Universidad del País Vasco
Euskal Herriko Unibertsitatea

Facultad de Informática

Ingeniería Informática

▪ Proyecto Fin de Carrera ▪

Desarrollo de componentes para el panel de administración de GeoServicios® on-line

Hodei López Leonardo

Noviembre 2013

Proyecto realizado en la empresa Geograma

informatika fakultatea
facultad de informática

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

1. INTRODUCCIÓN

En este documento, se detalla el proyecto realizado en la empresa Geograma por el alumno de Ingeniería Informática Hodei López Leonardo. Geograma es un grupo de empresas especializadas en la captura, suministro, tratamiento y gestión de Geoinformación.

En las siguientes páginas, se detalla la realización de un Panel de Administración que ofrece una solución completa a la hora de realizar visores para mapas y que permite al cliente configurar un visor en base a sus necesidades, mediante diferentes módulos. Dicho panel, se ha realizado utilizando un desarrollo basado en el prototipado. En el caso de este proyecto, se especifica el desarrollo de los dos primeros prototipos.

El documento comienza situando la empresa en la cual se ha realizado el proyecto y las tecnologías utilizadas. Tras esto, se especifican los objetivos del proyecto.

A continuación se detallan cada una de las fases del proyecto, comenzando por el aprendizaje, y siguiendo con la especificación, diseño y análisis realizado en cada uno de los módulos que componen los diferentes prototipos. Dichos módulos han sido los siguientes:

- **Panel de login**: Panel que permite identificar el acceso de usuarios.
- **Módulo de usuarios y grupos:** Módulo que permite la administración de usuarios y grupos.
- **Módulo de datos:** Módulo que permite la creación de almacenes y plantillas de datos.
- **Módulo geoAsset:** Módulo que permite la asociación de permisos con la aplicación móvil desarrollada paralelamente.
- **Módulo de mejora de datos:** Módulo que permite la subida de archivos para la creación de almacenes de datos.

En esto módulos, se han detallado tanto los componentes como los servicios que los componen.

Seguidamente, se expone lo relativo a la gestión del proyecto, que consta de los siguientes apartados:

- **Gestión del alcance:** Se detalla el alcance del proyecto y del producto, el EDT, las restricciones...
- **Gestión de actividades:** Se detallan las diferentes actividades y el cronograma.
- **Gestión de costes:** Se detalla una pequeña gestión de costes.
- **Gestión de calidad:** Se detalla cómo se ha realizado la gestión de la calidad, las diferentes pruebas y normativas seguidas.
- **Gestión de recursos humanos:** Se especifican los recursos humanos del proyecto.
- **Gestión de comunicaciones:** Se detalla cómo se han realizado las comunicaciones.
- **Gestión de riesgos:** Se especifican los diferentes riesgos localizados y como hacer frente a ellos.
- **Gestión de adquisiciones:** Se detalla las adquisiciones con las que se ha contado.

Por último, se especifican las conclusiones del proyecto, así como la bibliografía y los pertinentes agradecimientos.

En todo momento se dispone de varios anexos con la información detallada de cada uno de los apartados, así como información adicional sobre normativas de desarrollo y manuales de uso.

2. ANTECEDENTES

Este apartado contiene la información necesaria para conocer la empresa. Se hace hincapié en los campos que abarca, las soluciones que ofrece y las tecnologías que utiliza. A continuación se pueden ver con más detalles estos aspectos.

2.1. ACERCA DE GEOGRAMA

Geograma es un grupo de empresas especializadas en la captura, suministro, tratamiento y gestión de Geoinformación.

¿Qué es la Geoinformación?

La Geoinformación abarca cualquier tipo de datos digitales con una componente geográfica o territorial.

Desde imágenes de satélite hasta fotografías terrestres geolocalizadas, pasando por cartografía, datos socioeconómicos o de tránsito peatonal, callejeros, inventarios, información del tráfico, o datos de clientes y mercado. El amplio abanico de Geoinformación disponible hoy en día permite optimizar procesos de gestión y soporte a la toma de decisiones tanto en el ámbito público como en el privado.

Clientes

Movistar, La Caixa, Páginas Amarillas, Gobierno Vasco, Ayuntamiento de Sevilla, Red Eléctrica de España o la Comisión del Mercado de las Telecomunicaciones son algunos de los clientes.

2.2. SOLUCIONES QUE OFRECE

Geomarketing

El Geomarketing se refiere al uso del componente espacial contenido en los datos corporativos y del mercado, para la toma de decisiones empresariales inteligentes y más efectivas.

El Geomarketing consiste en el análisis y visualización de los datos con el objetivo de detectar relaciones y tendencias que de otro modo pasarían desapercibidos.

Por lo tanto, el Geomarketing constituye la base fundamental para una correcta definición y planificación de territorios de ventas o comerciales. La utilización de mapas digitales para la visualización del mercado, los clientes y los datos corporativos permiten una toma de decisiones efectiva y eficiente.

Geocodificación

La geocodificación es el proceso que permite obtener coordenadas a partir de direcciones postales. Las coordenadas obtenidas posibilitan la ubicación de los elementos en un mapa, y por tanto comenzar a analizar gráficamente dichos elementos en función de variables geográficas: Cercanía o lejanía, área de influencia, densidad, relación con otros elementos, rutas de acceso, etc.

Las soluciones de geocodificación están basadas en la arquitectura e-GIS, que utiliza una infraestructura de datos espaciales para construir aplicaciones que permitan calcular unas coordenadas a partir de una dirección postal.

DSS / IDE Verticales

Los sistemas de Soporte a la Decisión (DSS) y las Infraestructuras de Datos Espaciales (IDE) son desarrollos verticales donde la experiencia, los datos y las herramientas trabajan juntas para la solución de un problema específico.

2.3 TECNOLOGÍAS UTILIZADAS:

A continuación se indican las tecnologías utilizadas durante la realización del proyecto. A la hora de realizar el proyecto. Estas han sido impuestas por le empresa, ya que tanto las tecnologías a utilizar como la metodología de trabajo están especificadas en documentos de normativas de desarrollo.

OpenLayers

OpenLayers es una biblioteca de JavaScript de código abierto bajo una derivación de la licencia BSD para mostrar mapas interactivos en los navegadores web. OpenLayers ofrece un API para acceder a diferentes fuentes de información cartográfica en la red: Web Map Services, Mapas comerciales (tipo Google Maps, Bing, Yahoo), Web Features Services, distintos formatos vectoriales, mapas de OpenStreetMap, etc.

Inicialmente fue desarrollado por MetaCarta en Junio del 2006. Desde el noviembre del 2007 este proyecto forma parte de los proyectos de Open Source Geospatial Foundation. Actualmente el desarrollo y el soporte corre a cargo de la comunidad de colaboradores.

OpenStreetMap

OpenStreetMap (también conocido como OSM) es un proyecto colaborativo para crear mapas libres y editables.

Los mapas se crean utilizando información geográfica capturada con dispositivos GPS móviles, *ortofotografías (*presentación fotográfica de una zona de la superficie terrestre, en la que todos los elementos presentan la misma escala) y otras fuentes libres. Esta cartografía, tanto las imágenes creadas como los datos vectoriales almacenados en su base de datos, se distribuye bajo licencia abierta Open Database License (ODbL).

Los usuarios registrados pueden subir sus trazas desde el GPS y crear y corregir datos vectoriales mediante herramientas de edición creadas por la comunidad OpenStreetMap.

Framework extJs

Ext JS (pronunciado como "ekst") es una biblioteca de JavaScript para el desarrollo de aplicaciones web interactivas usando tecnologías como AJAX, DHTML y DOM. Fue desarrollada por Sencha.

Originalmente construida como una extensión de la biblioteca YUI por Jack Slocum, en la actualidad puede usarse como extensión para las biblioteca jQuery y Prototype. Desde la versión 1.1 puede ejecutarse como una aplicación independiente.

Dispone de un conjunto de componentes (widgets) para incluir dentro de una aplicación web, como:

- Cuadros y áreas de texto.
- Campos para fechas.
- Campos numéricos.
- Combos.
- Radiobuttons y checkboxes.
- Editor HTML.
- Elementos de datos (con modos de sólo lectura, datos ordenables, columnas que se pueden bloquear y arrastrar, etc.).
- Árbol de datos.
- Pestañas.
- Barra de herramientas.
- Menús al estilo de Windows.
- Paneles divisibles en secciones.
- Sliders.
- Gráficos

Varios de estos componentes están capacitados para comunicarse con el servidor usando AJAX. También contiene numerosas funcionalidades que permiten añadir interactividad a las páginas HTML, como:

- Cuadros de diálogo.
- quicktips para mostrar mensajes de validación e información sobre campos individuales.

PostgreSQL

PostgreSQL es un SGBD relacional orientado a objetos y libre, publicado bajo la licencia BSD.

Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa y/o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre y/o apoyados por organizaciones comerciales. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group).

PostgreSQL provee nativamente soporte para:

- Números de precisión arbitraria.
- Texto de largo ilimitado.
- Figuras geométricas (con una variedad de funciones asociadas).
- Direcciones IP (IPv4 e IPv6).
- Bloques de direcciones estilo CIDR.
- Direcciones MAC.
- Arrays.

Adicionalmente los usuarios pueden crear sus propios tipos de datos, los que pueden ser por completo indexables gracias a la infraestructura GiST de PostgreSQL. Algunos ejemplos son los tipos de datos GIS creados por el proyecto PostGIS.

Otras características

- Claves ajenas también denominadas Llaves ajenas o Claves Foráneas (foreign keys).
- Disparadores (triggers): Un disparador o trigger se define como una acción específica que se realiza de acuerdo a un evento, cuando éste ocurra dentro de la base de datos. En PostgreSQL esto significa la ejecución de un procedimiento almacenado basado en una determinada acción sobre una tabla específica.
- Integridad transaccional.
- Herencia de tablas.
- Tipos de datos y operaciones geométricas.
- Soporte para transacciones distribuidas. Permite a PostgreSQL integrarse en un sistema distribuido formado por varios recursos (p.ej, una base de datos PostgreSQL, otra Oracle, una cola de mensajes IBM MQ JMS y un ERP SAP) gestionado por un servidor de aplicaciones donde el éxito ("commit") de la transacción global es el resultado del éxito de las transacciones locales.

Hibernate

Hibernate es una herramienta de Mapeo objeto-relacional (ORM) para la plataforma Java (y disponible también para .Net con el nombre de NHibernate) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones.

Hibernate es software libre, distribuido bajo los términos de la licencia GNU LGPL. Como todas las herramientas de su tipo, Hibernate busca solucionar el problema de la diferencia entre los dos modelos de datos coexistentes en una aplicación: el usado en la memoria de la computadora (orientación a objetos) y el usado en las bases de datos (modelo relacional). Para lograr esto permite al desarrollador detallar cómo es su modelo de datos, qué relaciones existen y qué forma tienen. Con esta información Hibernate le permite a la aplicación manipular los datos en la base de datos operando sobre objetos, con todas las características de la POO. Hibernate convertirá los datos entre los tipos utilizados por Java y los definidos por SQL. Hibernate genera las sentencias SQL y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todos los motores de bases de datos con un ligero incremento en el tiempo de ejecución.

Hibernate está diseñado para ser flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente. También tiene la funcionalidad de crear la base de datos a partir de la información disponible.
Hibernate ofrece también un lenguaje de consulta de datos llamado HQL (Hibernate Query Language), al mismo tiempo que una API para construir las consultas programáticamente (conocida como "criteria").

Hibernate para Java puede ser utilizado en aplicaciones Java independientes o en aplicaciones Java EE, mediante el componente Hibernate Annotations que implementa el estándar JPA, que es parte de esta plataforma.

Notepad++

Notepad++ es un editor de texto y de código fuente libre con soporte para varios lenguajes de programación. De soporte nativo a Microsoft Windows.

Se parece al Bloc de notas en cuanto al hecho de que puede editar texto sin formato y de forma simple. No obstante, incluye opciones más avanzadas que pueden ser útiles para usuarios avanzados como desarrolladores y programadores.

Se distribuye bajo los términos de la Licencia Pública General de GNU. Las características de este editor son las que siguen:

- **Coloreado y envoltura de sintaxis**: si se escribe en un lenguaje de programación o marcado, Notepad++ es capaz de resaltar las expresiones propias de la sintaxis de ese lenguaje para facilitar su lectura.
- **Pestañas**: al igual que en muchos navegadores, se pueden abrir varios documentos y organizarlos en pestañas.
- **Resaltado de paréntesis**: cuando el usuario coloca el cursor en un paréntesis, Notepad++ resalta éste y el paréntesis correspondiente de cierre o apertura. También funciona con corchetes y llaves.
- **Grabación y reproducción de macros**.
- **Soporte de extensiones**: incluye algunas por defecto.

3. OBJETIVOS DEL PROYECTO

El proyecto fin de carrera consiste en el desarrollo de varios componentes del **panel de administración**. Mediante dicho panel, se ofrece una solución completa a la hora de realizar visores para mapas, que permite al cliente configurar el visor en base a sus necesidades, mediante los siguientes módulos:

- **Panel de login**: Este panel permite el acceso a la aplicación.
- **Módulo de usuarios/grupos**: Este componente permite gestionar grupos y usuarios, tanto la creación, edición de los mismos como la visualización de estos.
- **Módulo de datos**: Este componente permite gestionar las fuentes de datos del cliente. Crear plantillas a partir de datos procedentes de BBDD propias, así como la edición de dichas plantillas.
- **Módulo GeoAsset**: Este componente permite configurar aplicaciones web o visores. Un visor tendrá asociado un mapa, una lista de control de accesos, etc.

Siguiendo la estructura de módulos, también es objeto del proyecto la realización de un segundo prototipo que contiene la mejora de uno de los módulos ya creados:

- **Mejora de módulo de datos:** Este componente implementa además de las funcionalidades creadas en la anterior versión, una funcionalidad para permitir la subida de datos a la aplicación.

Por otro lado, también es objeto del proyecto la realización inicial de una fase de aprendizaje con una serie de tareas que permiten conocer las tecnologías utilizadas en dicho proyecto. Los objetivos de esta fase son los siguientes:

- **Visor de prueba:** Creación de un visor de prueba que permite familiarizarse con el framework Javascript Sencha extJs y la tecnología relativa a mapas ofrecida por openLayers.
- **Servicio de prueba**: Creación de un servicio de prueba que permite familiarizarse con la creación de capas SOAP, SERVLET y JAVASCRIPT y la utilización de Hibernate para la abstracción de la base de datos a utilizar.

4. APRENDIZAJE

El objeto de este apartado es describir las tareas realizadas relativas al aprendizaje. Al comienzo del proyecto, se consideró oportuno la realización de varias tareas con el objetivo de familiarizarse con las tecnologías a utilizar.

