en Simple Features Specification for SQL [11], que incluye operaciones como el cálculo de la longitud de una curva o el área de una superficie; operaciones de composición como intersección, diferencia o unión; métricas como la distancia entre geometrías; etc.
- Procesos específicos de GeoServer, como reproyecciones o recolección de conjuntos de geometrías.

Las peticiones WPS pueden incluir la ejecución de procesos encadenados, de manera que la entrada de un proceso sea la salida del proceso anterior.

En la documentación de GeoServer [12] aparece un ejemplo de una encadenación de procesos para realizar un análisis geoespacial parecido al que queríamos resolver en el caso del visor de SACOSTA. En ese ejemplo explica cómo obtener cuántas millas de carreteras discurren en áreas protegidas, teniendo en cuenta que se disponen de dos capas, una con las carreteras y otra con las áreas protegidas. Para calcular el área total propone encadenar estos tres procesos geoespaciales:

1. gs:IntersectionFeatureCollection, con la que se obtiene la intersección entre dos colecciones de elementos.
2. gs:CollectGeometries, para recoger en una única geometría la colección de elementos intersecados.
3. JTS:length, que obtiene la longitud de la geometría.

Para realizar esta petición WPS, es necesario construir un XML que describa la ejecución de los tres procesos, donde cada uno recibe como entrada el resultado de la ejecución del proceso anterior. Por ello, en el XML resultante aparece primero el tercer proceso, con el segundo proceso anidado y el primer proceso anidado dentro de el segundo.

El ejemplo de la documentación de GeoServer era un punto de partida interesante, aunque nuestro caso era bastante más complejo, ya que necesitábamos obtener la longitud para cada tipo de costa, además de recuperar las rutas de las imágenes que se almacenan en la base de datos para cada tramo de costa.

Entonces, para poder cumplir nuestros requerimientos, era necesario combinar las funciones ya implementadas en GeoServer, además de un nuevo proceso ad hoc. La documentación de GeoServer explica cómo implementar un nuevo proceso. Para ello es necesario construir la aplicación en local, y añadir el proceso al paquete org.geoserver.wps.gs (nueva clase que implemente org.geotools.process.gs.GSProcess) [13].

Aunque esta aproximación podía ser factible, decidimos implementar una solución independiente a GeoServer.

LA SOLUCIÓN AD HOC (POSTGIS Y PYTHON)

Teniendo en cuenta nuestro escenario, la nueva funcionalidad del visor de SACOSTA también podía ser abordada a partir del acceso a los datos almacenados en la base de datos PostGIS.

Utilizamos las siguientes funciones geoespaciales disponibles en PostGIS para obtener el resultado [14]:

- ST_GeomFromText: función para construir geometrías 2D a partir de *Well Known Text* (WKT).
- ST_Transform: función que transforma los puntos de una determinada geometría a coordenadas en otro sistema de referencia espacial
- ST_Intersects: función para saber si dos geometrías tienen alguna región en común.
- ST_Intersection: función que devuelve la geometría que representa la región compartida por dos geometrías.
- ST_Lenght: cálculo de la longitud 2D de una geometría.