Los objetivos de la tarea de aprendizaje, son los siguientes:

- Lectura y comprensión de la api del framework Sencha extJs.
- Lectura y comprensión de la api de openLayers.
- Creación de un visor simple de mapas que contiene las siguientes características:
    - Una distribución de paneles que contiene una cabecera, una zona a la derecha, y una parte central donde es ubicado el mapa.
    - La zona de la derecha contiene un panel de tipo acordeón con 3 funcionalidades diferentes:
        - Un apartado para seleccionar capas y que estas se visualicen en el mapa central.
        - Un apartado con un formulario en el que se puede introducir los datos de una calle (En este caso de Vitoria-Gasteiz), que localiza dicha ubicación e introduce un marcador.
        - Un apartado con herramientas de dibujo sobre mapas: Líneas, polígonos y puntos.
- Creación de la capa SOAP de un servicio de prueba.

A continuación se detalla cómo se han desarrollado cada una de ellas.

4.1. VISOR DE PRUEBA

Para comenzar a desarrollar el visor de prueba lo primero ha sido crear la distribución de paneles que se nos ha indicado en las restricciones de la tarea. Para ello, el framework de extJs, contiene un Layout de tipo border, que permite indicarle el posicionamiento de los diferentes paneles que se van a crear.

Para ello, en la clase principal, se tiene que introducir un código que tenga la siguiente estructura:

```
    var viewport = Ext.create('Ext.Viewport', {
                layout: {
                    type: 'border',
                    padding: 15
                },
                defaults: {
                    split: true
                },
                items: [{
                    region: 'north',
                    collapsible: true,
                    title: 'Cabecera',
                    split: true,
                    height: 100,
                    minHeight: 60,
                },
                {
                    region: 'center',
                    title: 'Mapa',
                    html:'<div id="map"></div>',
                    minHeight: 80
                },{
                    region: 'east',
                    collapsible: true,
                    floatable: true,
                    split: true,
                    width: 300,
                    minWidth: 150,
                    minHeight: 140,
                    title: 'Herramientas',
                    items: [
                        oThis.panelDerecha.getComponent()
                    ]
                }]
            });

            oThis.panelMapa = new PanelMapa();
    }
```

Como se puede ver en la figura anterior, se han marcado en negrita los parámetros donde se indican la región donde se quieren ubicar los diferentes paneles. En este caso, se han creado los siguientes:

- Panel de cabecera: Región: 'Norte': En este caso se deja como un panel vacio.
- Panel central: Región: 'Central': En este caso, se le asocia el panel llamado panelMapa.
- Panel derecho: Región: 'Este': En este caso, se le asocia un panel llamado panelDerecha.

A continuación se detalla cada uno de los paneles. El panel central es el panel encargado de mostrar el mapa. Este panel se encarga de soportar las funcionalidades para poder:

- Establecer un marcador en una ubicación determinada.
- Permitir dibujar líneas.
- Permitir dibujar polígonos.
- Permitir dibujar puntos.

El código del panel es el siguiente:

```
initialize: function() {

    this.map = new OpenLayers.Map('map',{
        controls: [
                    new OpenLayers.Control.Navigation(),
                    new OpenLayers.Control.PanZoomBar(),
                    new OpenLayers.Control.LayerSwitcher(),
                    new OpenLayers.Control.MousePosition(),
                   ]
    });

    //this.marcar(-400555.40131135966,4924376.402975677);

    this.mapLayer = new OpenLayers.Layer.XYZ(
        " TomTom",
        " url del servicio",
        {
            sphericalMercator: true,
            wrapDateLine: true,
            transitionEffect: "resize",
            buffer: 1,
        }
    );
    this.map.addLayer(this.mapLayer);

    // capa para las marcas
    this.vectorLayerMarks = new OpenLayers.Layer.Vector(" Marca");
    // capa para dibujar puntos
    this.vectorLayerPoint = new OpenLayers.Layer.Vector(" Point Layer");
    this.punto = new OpenLayers.Control.DrawFeature();
    // capa para dibujar lineas
    this.vectorLayerLine = new OpenLayers.Layer.Vector(" Line Layer");
    this.linea = new OpenLayers.Control.DrawFeature();
    // capa para dibujar poligonos
    this.vectorLayerPolygon = new OpenLayers.Layer.Vector(" Polygon Layer");
    this.poligono = new OpenLayers.Control.DrawFeature();

    var lon = -410623.31605338;
    var lat = 4927018.3673945;
    var zoom = 5;
    this.map.setCenter(new OpenLayers.LonLat(lon, lat), zoom);

},
```

En este panel, lo primero que se ha hecho, es añadir diferentes controles al mapa como puede ser la barra de zoom, que aparezcan las diferentes capas... A continuación, se ha creado el mapa utilizando una capa obtenida mediante el servicio de TomTom disponible en la Plataforma GeoServicios Online. Tras esto, se han añadido las funcionalidades anteriormente comentadas. Por último, se ha establecido el punto donde se quiere que se centre el mapa cuando se inicializa el visor.

Después de completar el panel central, se continua con el panel derecho. Este panel, como ya se ha comentado anteriormente, será de tipo acordeón. Para crear un panel de este tipo, el framework de extJs nos proporciona el Layout 'accordion'. El código es el que sigue:

```
        initialize: function() {

            this.panelCapas = new PanelCapas();
            this.panelGeocode = new GeocodePanel();
            this.panelDibujar = new PanelDibujar();

            var item1 = Ext.create('Ext.Panel', {
                title: 'Capas',
                items:[
                    this.panelCapas.getComponent()
                ]
            });
            var item2 = Ext.create('Ext.Panel', {
                title: 'B&uacutesqueda de calle',
                items:[
                    this.panelGeocode.getComponent()
                ]
            });
            var item3 = Ext.create('Ext.Panel', {
                title: 'Otras herramientas',
                items:[
                    this.panelDibujar.getComponent()
                ]
            });
            this.panel = Ext.create('Ext.Panel', {
                title: 'Men&uacute',
                collapsible: true,
                region:'east',
                split:true,
                width: 210,
                layout:'accordion',
                items: [item1,item2, item3],
            });
        },
```

En la figura anterior, se puede observar como se ha creado el panel de tipo acordeón y se le ha añadido tres paneles diferentes. Estos paneles han sido:

- Panel de capas: Se encarga de la funcionalidad para cambiar de una capa a otra.
- Panel de geocoding: Este panel nos permite introducir una dirección de una calle concreta y ubicarla en el mapa.
- Panel para dibujar: Se encarga de las funcionalidades de dibujar puntos, líneas y polígonos en el mapa.

Seguidamente, se especifica el primero de estos paneles. El panel de capas, sigue la siguiente estructura:

```
    initialize:function(){

      this.panel = {
          xtype: 'buttongroup',
          margin: '15 12 12',
          columns: 2,
          items: [
         {
               xtype: 'button',
               margin: '5 5 5',
               text: 'Callejero',
               handler : function(){
                   createStreetLater();
               }
          },{
               xtype: 'button',
               margin: '5 5 5',
               text: 'Base',
               handler : function(){
                   createBaseLayer();
               }
          }]
      }
  },
```

En este panel, se han creado dos botones. Dichos botones, nos permiten cambiar de capa cuando se pulsa en ellos.

Tras esto se analiza el panel de geocoding. Este panel, tiene la siguiente estructura:

```
initialize:function(){

    this.panel = Ext.widget({
        xtype: 'form',
        layout: 'form',
        collapsible: false,
        id: 'simpleForm',
        frame: true,
        bodyPadding: '5 5 5 5',
        width: 295,
        fieldDefaults: {
            msgTarget: 'side',
            labelWidth: 75
        },
        defaultType: 'textfield',
        items: [{
            fieldLabel: 'Provincia',
            afterLabelTextTpl: required,
            name: 'provincia',
        },{
            fieldLabel: 'Municipio',
            afterLabelTextTpl: required,
            name: 'municipio',

        },{
            fieldLabel: 'Calle',
            afterLabelTextTpl: required,
            name: 'calle',

        },{
```

```
            },{
                fieldLabel: 'N&uacutemero',
                name: 'numero',

            }],

            buttons: [{
                text: 'Buscar',
                handler: function(){
                    this.provincia = getForm().findField("provincia").getValue();
                    this.municipio =getForm().findField("municipio").getValue();
                    this.calle = oThis.panel.getForm().findField("calle").getValue();
                    this.numero = ogetForm().findField("numero").getValue();
                    if (this.provincia=="" || this.municipio=="" || this.calle==""){
                        alert('Introduzca los campos requeridos');
                    }
                    else{
                        oThis.locateStreet(this.provincia,this.municipio,
                        this.calle,this.numero);
                    }
                },
                },{
                text: 'Cancelar',
                handler: function(){
                    oThis.panel.getForm().reset();
                    }
                },{
                text: 'Limpiar marcas',
                handler: function(){
                    MiDemo.panelMapa.limpiarMarcas();
                }
            }]
        });
    },
```

Como se puede observar, el panel de geocoding, contiene un formulario para introducir los dates de la calle a ubicar. A continuación, mediante un botón se recogen los parámetros introducidos en el formulario y mediante un servicio de la Plataforma GeoServicios Online, se consiguen los puntos a establecer en el mapa.

Por último se analiza el panel para dibujar. Este panel, tiene la siguiente estructura:

```
    initialize : function(){

        this.panel = {
        xtype: 'buttongroup',
        margin: '15 12 12',
        columns: 3,
        items: [
        {
            xtype: 'button',
            margin: '5 5 5',
            text: 'Dibujar punto',
            handler : function(){
                MiDemo.panelMapa.dibujarPunto();
            }
        },{
            xtype: 'button',
            margin: '5 5 5',
            text: 'Dibujar l&iacutenea',
            handler : function(){
                MiDemo.panelMapa.dibujarLinea();
            }
        },{
```

```
            xtype: 'button',
            margin: '5 5 5',
            text: 'Dibujar pol&iacutegono',
            handler : function(){
                MiDemo.panelMapa.dibujarPoligono();
            }
        },{
            xtype: 'button',
            margin: '5 5 5',
            text: 'Limpiar',
            handler : function(){
                MiDemo.panelMapa.limpiarDibujos();
            }
        },{
            xtype: 'button',
            margin: '5 5 5',
            text: 'Cancelar',
            handler : function(){
                MiDemo.panelMapa.cancelar();
            }
        }]
        }
    },
```

Como se puede observar, se ha introducido un botón por cada una de las funcionalidades propuestas al inicio de la tarea.

El resultado final del visor completado es el que se puede ver en las siguientes 4 figuras:

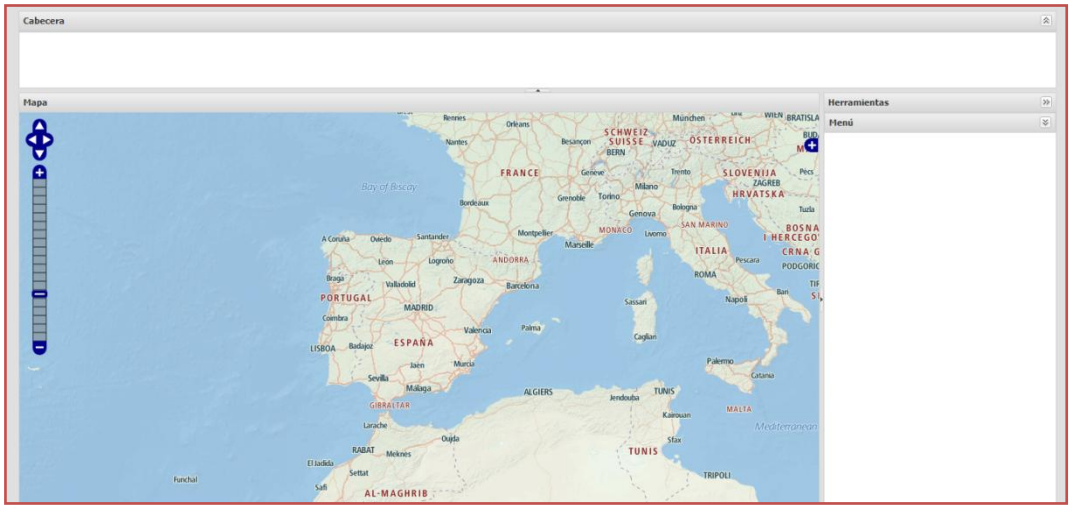


Ilustración 1 Visor de prueba 1

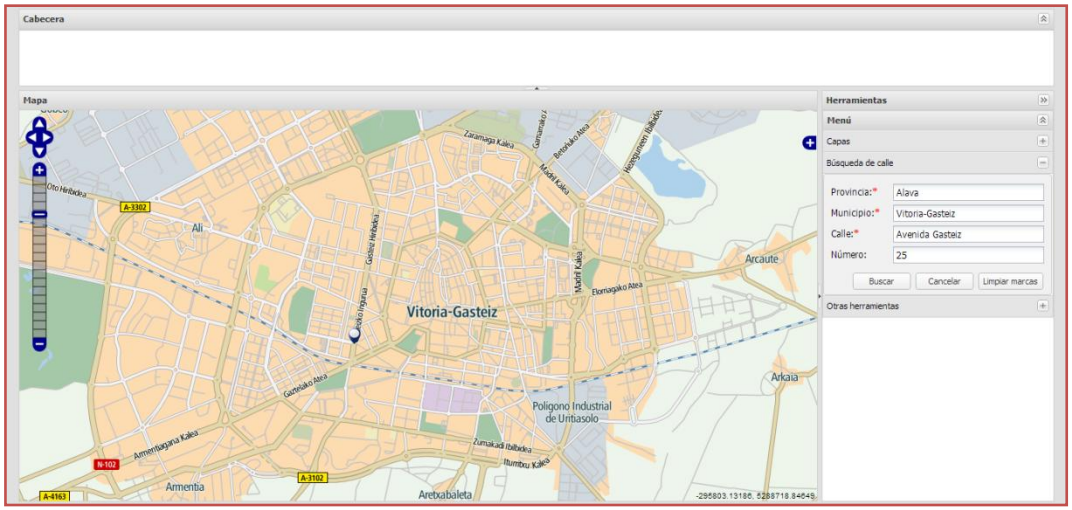


Ilustración 2 Visor de prueba 2

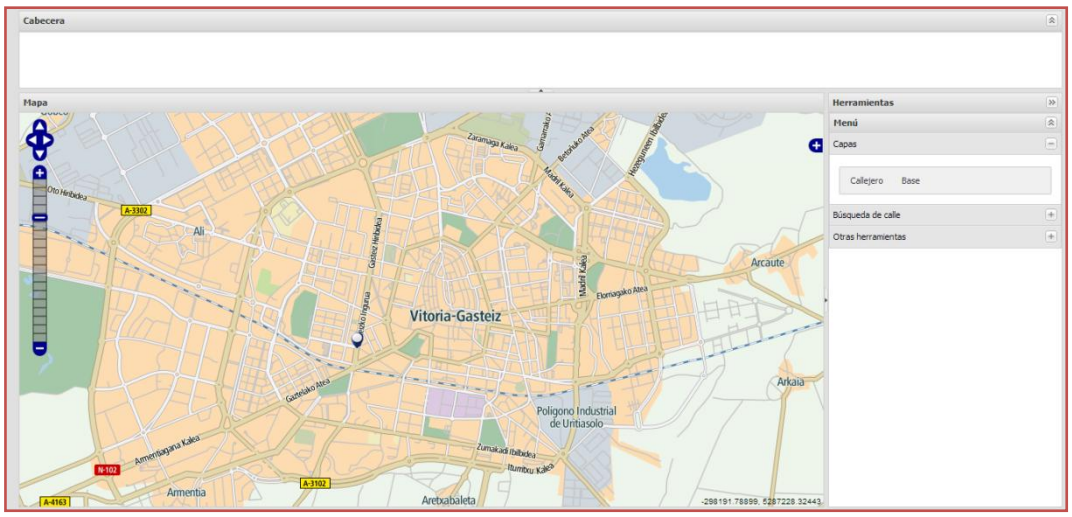


Ilustración 3 Visor de prueba 3

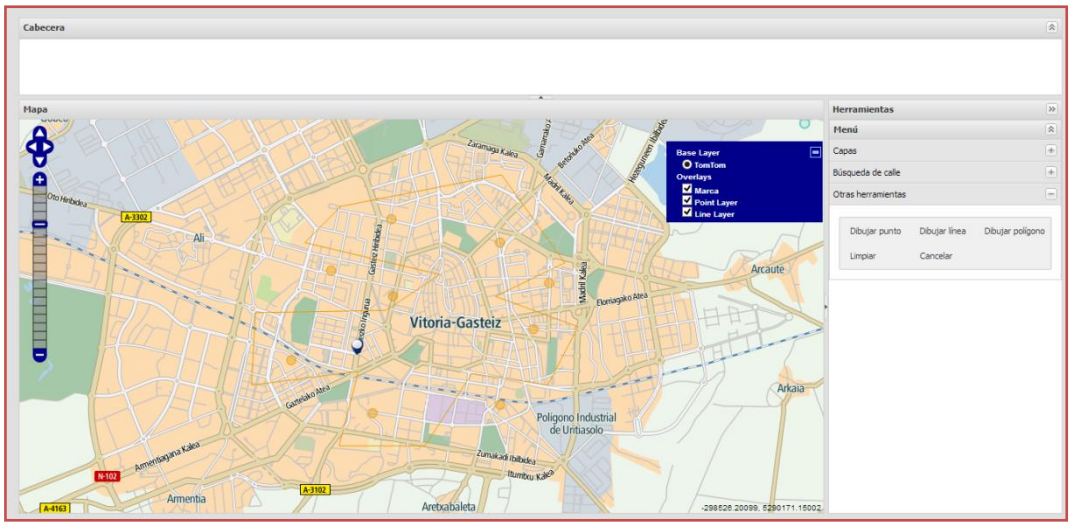


Ilustración 4 Visor de prueba 4

4.2. SERVICIO DE PRUEBA

El servicio de prueba está desarrollado en el Anexo II: Normativa de desarrollo y servicios y Hibernate: Creación de un nuevo servicio SOAP

5. ADMINISTRATION PANEL: PROTOTIPO 1.0

Mediante del panel de administración, se ofrece una solución completa a la hora de realizar visores para mapas, que permite al cliente configurar el visor en base a sus necesidades, mediante diferentes módulos.

Teniendo en cuenta estos módulos, la estructura general que sigue dicho panel es la siguiente:

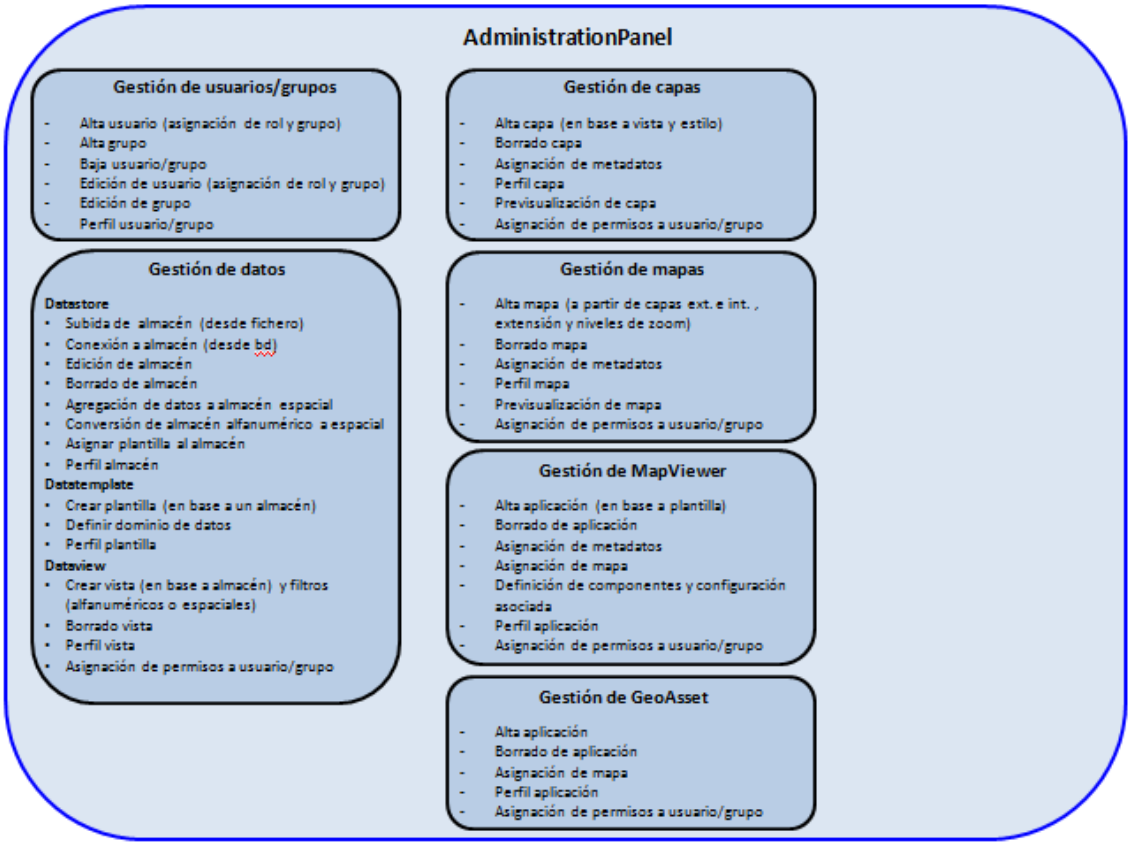


Ilustración 5 Estructura Administration Panel

Cada uno de los módulos, se encarga de una serie de funcionalidades, que se pueden combinar mediante un archivo de configuración, adaptándose a las necesidades de el cliente.

Los módulos de este proyecto son los que siguen:

- **Módulo de usuarios y grupos:** Encargado de la gestión y administración de usuarios y grupos.
- **Módulo de datos:** Encargado de la gestión y administración de plantillas y almacenes de datos.
- **Módulo geoAsset:** Encargado de asignar diferentes permisos para la aplicación móvil.

Aparte de estos módulos, se cuenta con un panel de login para acceder a la aplicación.

En los siguientes apartados, se expone el análisis y especificación de los prototipos desarrollados que componen el panel de administración. Como ya se ha comentado en la introducción, el proyecto está orientando al desarrollo basado en prototipado. A la hora de desarrollar los componentes, se han creado una serie de plantillas para facilitar el desarrollo de los mismos.

El primer prototipo, contiene una primera versión de dicho panel. Esta versión contiene los siguientes módulos:

- **Módulo de usuarios y grupos:** Encargado de la gestión y administración de usuarios y grupos.
- **Módulo de datos:** Encargado de la gestión y administración de plantillas y almacenes de datos.
- **Módulo de capas (no es objeto de este proyecto):** Encargado de la gestión y administración de capas.
- **Módulo de mapas (no es objeto de este proyecto):** Encargado de la gestión y administración de mapas.
- **Módulo geoAsset:** Encargado de asignar diferentes permisos para la aplicación móvil.

Estos módulos, nos permiten la administración necesaria para poder gestionar un visor de mapas. Mediante el **módulo de usuarios y grupos**, se puede administrar la creación y gestión de usuarios y grupos, permitiendo al cliente poder controlar el acceso al visor de mapas mediante diferentes permisos de acceso. Mediante el **módulo de datos**, se dota de la funcionalidad de crear almacenes de datos y plantillas de datos. Utilizando dichos datos, mediante el **módulo de capas**, se pueden crear diferentes capas que se pueden asociar a diferentes mapas. Por último, el **módulo de mapas**, se encarga de la creación de diferentes mapas y permite asociar a dichos mapas las capas deseadas.

Por ejemplo, una empresa desea visualizar en un mapa los diferentes puntos de venta que tiene en una zona determinada. Para ello, en una tabla de una base de datos, dispone de las coordenadas donde están ubicados dichos puntos de venta. Mediante el módulo de datos, dicho cliente podrá crear un almacén de datos con dichas coordenadas, y luego mediante el módulo de capas, podrá crear una capa que contenga todos los puntos especificados. A continuación, asociando esta capa a un mapa determinado, podrá visualizarlos en su visor de mapas.

Para dar un mayor servicio, el panel de administración también se ha desarrollado para entorno móvil. En la aplicación móvil, se pueden gestionar y administrar los diferentes almacenes de datos, de manera offline, facilitando el trabajo de campo. Mediante el **módulo geoAsset**, se pueden establecer permisos relativos a la aplicación móvil, es decir, seleccionar que usuarios, grupos, almacenes de datos y mapas pueden ser visibles en dicha aplicación.

El segundo prototipo, se centra en la mejora del **módulo de datos.**

Mediante esta mejora, los clientes pueden subir datos para poder crear almacenes de datos, sin que estos deban estar en una base de datos determinada. Esto se realiza mediante dos tipos de archivos: CSV y SHP.

5.1. PLANTILLAS GENÉRICAS

Para la realización del proyecto, debido a que varios componentes tienen la misma estructura, se han utilizado una serie de plantillas, para reducir el tiempo invertido en realizar cada uno de los componentes.

Para el actual prototipo, se han utilizado 3 plantillas diferentes. Estas son las siguientes:

5.1.1. PLANTILLA PARA LISTADOS

Este tipo de plantilla tiene la siguiente apariencia:

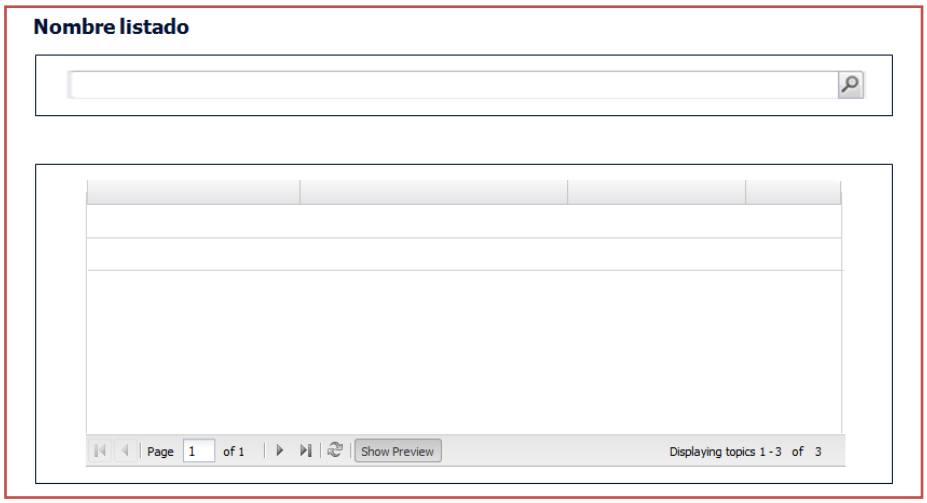


Ilustración 6 Plantilla listados

Esta plantilla es utilizada para realizar todos los componentes que implementen un listado con su respectiva funcionalidad de búsqueda.

Los componentes que hacen uso de dicha plantilla son los siguientes:

- **ListUserPanel:** Panel de usuarios.
- **ListGroupPanel:** Panel de grupos.
- **TemplateListDataPanel:** Panel de plantillas de datos.
- **WarehouseListDataPanel:** Panel de almacenes de datos.

5.1.2. PLANTILLAS DE CREACIÓN, EDICIÓN Y MUESTRA DE PERFILES

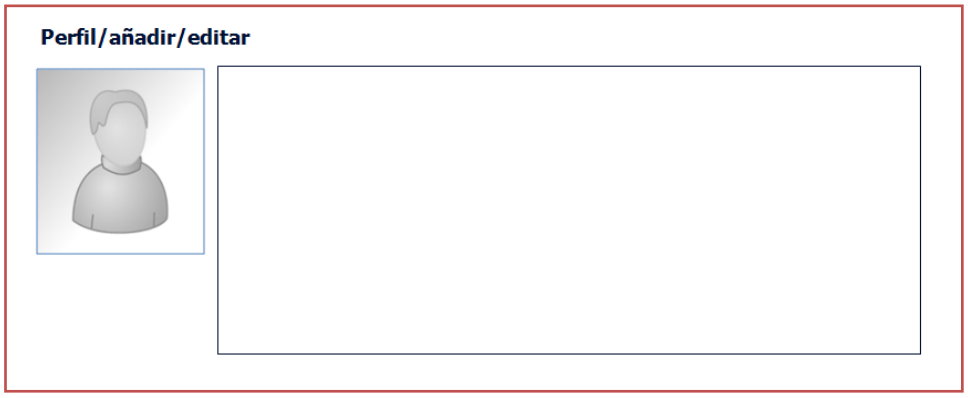


Ilustración 7 Plantilla creación, edición, muestra de perfiles

Esta plantilla es utilizada para la realización de todos los componentes que implementan la edición y visualización de datos o creación de nuevos objetos.

Los componentes que hacen uso de dicha plantilla son los siguientes:

- **AddUserPanel:** Panel para añadir usuarios.
- **AddGroupPanel:** Panel para añadir grupos.
- **ProfileUserPanel:** Panel de perfil de usuario.
- **ProfileGroupPanel:** Panel de perfil de grupo.
- **EditUserPanel:** Panel para editar usuarios.
- **EditGroupPanel:** Panel para editar grupos.