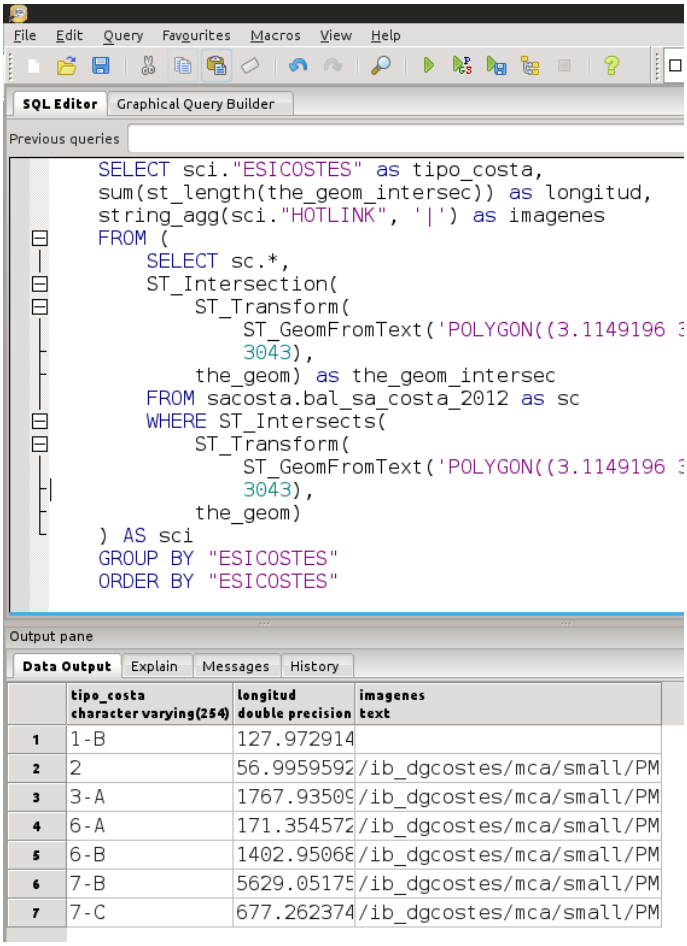

Figura 2: *Consulta SQL sobre PostGIS en Pgadmin3.*

En la Figura 2 podemos observar la consulta SQL que realizamos para obtener los datos que necesitamos. Seleccionamos todos los registros de línea de costa que tienen intersección con el polígono, agrupados por tipo de costa y con el cálculo de la suma de las longitudes de las líneas de costa intersecadas. El polígono está definido

utilizando el sistema de referencia EPGS:4326 (WSG84). Como el sistema de referencia espacial de nuestra capa de línea de costa es EPGS:3043, necesitamos realizar la transformación de la geometría obtenida a partir del WKT que representa el polígono.

Una vez construida la sentencia SQL, necesitábamos poder ejecutarla y obtener el resultado desde el visor SACOSTA. Para ello, encontramos que una buena opción podía ser el microframework de python Flask [15], que es realmente simple pero potente. A partir de muy pocas líneas de código se puede tener una aplicación web en funcionamiento. Para poder ejecutar la sentencia SQL, utilizamos el paquete más popular para acceder a bases de datos PostgreSQL desde Python: Psycopg [16].

```
@app.route('/api/v1.0/sacosta/<polygon>', methods=['GET'])
@app.route('/api/v1.0/sacosta/', methods=['POST'])
@crossdomain(origin='*')
def get_sacosta_polygon(polygon=None):
    if request.method == 'POST':
        polygon = request.form['polygon']
    try:
        region = utils.get_geometry_from_text(polygon)
    except ValueError, e:
        return jsonify({'error': str(e)})
    return utils.send_json_data(gisdata.get_data_sacosta(current_app.config, region))
```

Figura 3: código python para publicar una función en Flask

En Flask, podemos hacer que una función esté disponible como una ruta web añadiendo el decorador @app.route. Ese decorador recibe como parámetro una regla con la que se define la dirección URL. Esta regla puede contener parámetros que se parsean de la URL de la petición y se pasan como parámetros a la función python.

```
def get_polygon_vertices_from_text(polygon):
    """ Get list of vertexs from comma separated values

    :param polygon: string with a list of floats separated by commas that represents polygon vertexs
    :returns: list of vertexs
    """
    # Security check. All elements in polygon must be floats
    try:
        list_coord = [float(coord) for coord in polygon.split(',')]
    except ValueError:
        raise ValueError(
            'You must specify a polygon with a list of floats representing its vertices (comma separated)')

    vertices = zip(*[list_coord[i::2] for i in range(2)])
    return vertices


def get_geometry_from_text(polygon):
    """ Get ST_GeomFromText PostGIS SQL statement from a polygon

    :param polygon: string with a list of floats separated by commas that represents polygon vertexs
                    or a WKT representing a polygon
    :returns: string
    """
    if polygon.startswith('POLYGON'):
        try:
            # Check polygon is a valid wkt
            polygon_text = wkt.loads(polygon).wkt
        except:
            raise ValueError('Could not create geometry')
    else:
        # get polygon WKT from a list of vertices
        vertices = get_polygon_vertices_from_text(polygon)
        polygon_text = Polygon(vertices).wkt

    region = "ST_GeomFromText('%s', 4326)" % polygon_text
    return region
```

Figura 4: código python para generar la geometría de la región seleccionada

Nuestra función puede recibir en la misma URL una lista de coordenadas que representan los vértices de un polígono, o bien, puede recibir un polígono en formato WKT por POST. Utilizamos el módulo Shapely [17] para construir el WKT del polígono o para verificar que es correcto antes de construir la sentencia SQL.