5.1.3. PLANTILLAS RELATIVAS A PERMISOS

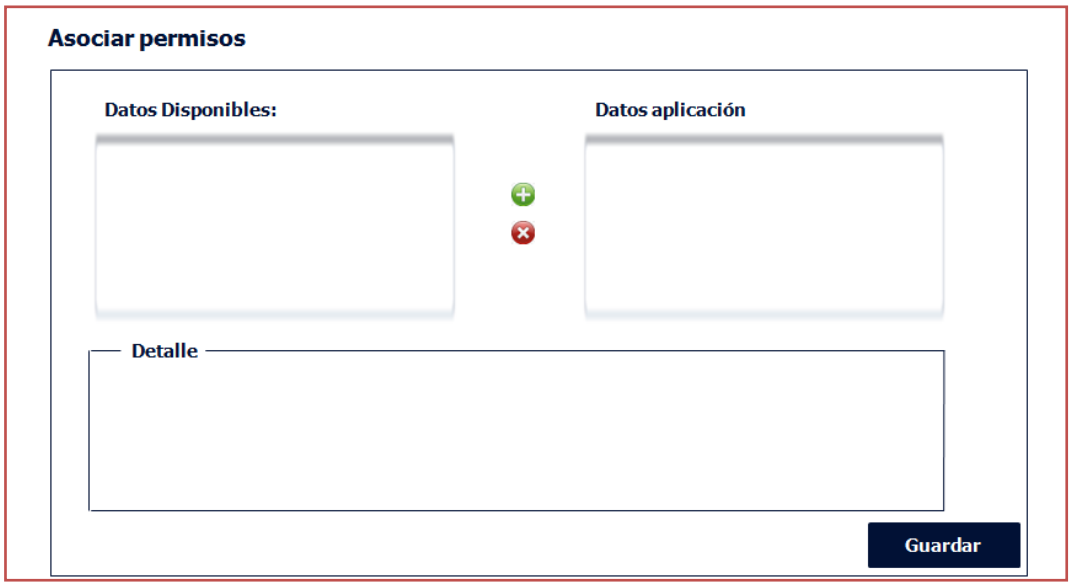


Ilustración 8 Plantilla permisos

Esta plantilla es utilizada para la realización de todos los componentes que implementen la asociación de permisos a nivel de aplicación móvil.

Los componentes que hacen uso de dicha plantilla son los siguientes:

- AssociateUserGroupPanel: Panel para asociar grupos y usuarios a la aplicación móvil.
- AssociateMapPanel: Panel para asociar mapas a la aplicación móvil.
- AssociateDataPaenl: Panel para asociar datos a la aplicación móvil.

5.2. Panel de Login y Archivo de configuración

Como ya se ha comentado anteriormente, el acceso al panel de administración se realiza mediante un panel de login. Dicho panel, se utilizo tanto para la aplicación como parte del aprendizaje y formación del proyecto. Se considera un elemento más del proyecto a pesar de que solo se integró la parte gráfica del panel y no así las funcionalidades. El aspecto de dicho panel es el siguiente:

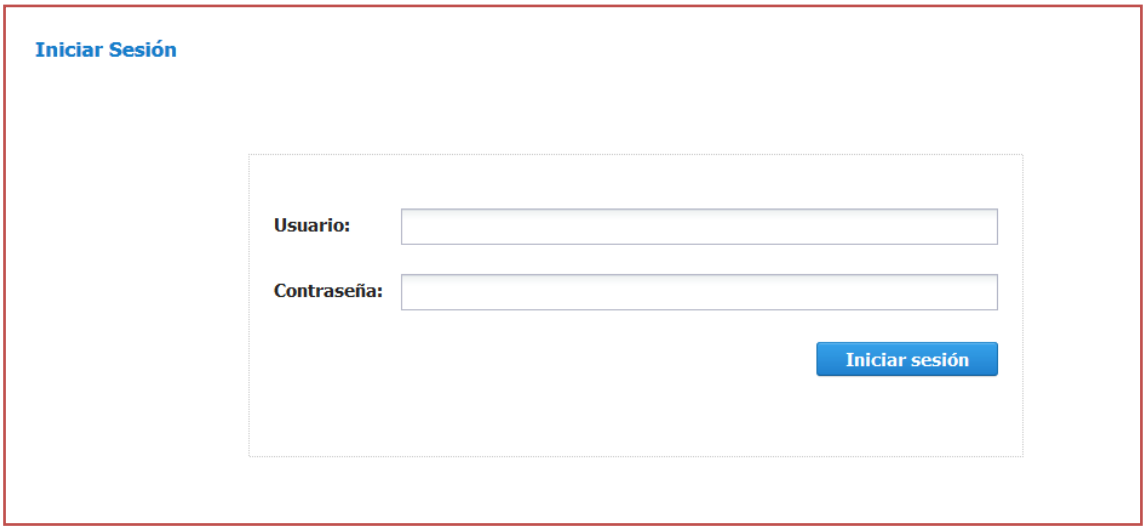


Ilustración 9 Interfaz panel de login

Por otro lado, el producto está basado en el desarrollo mediante componentes. Por medio de un archivo de configuración que se encuentra en formato JSON, se pueden seleccionar los módulos que el cliente necesite de una manera muy sencilla. El archivo de configuración es el que sigue:

```
{

    "application": [
        {
            "parameters": [
                {
                    "imgPrefix": [
                        {
                            "#text": "/admin"
                        }
                    ],
                    "windowname": [
                        {
                            "#text": "@window.name"
                        }
                    ],
                    "languages": [
                        {
                            "#attributes": {
                                "default": "ES"
                            },
                            "language": [
                                {
                                    "#attributes": {
                                        "name": "ES"
                                    },
                                    "#text": "Español"
                                },{
                                    "#attributes": {
                                        "name": "EU"
```

```
                            "#text": "Euskara"
                        }
                    ]
                }
            ]
        }
    ],
    "managers":[{

            "manager":[{
                "#attributes":{
                    "name": "adminPanelMng",
                    "className": "AdministrationPanelMng_c"
                }
            }]
    }],
    "header" : [{
            "#attributes" : {
                "id" : "header",
                "height" : "80",
                "class" : "custom-g-header"
            },
            "logo" : [{
                    "#attributes" : {
                        "title" : "@header.logo.title",
                        "url" : "javascript:void(0);",
                     },
                    "#text" : "/admin/img/GEOSTREET-ADMINISTRATION-PANEL.png"
                }
            ],
            "languageselector" : [
                {
                    "#attributes" :
                    {
                        "class" : "idiomas"
                    },
                    "#text" : "1"
                }
            ],
            "footer" : [{
                  "#attributes" : {
                        "id" : "footer",
                        "height" : "25px",
                        "class" : "custom-g-footer"
                  },
                  "labels" : [{
                              "#attributes" : {
                                    "text" : "@footer.geograma",
                                    "position" : "left"
                              }
                        },{
                              "#attributes" : {
                                    "text" : "@footer.contact",
                                    "position" : "left",
                                    "href" : "http://gis.geograma.com",
                                     "target" : " blank"
                              }
                        },{
                              "#attributes" : {
                                    "text" : "@footer.conditions",
                                    "position" : "left",
                                    "href" : "http://gis.geograma.com",
                                    "target": "_self"
                              }
                        },{
                              "#attributes" : {
                                    "text" : "@footer.privacy",
                                    "position" : "left",
                                    "href" : "http://gis.geograma.com",
                                    "target": "_self"
                              }
                        }
                  ]
```

```
"container": [
    {
        "#attributes": {
            "id": "container central usuariosGrupos",
            "position": "center",
            "header": "false",
            "default": true
        },
        "parameters": [
            {
                "collapsible": [
                    {
                        "#text": "false"
                    }
                ],
                "collapsed": [
                    {
                        "#text": "false"
                    }
                ],
                "split": [
                    {
                        "#text": "false"
                    }
                ],
                "type": [
                    {
                        "#text": "fit"
                    }
                ],
                "class": [
                    {
                        "#text": "gs-container"
                    }
                ],
                "title": [
                    {
                        "#text": "@container.userGroup"
                    }
                ]
            }
        ],
        "panel": [
            {
                "#attributes": {
                    "id": "GroupUsersPanel_c",
                    "className": "EmptyPanel_c"
                },
                "toolbar": [
                    {
                        "#attributes": {
                            "id": "groupUsersToolbar",
                            "position": "top"
                        },
                        "tools": [
                            {
                                "#attributes" : {
                                    "id" : "usersGroup",
                                    "className" : "GroupTool_c",
                                    "title": "@header.usuarios",
                                    "columns": 2,
                                    "headerPosition": "bottom"
                                },
                                "tools":[
                                    {
                                        "#attributes": {
```

Este solo es un fragmento de este archivo. En términos generales, la estructura de dicho archivo es la que siguiente:

- **Header:** Mediante esta etiqueta se puede configurar toda la cabecera de la aplicación.

- **Footer**: Mediante esta etiqueta se puede configurar todo el pie de la aplicación.

- **Container**: Esta etiqueta indica el container central de la aplicación y donde se encuentran los componentes.
    - Panel: Esta etiqueta permite crear pestañas con diferentes botoneras
        - Botones : Esta etiqueta permite configurar las botoneras de la aplicación.
            - Panel: Esta etiqueta permitie asociar los paneles de los diferentes componentes.

5.3. MÓDULO DE USUARIOS Y GRUPOS

El siguiente módulo se encarga de la administración de usuarios y grupos del panel de administración. Se comienza detallando el modelo de datos que corresponde a dicho módulo. Por otro lado se realiza un caso de uso para tener una visión general de las funcionalidades de dicho módulo. Por último, también se especifican todos los componentes y servicios que componen este módulo.

5.3.1. MODELO DE DATOS

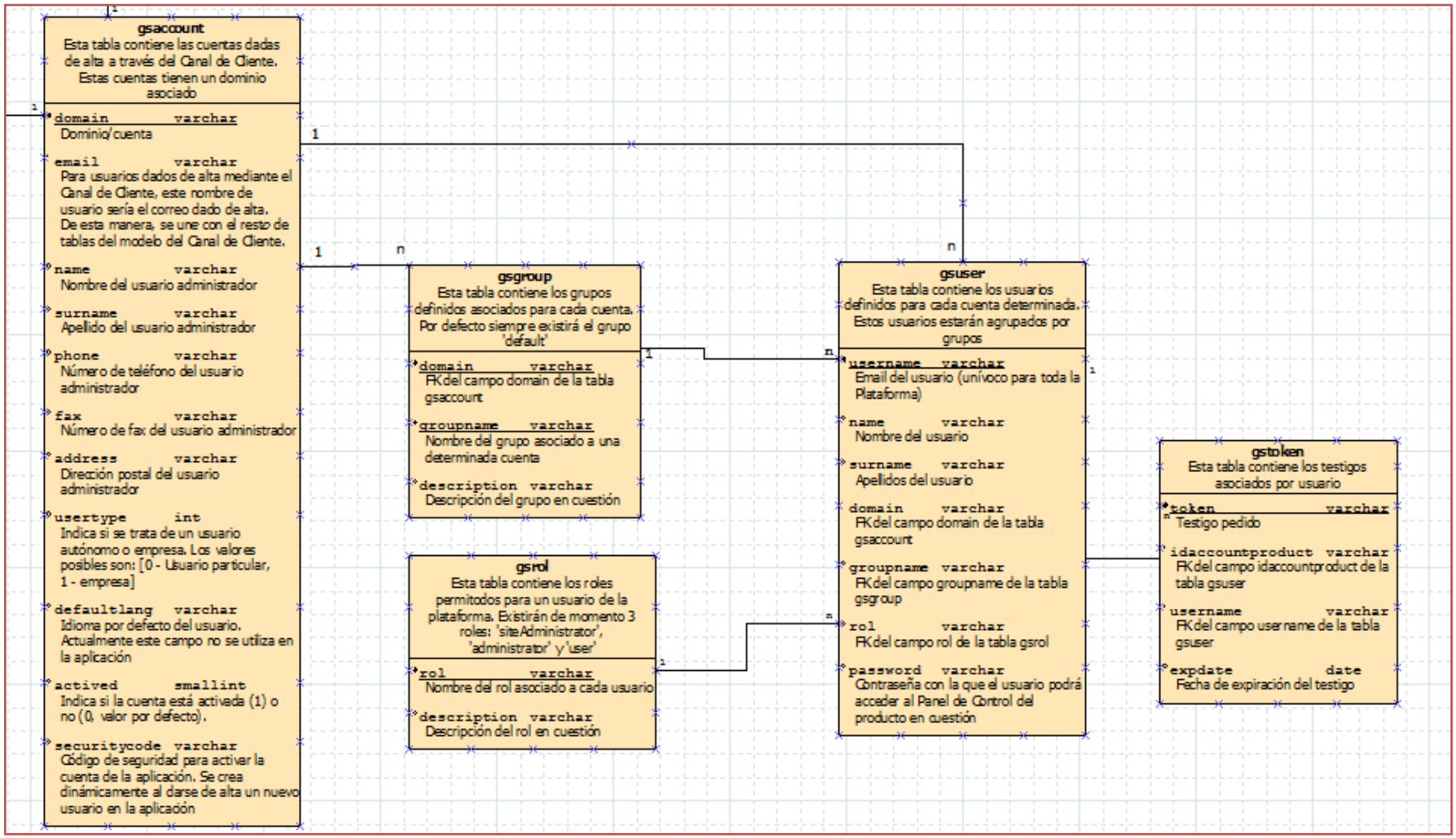


Ilustración 10 Módelo de datos módulo usuarios y grupos

5.3.2. CASO DE USO GENERAL

El caso de uso del módulo de usuarios y grupos es el que se puede ver a continuación:

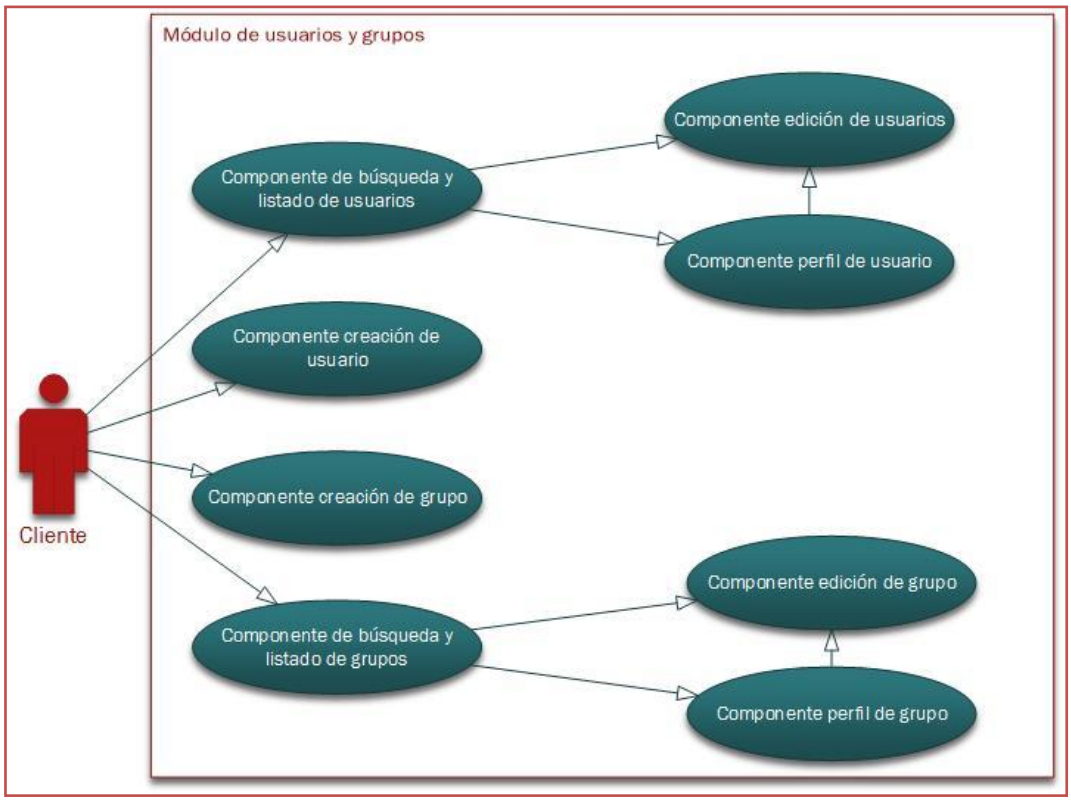


Ilustración 11 Caso de uso general modelo de usuarios y grupos

En los siguientes apartados, se especifica la funcionalidad de cada uno de los componentes identificados y los servicios utilizados para integrar cada una de las funcionalidades.