El programa ejecuta la consulta SQL y devuelve una lista con los registros obtenidos. Finalmente, como queremos consumir esta petición a partir de una nueva herramienta en el visor de SACOSTA, la función devuelve esa lista en formato JSON. Para ello, utiliza el módulo jsonify de Flask.

```
conn = psycopg2.connect(config['DATABASE_URI'])
cur = conn.cursor(cursor_factory=DictCursor)
cur.execute(sql_sacosta)
# Process the result
result = []
for row in cur:
    obj = {
        'esicostes': row['ESICOSTES'],
        'longitud': round(row['longitud_intersec'], 2),
        'hotlink': []
    }
    if row['hotlink'] is not None:
        obj['hotlink'] = [link for link in row['hotlink'].split('|')]
    result.append(obj)
return result
```

Figura 5: código python de acceso a la base de datos

AMPLIANDO GEOEXPLORER

El visor SACOSTA está desarrollado sobre GeoExplorer, aplicación web basada en el framework GeoExt. GeoExplorer permite configurar y ampliar el visor con funcionalidades propias.

Para añadir una nueva herramienta, se puede crear una nueva clase javascript que extienda la clase gxp.plugins.Tool, para después añadirla al objeto de configuración del visor GeoExplorer (config.tools) [18]. Para ello, esta clase debe tener un atributo llamado *ptype* que identifique la herramienta.

```
Ext.namespace("gxp.plugins");
gxp.plugins.SensibilidadAmbiental = Ext.extend(gxp.plugins.Tool, {
    ptype: "sacosta_sensibilidadambiental",
    menuText: "Sensibilidad Ambiental",
    toolTip: "Muestra la sensibilidad ambiental de una área seleccionada",
    mapControls: {
        draw: null,
        drag: null
```

Figura 6: código javascript para definir una nueva herramienta en GeoExplorer

La nueva herramienta de GeoExplorer contiene una lista de acciones, consistentes en menús o botones, en los que se define las funciones a ejecutar. En nuestro caso, hemos añadido un único botón que activa o desactiva un control de dibujo de OpenLayers [19] (*OpenLayers.Control.DrawFeature, con el handler OpenLayers.Handler.Polygon*). Este control permite al usuario seleccionar el área de la que quiere obtener su análisis de sensibilidad ambiental.

Cuando el usuario finaliza el dibujo del polígono (*featureAdded* del control), por una parte se desactiva el control de dibujo y se activa un nuevo control para mover el polígono sobre el mapa (*OpenLayers.Control.DragFeature*), y por otra, realiza una llamada AJAX a la aplicación web Flask para obtener un JSON con los datos de la sensibilidad ambiental de la costa.

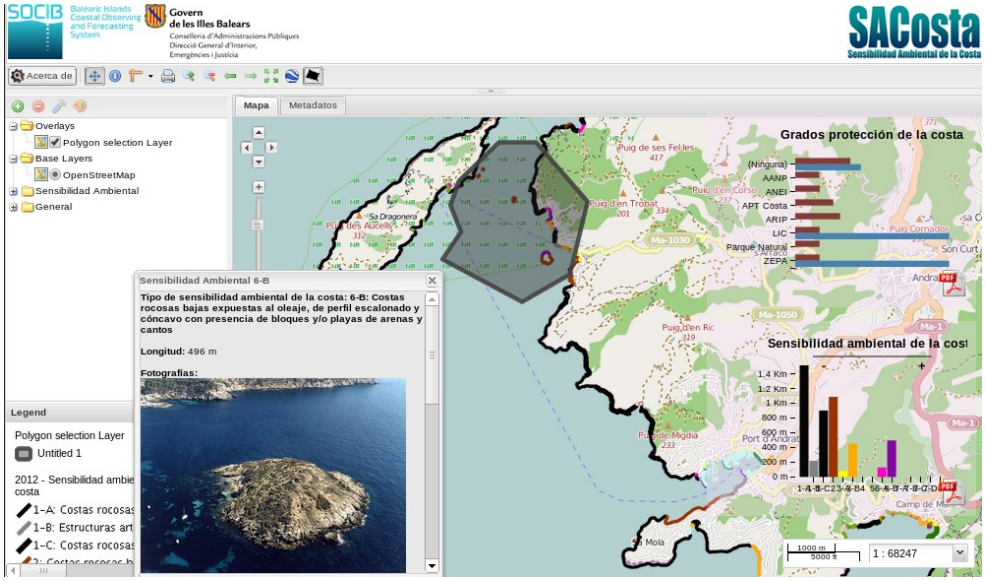

Figura 7: Visor SACOSTA con la herramienta de análisis del tipo de costa

Una vez obtenidos los datos, utilizamos D3.js [20] para dibujar un gráfico de barras sobre el mapa. Esta biblioteca permite que los gráficos generados sean interactivos. Cuando recibe nuevos datos, actualiza las barras de forma animada (función *transition*). Desde cada barra del gráfico se puede acceder al objeto de tipo de costa que representa, de manera que utilizamos esa información para:

- Mostrar un tooltip con la información básica cuando el usuario pasa por encima de la barra.
- Filtrar la capa de línea de costa para mostrar sólo las features de ese tipo de costa (utilizamos la clase *OpenLayers.Filter.Comparison*)
- Al hacer clic en la barra, se muestran las fotografías recuperadas de ese tipo de costa en una ventana interna.