5.3.3. COMPONENTES

Los componentes identificados en el módulo de usuarios y grupos son los siguientes:

Relativos a la administración de usuarios:

Tipo componente	Nombre	Descripción
Componente de búsqueda y listado de usuarios	listUserPanel	El siguiente componente, se encarga de mostrar un listado de usuarios relacionados con una determinada cuenta correspondiente a la Plataforma GeoServicios Online. También se encarga de integrar la funcionalidad de búsqueda de dicho listado. Por otro lado, nos permite el borrado de usuarios y acceso a los componentes de edición y perfil de usuarios

Tipo componente	Nombre	Descripción
Componente de búsqueda y listado de usuarios	ListUserPanel	El siguiente componente, se encarga de mostrar un listado de usuarios relacionados con una determinada cuenta correspondiente a la Plataforma GeoServicios Online. También se encarga de integrar la funcionalidad de búsqueda de dicho listado. Por otro lado, nos permite el borrado de usuarios y acceso a los componentes de edición y perfil de usuarios
Componente edición de usuarios	EditUserPanel	El siguiente componente, se encarga de la edición de un determinado usuario
Componente perfil de usuarios	ProfileUserPanel	El siguiente componente, se encarga de mostrar el perfil de un determinado usuario. Por otro lado también permite acceder al componente de edición de usuarios EditUserPanel
Componente creación de usuarios	AddUserPanel	El siguiente componente, se encarga de la creación de nuevos usuarios

Tabla 1 Componentes módulo usuarios y grupos

Relativos a la administración de grupos:

Tipo componente	Nombre	Descripción
Componente de búsqueda y listado de grupos	ListGroupPanel	El siguiente componente, se encarga de mostrar un listado de grupos relacionados con una determinada cuenta correspondiente a la Plataforma GeoServicios Online. También se encarga de integrar la funcionalidad de búsqueda de dicho listado. Por otro lado, permite el borrado de grupos y acceso a los componentes de edición y perfil de grupos
Componente edición de grupos	EditGroupPanel	El siguiente componente, se encarga de la edición de un determinado grupo
Componente perfil de grupos	ProfileGroupPanel	El siguiente componente, se encarga de mostrar el perfil de un determinado grupo. Por otro lado también nos permite acceder al componente de edición de grupos EditGroupPanel
Componente creación de grupos	AddGroupPanel	El siguiente componente, se encarga de la creación de nuevos grupos

Tabla 2 Componentes módulo usuarios y grupos

Para una mayor especificación de cada uno de los componentes, con los casos de uso y diagramas de secuencia especificados por cada uno de ellos, se puede acceder al Anexo III: Componentes Módulo de usuarios y grupos.

5.3.4. SERVICIOS

Los servicios identificados en el módulo de usuarios y grupos son los siguientes:

Relativos a la administración de usuarios:

Tipo servicio	Nombre	Descripción
Servicio_añadir usuario	createUser	El siguiente servicio se encarga de la funcionalidad de añadir usuarios
Servicio modificar usuario	modifyUser	El siguiente servicio se encarga de la funcionalidad de modificar usuarios
Servicio borrar usuario	dropUser	El siguiente servicio se encarga de la funcionalidad de borrar usuarios
Servicio obtener usuarios	getUsers	El siguiente servicio se encarga de la funcionalidad de obtener los usuarios disponibles relacionados con una determinada cuenta de la Plataforma GeoServicios Online
Servicio buscar usuarios	searchUser	El siguiente servicio se encarga de la funcionalidad de búsqueda de usuarios dentro de un determinado listado de dichos usuarios

Tabla 3 Servicios módulo de usurios y grupos

Relativos a la administración de grupos:

Tipo servicio	Nombre	Descripción
Servicio_añadir grupo	createGroup	El siguiente servicio se encarga de la funcionalidad de añadir grupos
Servicio modificar grupo	modifyGroup	El siguiente servicio se encarga de la funcionalidad de modificar grupos
Servicio borrar grupo	dropGroup	El siguiente servicio se encarga de la funcionalidad de borrar grupos
Servicio obtener grupos	getGroups	El siguiente servicio se encarga de la funcionalidad de obtener los grupos disponibles relacionados con una determinada cuenta de la Plataforma GeoServicios Online
Servicio buscar grupos	searchGroup	El siguiente servicio se encarga de la funcionalidad de búsqueda de grupos dentro de un determinado listado de dichos usuarios

Tabla 4 Servicios módulo usuarios y grupos

Para una mayor especificación de cada uno de los servicios, con la especificación de las capas SOAP, SERVLET y JAVASCRIPT por cada uno de ellos, se puede acceder al Anexo IV; Servicios módulo de usuarios y grupos.

5.4. MÓDULO DE DATOS

El siguiente módulo se encarga de la administración de almacenes y plantillas de datos del panel de administración. Se comienza detallando el modelo de datos que corresponde a dicho módulo. Por otro lado se realiza un caso de uso general para tener una visión general de las funcionalidades de dicho módulo. también se especifican todos los componentes y servicios que componen este módulo.

Un almacén de datos hace referencia a los datos de conexión a una determinada tabla alojada en una base de datos. Estos almacenes están relacionados con una o más plantillas de datos.

Una plantilla de datos, contiene la información de una determinada tabla de una base de datos. Estas plantillas están asociadas a un almacén de datos en particular.

5.4.1. MODELO DE DATOS

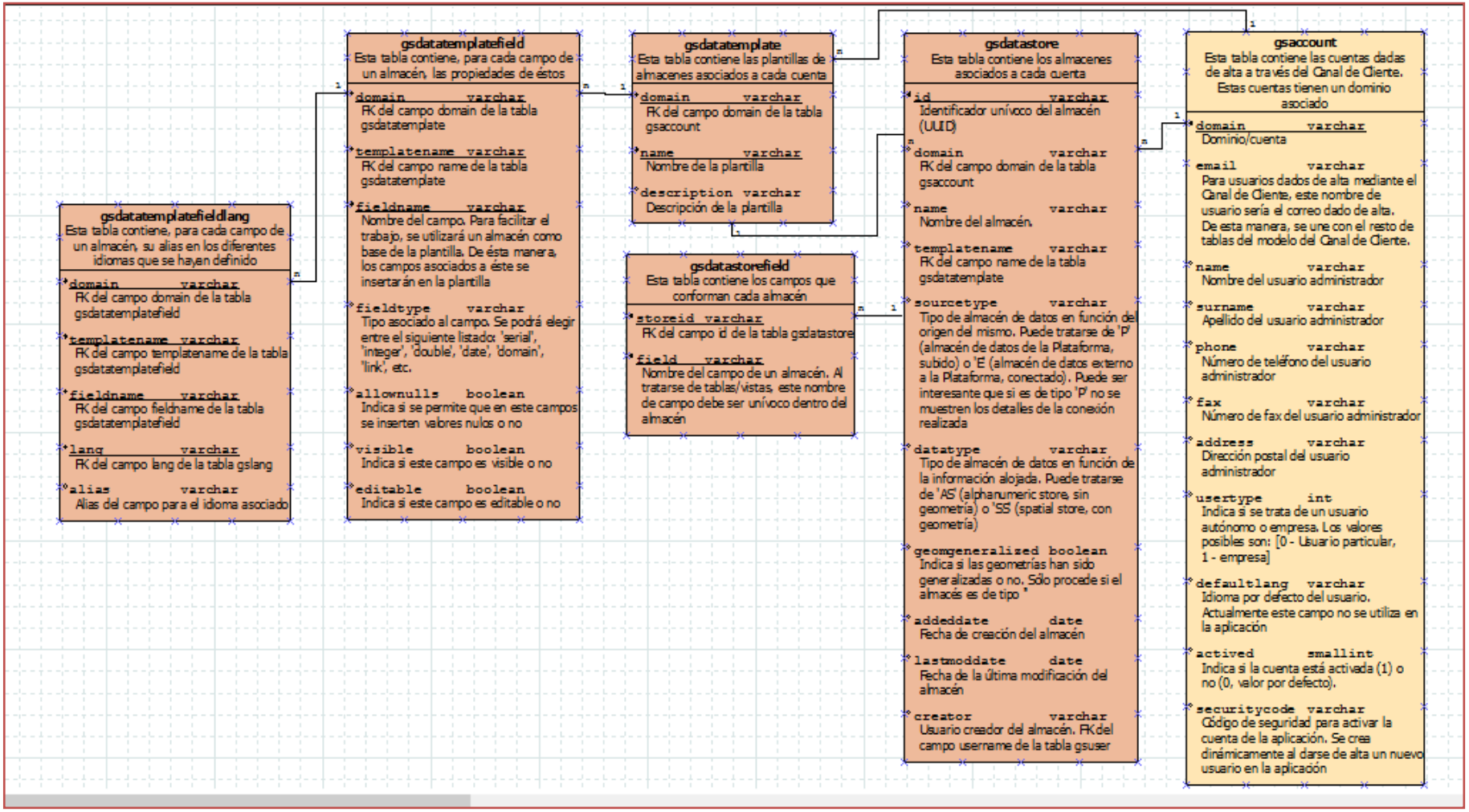


Ilustración 12 Modelo de datos módulo de datos

5.4.2. CASO DE USO GENERAL

El caso de uso del módulo de datos es el que se puede ver a continuación:

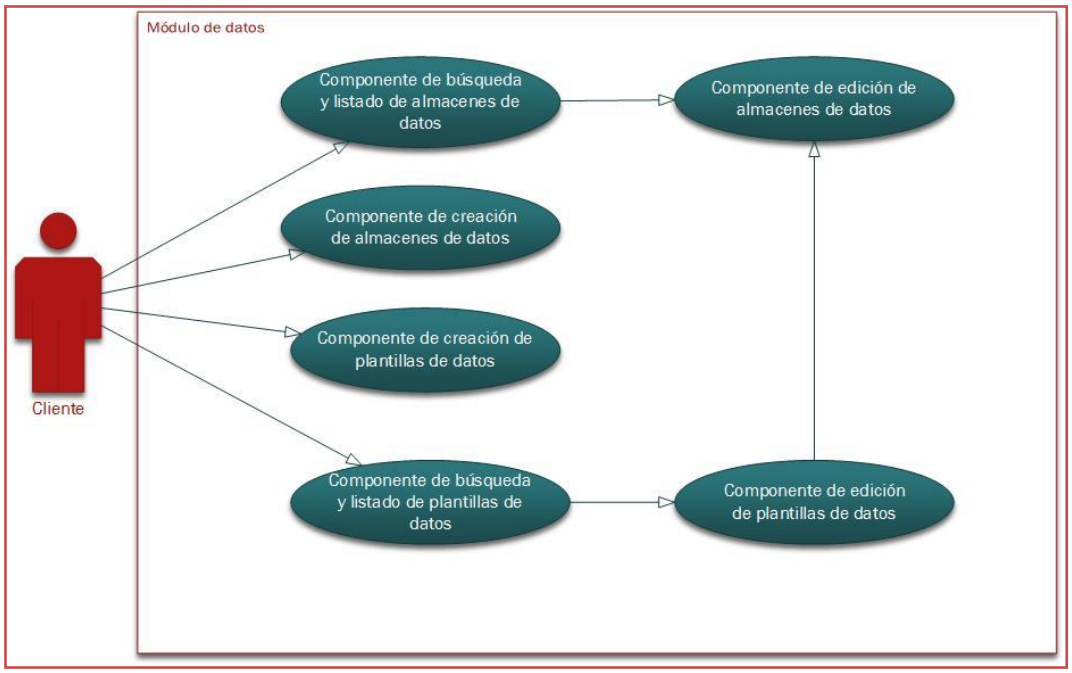


Ilustración 13 Caso de uso general módulo de datos

En los siguientes apartados, se especifica la funcionalidad de cada uno de los componentes identificados y los servicios utilizados para integrar cada una de las funcionalidades.

5.4.3. COMPONENTES

Los componentes identificados en el módulo de datos son los siguientes:
Relativos a la administración de almacenes de datos:

Tipo componente	Nombre	Descripción
Componente de búsqueda y listado de almacenes de datos	WarehouseListDataPanel	El siguiente componente, se encarga de mostrar un listado de almacenes de datos relacionados con una determinada cuenta correspondiente a la Plataforma GeoServicios Online. También se encarga de integrar la funcionalidad de búsqueda de dicho listado. Por otro lado, nos permite el borrado de almacenes de datos y acceso a el componente de edición de dichos almacenes de datos
Componente edición de almacenes de datos	WarehouseEditDataPanel	El siguiente componente, se encarga de la edición de un determinado almacén de datos
Componente creación de almacenes de datos	WarehouseConfigDataPanel	El siguiente componente, se encarga de la creación de nuevos almacenes de datos

Tabla 5 Componentes módulo de datos

Relativos a la administración de plantillas de datos:

Tipo componente	Nombre	Descripción
Componente de búsqueda y listado de plantillas de datos	TemplateListDataPanel	El siguiente componente, se encarga de mostrar un listado de plantillas de datos relacionados con una determinada cuenta correspondiente a la Plataforma GeoServicios Online. También se encarga de integrar la funcionalidad de búsqueda de dicho listado. Por otro lado, nos permite el borrado de plantillas de datos y acceso a el componente de edición de dichas plantillas de datos.
Componente edición de plantillas de datos	TemplateEditDataPanel	El siguiente componente, se encarga de la edición de una determinada plantilla de datos
Componente creación de plantillas de datos	TemplateCreateDataPanel	El siguiente componente, se encarga de la creación de nuevas plantillas de datos

Tabla 6 Componentes módulo de datos

Para una mayor especificación de cada uno de los componentes, con los casos de uso y diagramas de secuencia especificados por cada uno de ellos, se puede acceder al Anexo V: Componentes módulo de datos.