GENERACIÓN DE INFORMES

Entre las nuevas funcionalidades incluidas en el visor de SACOSTA también se encuentra la posibilidad de generar informes sobre la sensibilidad ambiental de una área seleccionada en formato PDF. Esta funcionalidad se ha incluido en la misma aplicación web desarrollada en Flask.

El PDF se genera al vuelo a partir de un enlace que aparece junto a la gráfica en el visor SACOSTA. La aplicación web recibe la petición, y reutiliza las funciones de acceso a la base de datos PostGIS para obtener la información agregada por cada tipo de la línea de costa. El informe también incluye un mapa de la región seleccionada, generado a partir de capas almacenadas en nuestro servidor GeoServer. Para ello, utilizamos el cliente WMS de OWSLib [21].

El mapa y los datos se combinan en el informe PDF a partir de la biblioteca Pyjon.Reports [22], que es un wrapper de ReportLab, un potente generador de PDFs para python [23]. ReportLab permite definir el contenido del PDF a partir de una plantilla escrita en Report Markup Language (RML), que es un lenguaje basado en XML [24]. Las plantillas RML se pueden procesar a partir de una biblioteca comercial de ReportLab, o a partir de una implementación alternativa de la biblioteca libre z3c.rml (del ecosistema del servidor de aplicaciones web Zope) [25].

Figura 8: Informe generado por la aplicación web

PUBLICACIÓN DEL CÓDIGO

El código fuente de la aplicación web desarrollada en Flask y de la nueva herramienta desarrollada sobre GeoExplorer está disponible en repositorios públicos del perfil del SOCIB en la plataforma GitHub [26].

CONCLUSIONES

Hemos presentado la implementación de una nueva herramienta en GeoExplorer que utiliza como medio para acceder a datos geoespaciales una solución que no utiliza algunos estándares propuestos por la OGC (WPS o WFS), pero que se adaptaba a un escenario concreto y desarrollada con relativa facilidad.

Con esta experiencia queremos hacer una reflexión sobre el papel de los estándares, más concretamente los de procesado geoespacial. El objetivo de estos es posibilitar o facilitar el desarrollo de soluciones dentro de su dominio, y en nuestro escenario no ha sido el caso, al menos en su estado actual. El desarrollo basado en servicios OGC era más complejo y imponía unas dependencias más fuertes para llegar al mismo resultado final.

Bien sea porque las implementaciones de los servicios estándar no son lo suficientemente maduras o porque su uso no está muy extendido, el desarrollador tiene que sopesar la dificultades que introducen y los beneficios que aportan en su proyecto frente a otras alternativas. Al fin y al cabo, los estándares tienen que ser un medio para conseguir desarrollos eficientes y reusables, y cuando no cumplen ese propósito, su uso es cuestionable [27].

AGRADECIMIENTOS

La información espacial del visor se obtuvo en 2005-2006 en el marco de un convenio, entre la Conselleria d'Interior del Govern de les Illes Balears y la Universitat de les Illes Balears. Los datos presentados en SACOSTA son convenientemente revisados y actualizados por el SOCIB.

REFERENCIAS

- [1] SOCIB, *Visor cartográfico de la sensibilidad ambiental de la línea de costa las Illes Balears (SACOSTA)*, http://gis.socib.es/sacosta/
- [2] NOAA (2002), *Environmental sensitivity index guidelines.* NOAA Technical Memorandum NOS OR&R 11, p. 89
- [3] Balaguer, P.; Sardá, R.; Ruiz, M.; Diedricha, A.; Vizoso, G.; Tintoré, J. (2008) *A proposal for boundary delimitation for integrated coastal zone management initiatives*, Ocean & Coastal Management 51, pp. 806–814
- [4] OpenGeo Suite, http://suite.opengeo.org
- [5] Orfila, A.; Vizoso, G.; Álvarez, A.; Onken, R.; Jordi, A.; Basterretxea, G.; Fernández, V.; Casas, B.; Fornes, A.; Tintoré, J. (2004), *La respuesta científica ante el vertido del buque Prestige: oceanografía operacional en España y la experiencia del IMEDEA*, Revista Real Academia Ciencias Exactas, Física y Naturales. Monográfico: Oceanografía y recursos marinos: riesgos y desarrollo sostenible, Vol. 98, Nº. 1, pp. 191-207,
- [6] Tintoré, J. (2005), *La investigación, un elemento clave del futuro Económico y social: el papel del CSIC ante el vertido del buque Prestige*. En: *Las lecciones de la catástrofe del Prestige*, CSIC, pp. 45-53
- [7] IMEDEA (2007), *Tipos de costa potencialmente afectados por el vertido del buque Don Pedro*, http://repository.socib.es/repository/entry/get/Informe_interno_donpedro_04.pdf?entryid=cb0929da-a138-4e7c-93d2-d1b2f8e57b2d
- [8] Tintoré, J.; Vizoso, G.; Casas, B.; Heslop, E.; Pascual, A.; Orfila, A.; Ruiz, S.; Martínez-Ledesma, M.; Torner, M.; Cusí, S.; Diedrich, A.; Balaguer, P.; Gómez-Pujol, L.; Álvarez-Ellacuría, A.; Gómara, S.; Sebastian, K.; Lora, S.; Beltrán, J.P.; Renault, L.; Juzà, M.; Álvarez, D.; March, D.; Garau, B.; Castilla, C.; Cañellas, T.; Roque, D.; Lizarán, I.; Pitarch, S.; Carrasco, M.A.; Lana, A.; Mason, E.; Escudier, R.; Conti, D.; Sayol, J.M.; Barceló, B.; Alemany, F.; Reglero, P.; Massuti, E.; Velez-Belchí, P.; Ruiz, J.; Gómez, M.; Álvarez, A.; Ansorena, L.; Manríquez, M. (2013), *SOCIB: the Balearic Islands Observing and Forecasting System responding to science, technology and society needs*, Marine Technology Society Journal, Vol. 47, N. 1., pp. 101-117 http://doi.org/10.4031/MTSJ.47.1.10
- [9] Jordi, A.; Ferrer,M.I.; Vizoso, G.; Orfila, A.; Basterretxea, G.; Casas, B.; Álvarez, A.; Roig, D.; Garau, B.; Martínez, M.; Fernández, V.; Fornés, A.; Ruiz, M.; Fornós, J.J.; Balaguer, P.; Duarte, C.M.; Rodríguez, I.; Alvarez, E.; Onken, R.; Orfila, P.; Tintoré, J. (2006), *Scientific management of Mediterranean coastal zone: A hybrid ocean forecasting system for oil spill and search and rescue operations*. Marine Pollution Bulletin 53, pp. 361–368
- [10] JTS Topology Suite, http://www.vividsolutions.com/jts/JTSHome.htm
- [11] OGC (2010), *OpenGIS Implementation Specification for Geographic information - Simple feature access – Part 2: SQL Option*, http://www.opengeospatial.org/standards/sfs
- [12] OpenPlans (2014), *GeoServer 2.5.x User Manual*, http://docs.geoserver.org/latest/en/user/index.html
- [13] Boundless (2012), *Creating a GeoServer Split Polygon WPS Process*, http://boundlessgeo.com/2012/06/splitpolygon-wps-process-p1/

- [14] Obe,R. O.; Hsu, L. S. (2011), *PostGIS in Action*, Manning Publications Co.
- [15] Flask, http://flask.pocoo.org/
- [16] Psycopg, http://initd.org/psycopg/
- [17] Shapely, https://pypi.python.org/pypi/Shapely
- [18] GeoExplorer, http://suite.opengeo.org/opengeo-docs/geoexplorer/using/index.html
- [19] OpenLayers, http://openlayers.org/
- [20] D3.js, http://d3js.org/
- [21] OWSLib, http://geopython.github.io/OWSLib/
- [22] Pyjon.Reports, https://bitbucket.org/jon1012/pyjonreports
- [23] ReportLab, http://www.reportlab.com/r
- [24] ReportLab, RML Tag Reference, http://www.reportlab.com/software/rml-reference/
- [25] z3c.rml, https://pypi.python.org/pypi/z3c.rml
- [26] SOCIB public code at GitHub, https://github.com/socib
- [27] xkcd, How standards proliferate, https://xkcd.com/927/