5.4.4. SERVICIOS

Los servicios identificados en el módulo de datos son los siguientes:

Relativos a la administración de almacenes de datos:

Tipo servicio	Nombre	Descripción
Servicio añadir almacén de datos	createDataStore	El siguiente servicio se encarga de la funcionalidad de añadir almacenes de datos
Servicio modificar almacén de datos	modifyDataStore	El siguiente servicio se encarga de la funcionalidad de modificar almacenes de datos
Servicio borrar almacenes de datos	dropDataStore	El siguiente servicio se encarga de la funcionalidad de borrar almacenes de datos
Servicio obtener almacenes de datos	getDataStores	El siguiente servicio se encarga de la funcionalidad de obtener los almacenes de datos disponibles relacionados con una determinada cuenta de la Plataforma GeoServicios Online
Servicio buscar almacenes de datos	searchDataStore	El siguiente servicio se encarga de la funcionalidad de búsqueda de almacenes de datos dentro de un determinado listado que contiene dichos almacenes de datos
Servicio obtener tablas de un almacén de datos	getTablesFromConn	El siguiente servicio se encarga de la funcionalidad de obtener las tablas asociadas a un determinado almacén de datos

Tipo servicio	Nombre	Descripción
Servicio obtener campos de un almacén de datos	getFieldsFromDataStore	El siguiente servicio se encarga de la funcionalidad de obtener los campos asociados a una determinada tabla de un almacén de datos

Tabla 7 Servicios módulo de datos

Relativos a la administración de plantillas de datos:

Tipo servicio	Nombre	Descripción
Servicio añadir plantilla de datos	createDataTemplate	El siguiente servicio se encarga de la funcionalidad de añadir plantillas de datos
Servicio modificar plantilla de datos	modifyDataTemplate	El siguiente servicio se encarga de la funcionalidad de modificar plantillas de datos
Servicio borrar plantilla de datos	dropDataTemplate	El siguiente servicio se encarga de la funcionalidad de borrar plantillas de datos
Servicio obtener plantillas de datos	getDataTemplate	El siguiente servicio se encarga de la funcionalidad de obtener las plantillas de datos disponibles relacionadas con una determinada cuenta de la Plataforma GeoServicios Online
Servicio buscar plantillas de datos	searchDataTemplate	El siguiente servicio se encarga de la funcionalidad de búsqueda de plantillas de datos dentro de un determinado listado que contiene plantillas de datos
Servicio obtener campos plantillas de datos	getFieldsFromConn	El siguiente servicio se encarga de la funcionalidad de obtener los campos asociados a una determinada plantilla de datos

Tabla 8 Servicios módulo de datos

Para tener un mayor detalle de cada uno de los servicios, con la especificación de las capas SOAP, SERVLET y JAVASCRIPT por cada uno de ellos, se puede acceder al Anexo VI: Servicios módulo de datos.

5.5. MÓDULO GEOASSET

El siguiente módulo se encarga de la administración de permisos sobre la aplicación móvil. Se comienza detallando el modelo de datos que corresponde a dicho módulo. Por otro lado se realiza un caso de uso general para tener una visión de las funcionalidades de dicho módulo. También se especifican todos los componentes y servicios que componen este módulo.

5.5.1. MODELO DE DATOS

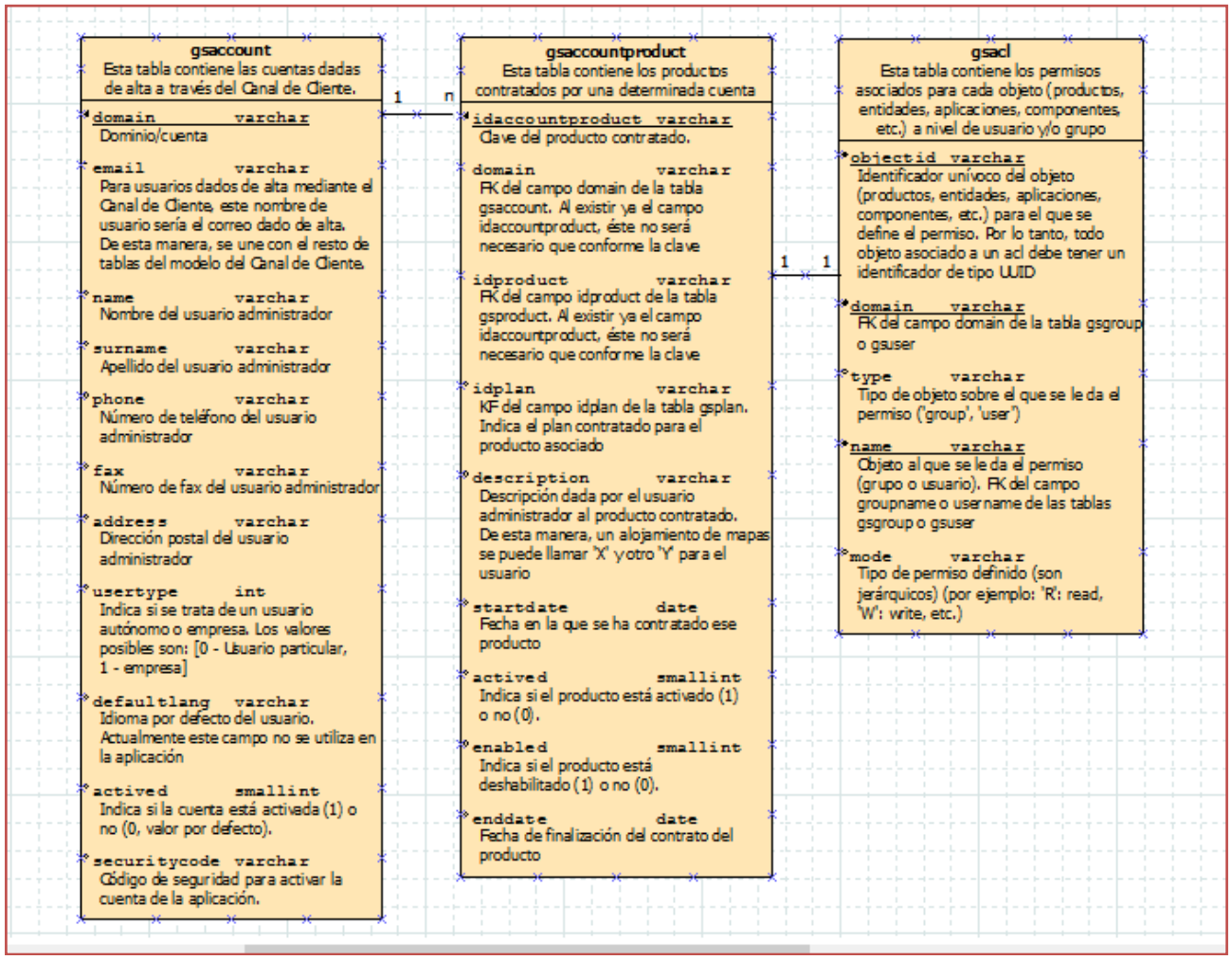


Ilustración 14 Modelo de datos módulo geoAsset

5.5.2. CASO DE USO GENERAL

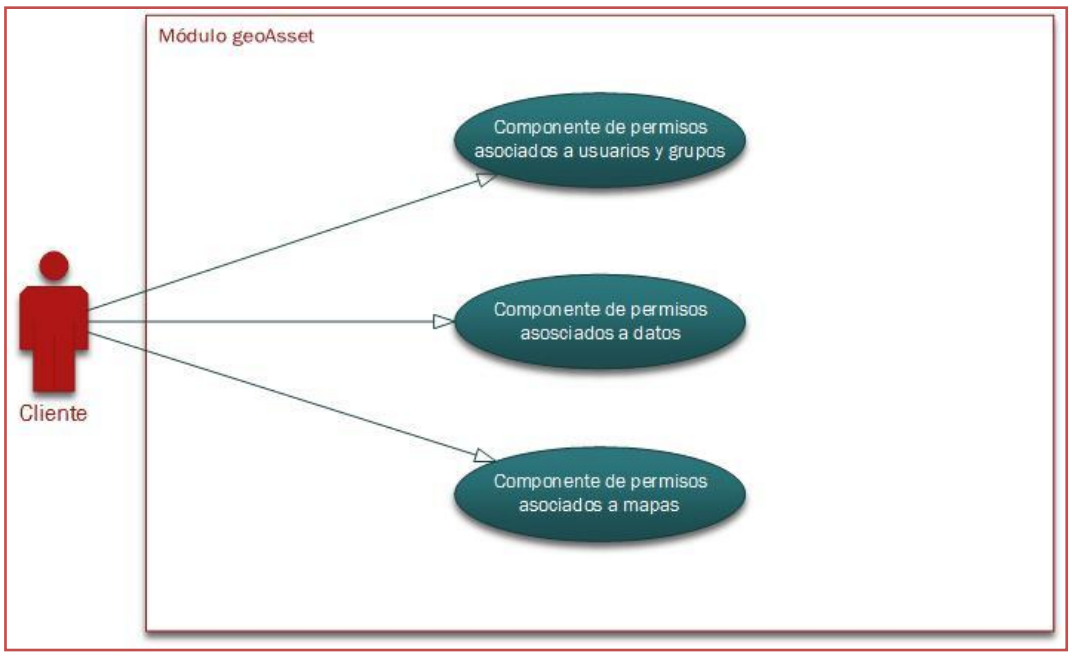


Ilustración 15 Caso de uso general módulo geoAsset

5.5.3. COMPONENTES

Los componentes identificados en el módulo geoAsset son los siguientes:

Tipo componente	Nombre	Descripción
Componente de permisos asociados a usuarios y grupos	AssociateUserGroupPanel	El siguiente componente, se encarga de mostrar cuatro listados. Por un lado muestra los listados de usuarios y grupos disponibles relacionados con una determinada cuenta correspondiente a la Plataforma GeoServicios Online. Por otro lado, muestra los otro dos listados que corresponden a los usuarios y grupos a los que se desea dar permisos para acceder a la aplicación móvil
Componente de permisos asociados a datos	AssociateDataPanel	El siguiente componente, se encarga de mostrar dos listados. Por un lado muestra el listado de almacenes de datos disponibles relacionados con una determinada cuenta correspondiente a la Plataforma GeoServicios Online. Por otro lado, muestra otro listado que corresponde a los almacenes de datos a los que se desea dar permisos para acceder a la aplicación móvil

Tipo componente	Nombre	Descripción
Componente de permisos asociados a mapas	AssociateMapPanel	El siguiente componente, se encarga de mostrar dos listados. Por un lado muestra el listado de mapas disponibles relacionados con una determinada cuenta correspondiente a la Plataforma GeoServicios Online. Por otro lado, muestra otro listado que corresponde a los mapas a los que se desea dar permisos para acceder a la aplicación móvil

Tabla 9 Componentes módulo GeoAsset

Para una mayor especificación de cada uno de los componentes, con los casos de uso y diagramas de secuencia especificados por cada uno de ellos, se puede acceder al AnexoVII: Componentes módulo geoAsset.

5.5.4. SERVICIOS

Los servicios identificados en el módulo geoAsset son los siguientes:

Tipo servicio	Nombre	Descripción
Servicio obtener permisos aplicación móvil	getAcl	El siguiente servicio se encarga de la funcionalidad de obtener los permisos asociados a la aplicación móvil
Servicio guardar permisos aplicación móvil	setAcl	El siguiente servicio se encarga de la funcionalidad guardar los permisos asociados a la aplicación móvil

Tabla 10 Servicios módulo GeoAsset

Para una mayor especificación de cada uno de los servicios, con la especificación de las capas SOAP, SERVLET y JAVASCRIPT por cada uno de ellos, se puede acceder al AnexoVIII: Servicios módulo geoAsset.

6. ADMINISTRATION PANEL: PROTOTIPO 2.0

En el siguiente apartado, se especifica como se ha realizado el prototipo 2.0 correspondiente al panel de administración. A este prototipo se le ha añadido la funcionalidad para realizar subidas de datos. Esta funcionalidad, no se encuentra dentro del producto realizado para el Ayuntamiento de Txingudi, si no que se ha realizado como un valor añadido para el producto propio de Geograma, con intenciones de poder ofrecerlo en un futuro

6.1. MEJORA MÓDULO DE DATOS

Como ya se ha comentado en el anterior apartado, el siguiente módulo se encarga de la administración de almacenes y plantillas de datos del panel de administración. En el caso de este segundo prototipo, se ha añadido un componente de subida de datos que complementa los componentes anteriormente comentados. Por lo tanto, solo es objeto de este apartado, la descripción del nuevo componente realizado. Se comienza detallando el modelo de datos que corresponde a dicho módulo. Por otro lado se ha realizado un caso de uso general para tener una visión general de las funcionalidades de dicho módulo. Se especifican el nuevo componente y los servicios creados para completar las funcionalidades de dicho componente.

6.1.1. MODELO DE DATOS

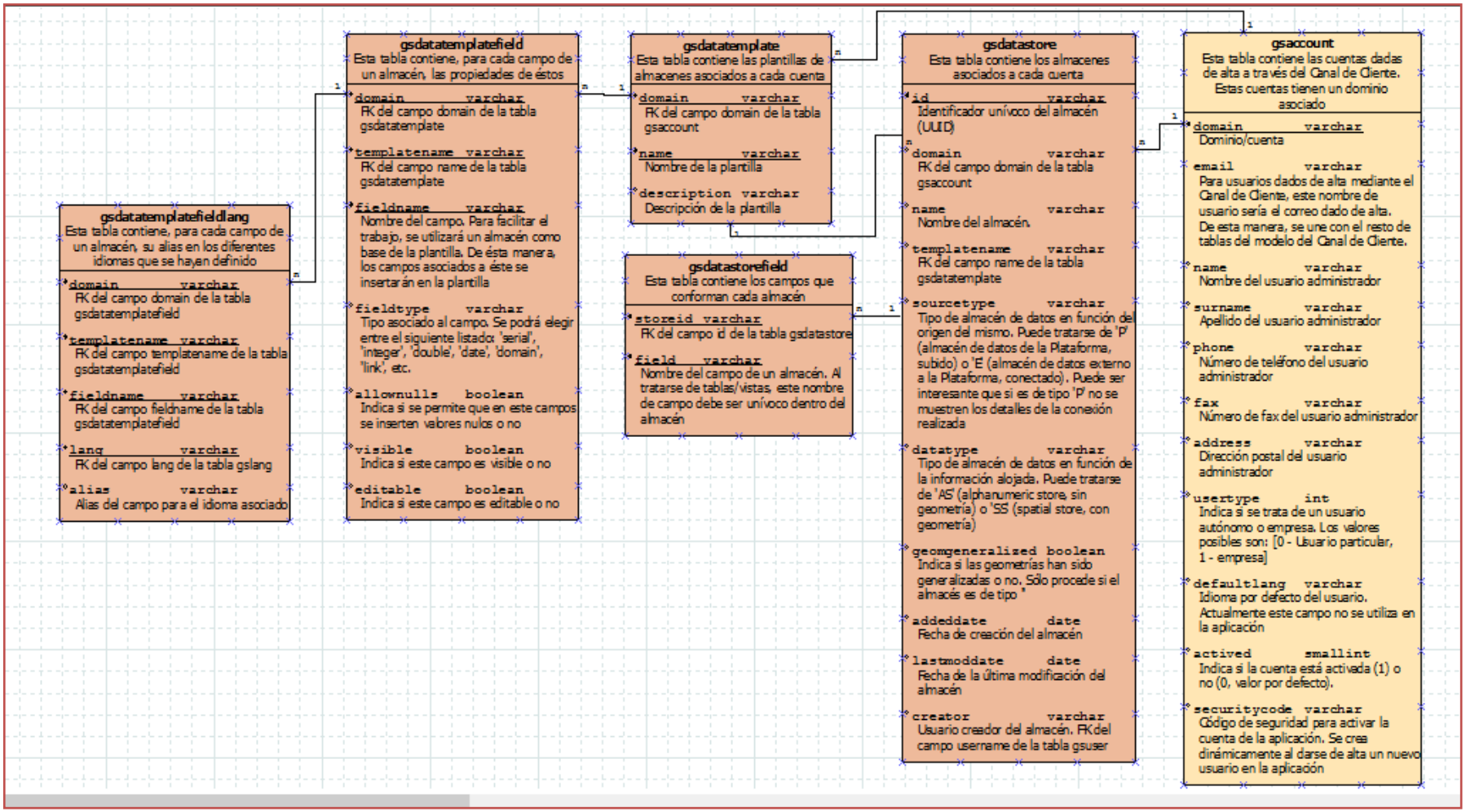


Ilustración 16 Modelo de datos mejora módulo de datos

6.1.2. CASO DE USO GENERAL

El caso de uso del módulo de datos es el que se puede ver a continuación:

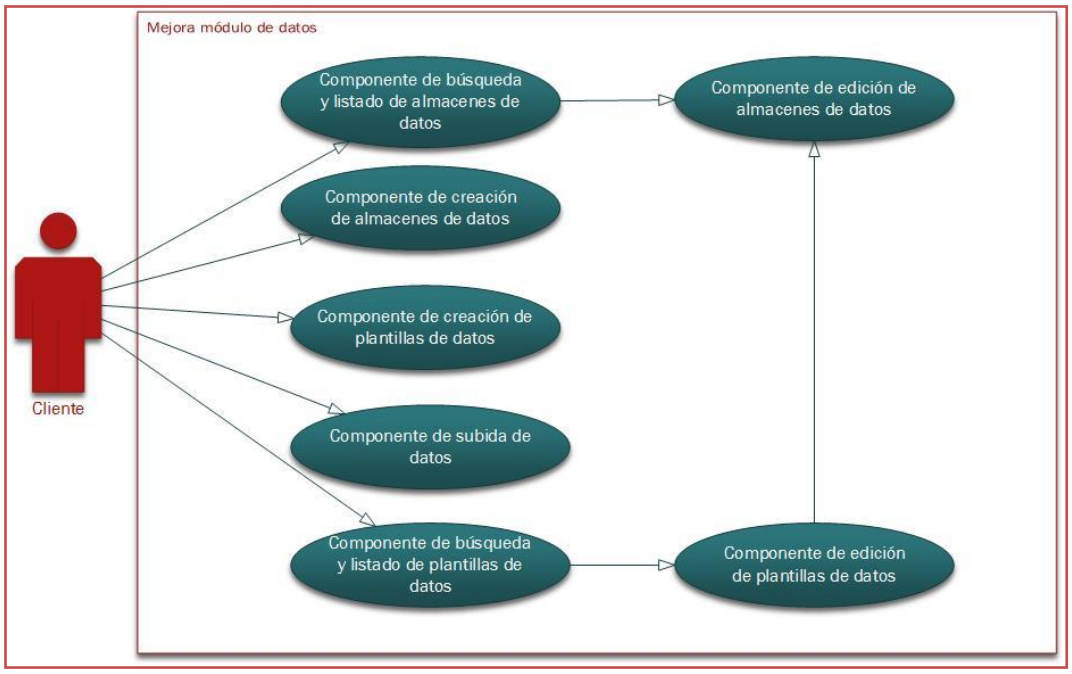


Ilustración 17 Caso de uso general mejora módulo de datos

En los siguientes apartados, se especifican la funcionalidad de cada uno de los componentes identificados y los servicios utilizados para integrar cada una de las funcionalidades. Como este apartado es la mejora de un módulo anteriormente especificado, se centra únicamente en el componente de subida de datos.

6.1.3. COMPONENTES

El componente identificado en el módulo mejora de datos es el siguiente:

Relativo a la administración de almacenes de datos:

Tipo componente	Nombre	Descripción
Componente de subida de datos	UploadDataPanel	El siguiente componente, se encarga de la subida de datos y creación de almacenes de datos relacionados con dichos datos relacionados con una determinada cuenta correspondiente a la Plataforma GeoServicios Online. Dicho componente, permite la subida de archivos de tipo CSV y SHP

Tabla 11 Componentes mejora módulo de datos

Para una mayor especificación de cada uno de los componentes, con los casos de uso y diagramas de secuencia especificados por cada uno de ellos, se puede acceder al. Anexo IX: Componentes mejora módulo de datos.

6.1.4. SERVICIOS

Los servicios identificados en el módulo de mejora de datos son los siguientes:

Relativos a la administración de almacenes de datos:

Tipo servicio	Nombre	Descripción
Servicio subir datos	uploadData	El siguiente servicio se encarga de la funcionalidad de subir archivos de datos. Estos pueden ser tanto archivos de tipo CSV como de tipo SHP
Servicio cargar datos	loadData	El siguiente servicio se encarga de la funcionalidad de carga de datos. Se encarga de leer los datos subidos y crear mediante dichos datos el almacén de datos correspondiente y una tabla en una determinada base de datos

Tabla 12 Servicios mejora módulo de datos

Para una mayor especificación de cada uno de los servicios, con la especificación de las capas SERVLET y JAVASCRIPT por cada uno de ellos, se puede acceder al Anexo X: Servicios mejora módulo de datos.

7. GESTIÓN DEL PROYECTO

En este capítulo, se muestra la gestión relativa al proyecto. Esta ha sido realizada a lo largo de el ciclo de vida del proyecto, en paralelo con el resto de actividades del producto.

Los apartados que se tratan son los siguientes:

- **Gestión del alcance:** Se detalla el alcance del proyecto y del producto, el EDT, las restricciones...
- **Gestión de actividades:** Se detallan las diferentes actividades y el cronograma.
- **Gestión de costes:** Se detalla una pequeña gestión de costes.
- **Gestión de calidad:** Se detalla cómo se ha realizado la gestión de la calidad, las diferentes pruebas y normativas seguidas.
- **Gestión de recursos humanos:** Se especifican los recursos humanos del proyecto.
- **Gestión de comunicaciones:** Se detalla cómo se han realizado las comunicaciones.
- **Gestión de riesgos:** Se especifican los diferentes riesgos localizados y como hacer frente a ellos.
- **Gestión de adquisiciones:** Se detallan las adquisiciones con las que se ha contado.

Cabe destacar, que el proyecto ha sido realizado en el ámbito de empresa, por lo que de la mayoría de las gestiones que se realizan en este apartado, se han encargado los miembros del equipo de dirección de dicho proyecto.

Por otro lado, la utilización del tiempo en pasado en este capítulo, hace referencia a que parte de la planificación se hizo con carácter previo al desarrollo del proyecto

7.1. GESTIÓN DEL ALCANCE

7.1.1. ALCANCE

7.1.1.1. Alcance del producto

Se realizará un panel de administración basado en prototipado que constará de dos versiones del mismo.

Se comenzará con un periodo de aprendizaje que constará de las siguientes tareas:

- Creación de un visor de mapas de prueba.
- Creación de un servicio web de prueba.

Relativos al Prototipo 1.0

- Una pantalla de login que permitirá el acceso a la aplicación
- Un archivo de configuración mediante el cual se podrá configurar la aplicación.
- Un módulo de usuarios y grupos, que permitirá gestionar grupos y usuarios. Tanto la creación, edición de los mismos como la visualización de estos:
    - Creación de usuarios y grupos.

  - Edición de usuarios y grupos.
  - Borrado de usuarios y grupos.

- Un módulo de datos que permitirá gestionar las fuentes de datos del cliente. Crear plantillas a partir de datos procedentes de BBDD propias, así como la edición de dichas plantillas:
  - Creación de plantillas y almacenes de datos.
  - Edición de plantillas y almacenes de datos.
  - Borrado de plantillas y almacenes de datos.

- Un módulo GeoAsset que permitirá configurar aplicaciones web o visores. Un visor tendrá asociado un mapa, una lista de control de accesos, etc.:
  - Asociar permisos de usuarios y grupos.
  - Asociar permisos de almacenes de datos.
  - Asociar permisos de mapas.

Relativos al Prototipo 2.0

- Mejora de módulo de datos que implementará además de las funcionalidades creadas en la anterior versión, una funcionalidad para permitir la subida de datos a la aplicación:
  - Subida y carga de datos de tipo CSV y SHP.

En cuanto al diseño de la aplicación web:

- Se enfocará hacia tonos azulados, siguiendo el estilo que proporciona la interfaz gráfica de dropbox.

7.1.1.2. Alcance del proyecto

El alcance del proyecto engloba el alcance del producto y los siguientes puntos:

- Se realizará un plan de proyecto conteniendo:
  - Antecedentes del proyecto.
  - Objetivos del proyecto.
  - Gestión del alcance.
  - Gestión de actividades.
  - Gestión de costes.
  - Gestión de calidad.
  - Gestión de recursos humanos.
  - Gestión de comunicaciones
  - Gestión de riesgos
  - Gestión de adquisiciones.
- Se realizará un seguimiento diario de las tareas realizadas

7.1.2. RESTRICCIONES Y LIMITACIONES

- La aplicación web deberá ser correctamente visualizable en los siguientes navegadores:
  - Internet Explorer 9 y posteriores.
  - Firefox 24 y posteriores.
  - Chrome 30 y posteriores.

7.1.3. CONTROLAR EL ALCANCE

El control del alcance será responsabilidad del equipo de dirección de proyecto. En caso de abordar líneas de mejoras en un momento dado, esto será comunicado por dicho equipo.

7.1.4. MODIFICACIÓN DEL ALCANCE

Durante el ciclo del proyecto, el alcance sufrió una modificación importante correspondiente al Prototipo 2.0. Inicialmente, dicho prototipo constaba de los siguientes apartados:

- Mejora módulo de datos
- Módulo de aplicación web

Debido a que la mejora del módulo de datos, fue más complejo de lo previamente estimado, se decidió por parte del equipo de dirección del proyecto suprimir el módulo de aplicación web y centrarse en el módulo de mejora de datos.

7.1.5. EDT

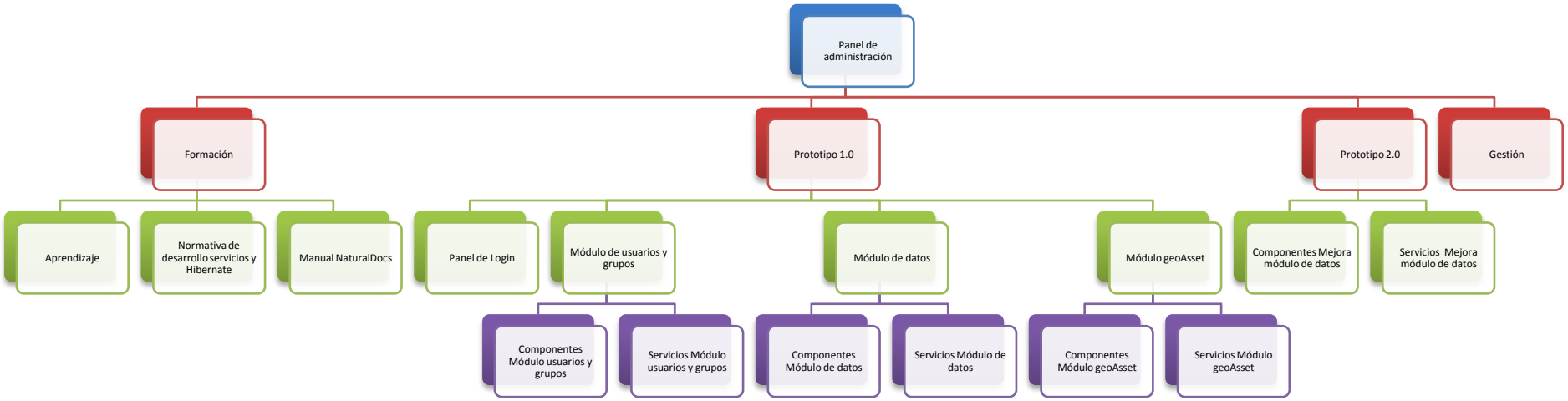


Ilustración 18 EDT

7.1.6.DICCIONARIO DE LA EDT

7.1.6.1. Formación

- Aprendizaje:
  - En esta etapa se realizan varias tareas para familiarizarse con las tecnologías a utilizar.
- Normativa de desarrollo servicios y Hibernate:
  - Esta etapa conlleva la redacción de la normativa de desarrollo relacionada con el desarrollo de servicios y la utilización de Hibernate.
- Manual NaturalDocs
  - Esta etapa conlleva la redacción de un manual para la utilización de NaturalDocs para la creación de documentación relativa a los servicios realizados.

7.1.6.2. Prototipo 1.0

- Panel de login: Este panel se encarga del acceso a la aplicación.

- Módulo de usuarios y grupos
  - Servicios Módulo de usuarios y grupos
    - Este bloque contiene la especificación de los servicios relativos al módulo de usuarios y grupos.
  - Componentes Módulo de usuarios y grupos
    - Este bloque contiene el diseño y especificación de los componentes relativos al módulo de usuarios y grupos.
- Módulo de datos
  - Servicios Módulo de datos
    - Este bloque contiene la especificación de los servicios relativos al módulo de datos.
  - Componentes Módulo de datos
    - Este bloque contiene el diseño y especificación de los componentes relativos al módulo de datos.
- Módulo geoAsset
  - Servicios Módulo geoAsset
    - Este bloque contiene las especificaciones de los servicios relativos al módulo geoAsset.
  - Componentes Módulo geoAsset
    - Este bloque contiene el diseño y especificación de los componentes relativos al módulo geoAsset.

7.1.6.2. Prototipo 2.0

- Servicios Mejora módulo de datos
  - Este bloque contiene la especificación de los servicios relativos a la mejora del módulo de datos.
- Componentes Mejora módulo de datos
  - Este bloque contiene el diseño y especificación de los componentes relativos a la mejora del módulo de datos.

7.1.6.3 Gestión

- Este apartado contiene los apartados relativos a la gestión del proyecto.

7.2. GESTIÓN DE ACTIVIDADES

7.2.1. GESTIÓN DE ACTIVIDADES

Para una mejor comprensión del desarrollo y ciclo de vida del proyecto, se especifican los hitos más significativos en la siguiente tabla:

Hito	Fecha
Propuesta inicial del proyecto	02/07/2013
Aprendizaje de tecnologías a utilizar	02/07/2013
Finalización Prototipo 1.0	30/09/2013
Finalización Prototipo 2.0	22/10/2013
Entrega documentación del proyecto	05/10/2013

Tabla 13 Actividades

A continuación se muestran una serie de gráficos para analizar el desarrollo del ciclo de vida del proyecto, teniendo en cuenta cada uno de los bloques más importantes del mismo.

Se comienza analizando el siguiente grafico:

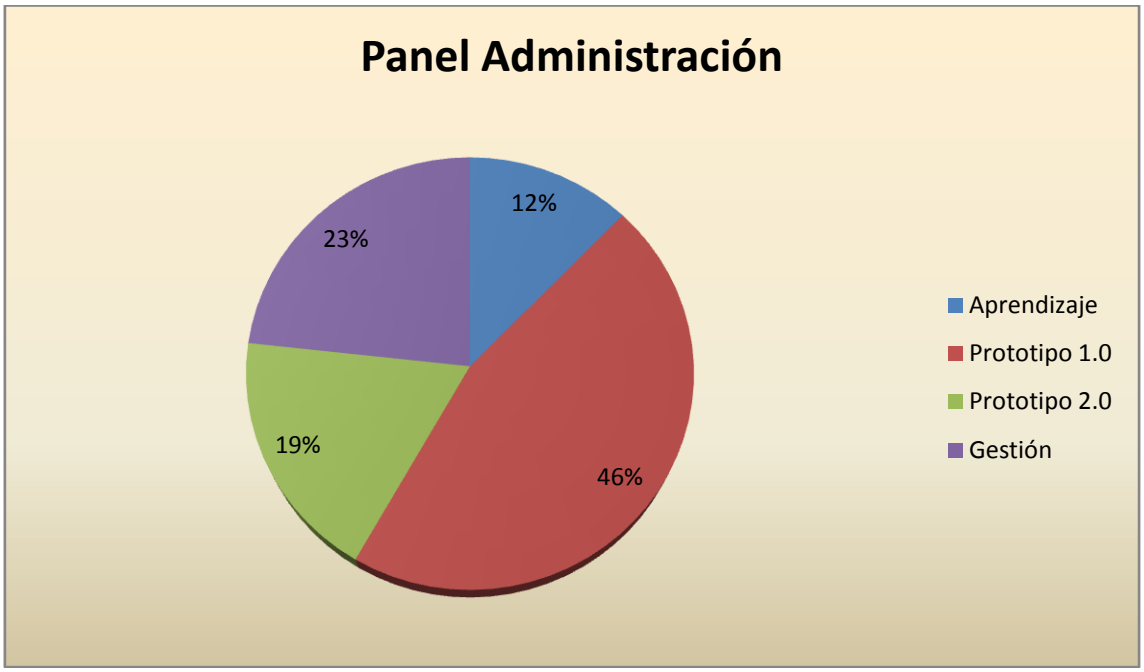


Ilustración 19 Grafico fases panel de administración

Como se puede observar en el gráfico anterior, de las 600 horas correspondientes al proyecto, prácticamente el 65% se han destinado a el desarrollo del producto. El 35% restante, se ha dividido tanto en formación como en la gestión de dicho proyecto.

Tras esto, en se analiza en profundidad el prototipo 1.0, para ver detalladamente el esfuerzo realizado en cada uno de los módulos de dicho prototipo:

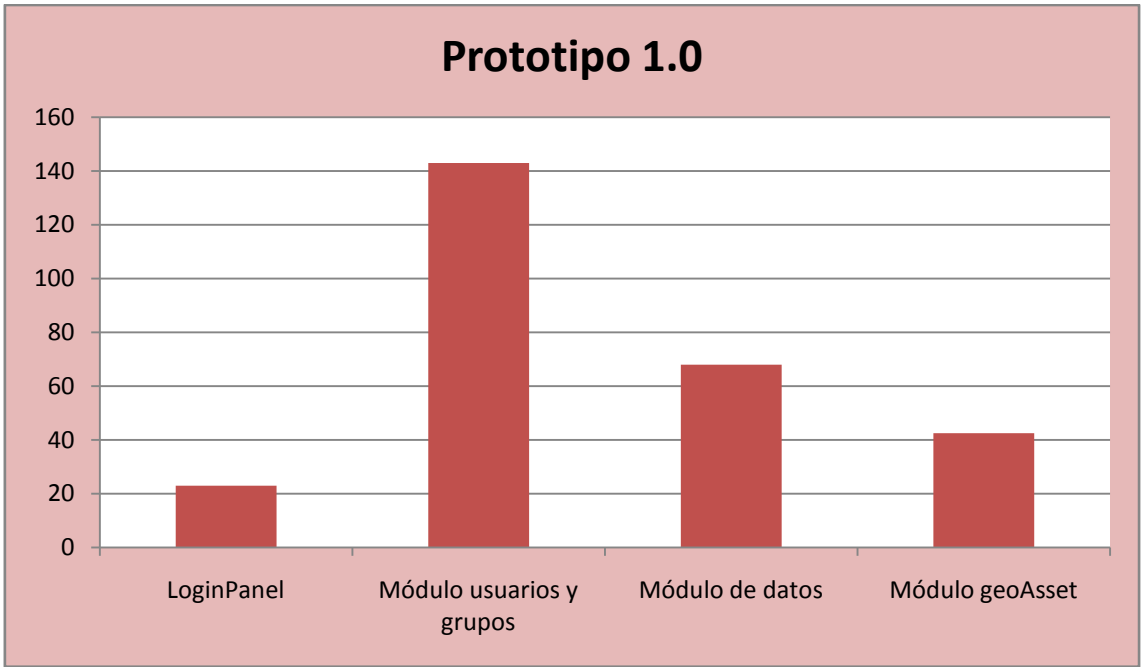


Ilustración 20 Gráfico fases prototipo 1.0

El gráfico anterior se ha centrado en el desarrollo del prototipo 1.0, para poder destacar la curva de aprendizaje. En este caso, se puede ver muy claramente en el gráfico anterior, dejando a un lado el desarrollo del panel de login. Teniendo en cuenta que la funcionalidad de los 3 módulos ha sido parecida, el esfuerzo realizado a la hora de implementar los diferentes módulos ha ido en descenso a lo largo de el ciclo de vida del proyecto.

Esta curva se puede observar más claramente en el siguiente gráfico:

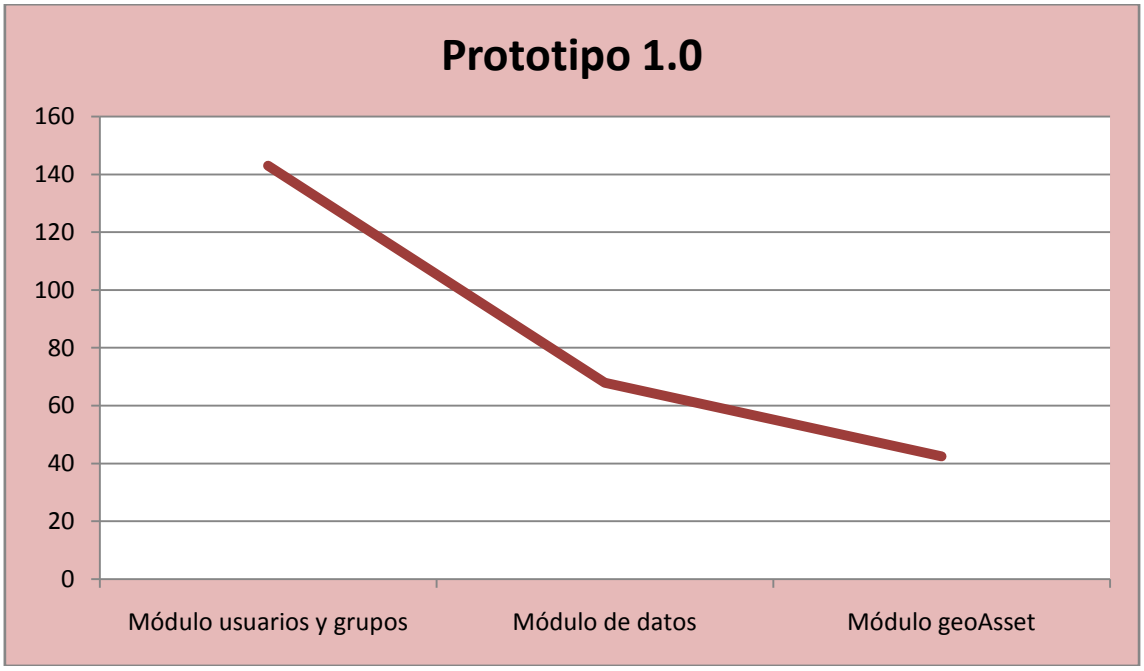


Ilustración 21 Curva de aprendizaje prototipo 1.0

El esfuerzo total del proyecto ha sido de 600 horas con una duración comprendida entre el 02/07/2013 al 04/10/2013.

En el Anexo XI: Seguimiento Diario se puede ver el trabajo diario en cada una de las tareas del proyecto.

7.2.2. CRONOGRAMA

Estas han sido las dependencias de las actividades más identificativas del proyecto.

- Se han realizado secuencialmente las siguientes actividades
  - Aprendizaje: esta actividad comprende el aprendizaje de las tecnologías a utilizar.
  - Prototipo 1.0: realización de los componentes del prototipo 1.0 del panel de administración.
  - Prototipo 2.0: realización del componente del prototipo 2.0 del panel de administración.

- En paralelo se ha realizado la siguiente actividad
  - Gestión: la actividad que comprende la gestión del proyecto y del producto.

7.3. GESTIÓN DE COSTES

7.3.1. COSTES

Para realizar la estimación de costes, se ha utilizado los datos suministrados por la empresa Geograma sobre el coste por hora de un programador Web estandar. Los costes se pueden ver en la siguiente tabla:

Categoría	Precio /Hora	Esfuerzo realizado	Total
Programador Web	15€/hora	450 horas	6.750 €
Total		**450 horas**	**6.750€**

Tabla 14 Gestión de costes

7.4. GESTIÓN DE CALIDAD

En este apartado, se describen los métodos utilizados para el aseguramiento de la calidad.

En lo relativo al producto:

- A la hora de realizar los servicios del producto, se ha seguido una normativa de desarrollo, que se puede ver en el Anexo II: Normativa y desarrollo de servicios y Hibernate, y se han documentado dichos servicios utilizando Natural Docs, que se puede ver en el Anexo I: Manual de Natural Docs.

- En cuanto a las interfaces del producto, el framework de ExtJs nos asegura una correcta visualización en los navegadores Chrome, Firefox y Explorer.

- Todos los componentes han sido sometidos a diferentes pruebas para comprobar el correcto funcionamiento de las diferentes funcionalidades de dichos componentes.

En lo relativo al proyecto:

- Se exige que todos los documentos sigan una estética con tonos rosados y tengan el formato .docx de Microsoft. También se deben seguir en la medida de lo posible las normas ortográficas y gramaticales de la Real Academia de la lengua Española.

7.5. GESTIÓN DE RECURSOS HUMANOS

El equipo de proyecto está compuesto por 1 trabajador que ha realizado las tareas de programador web.

Nombre	Rol
Hodei	Programador web

Tabla 15 Gestión de recursos humanos

Los costos del proyecto computan en base a el rol de programador web.

7.5.1. FORMACIÓN

Durante el comienzo del proyecto, se hacen efectivas las siguientes tareas de formación, en pos de mejorar las competencias con respecto a las tareas que se van a realizar. A continuación se detalla la formación recibida:

Nombre	Formación	Tiempo
Hodei	• Desarrollo visor de prueba • Desarrollo servicio de prueba	74 horas

Tabla 16 Formación

La formación recibida se puede observar más detalladamente en el apartado comentado anteriormente Aprendizaje.

7.5.2. MONITORIZACIÓN DEL AVANCE

Los miembros del equipo de dirección son los encargados de controlar y monitorizar el avance del proyecto.

7.6. GESTIÓN DE COMUNICACIONES

Se ha realizado un análisis de los posibles interesados, a partir del cual se ha obtenido la lista que se detalla a continuación.

- Equipo
  - Hodei López
  - Stakeholders
- Clientes
  - Ayuntamiento de Txingudi
- Proveedores de servicios
  - Amazon

7.6.1. GESTIÓN DE LAS COMUNICACIONES

Equipo

De entre los interesados, se ha mantenido una constante comunicación entre miembros del equipo. Las vías de comunicación principales han sido las siguientes:

- Email: por medio de Gmail y el correo interno de Geograma.
- Subversion Server - Control de versiones

Como se ha indicado, las comunicaciones entre los miembros del proyecto, se han realizado por email. Estas pueden ser tanto de índole informativo como un listado de las diferentes incidencias detectadas a solucionar.

Para realizar control de versiones del código, se ha utilizado subversión. Se ha tenido que actualizar la versión todos los días a primera hora y ha habido que guardarla a última hora.

La estructura para el almacenamiento de documentos es la siguiente:

- Software Base
    - Application: En esta carpeta se encuentran todas las aplicaciones.
        - Services: En esta carpeta se encuentran todos los servicios.
            - AdministrationPanel: En esta carpeta se encuentran todos los servicios relativos a el panel de administración.
        - Administration panel: En esta carpeta se encuentran todos los componentes de el panel de administración.

En cuanto a la estructura para localizar la información acerca de la estructura de el modelo de datos:

- Desarrollo
    - Software
        - Base de datos
            - Modelo de datos: El modelo de datos de la plataforma
            - Datos: Script con el contenido de la base de datos.

Clientes

De la comunicación con el cliente se encarga el equipo de dirección del proyecto.

Proveedores de Servicios

De la comunicación con los proveedores de servicios se encarga el equipo de dirección de proyecto.

7.7. GESTIÓN DE RIESGOS

Una vez identificados los riesgos del proyecto se procede a priorizar los mismos para poder realizar un plan frente a cada uno de ellos. Los riesgos identificados y ordenados por prioridad han sido los siguientes:

CUMPLIMIENTO DE PLAZOS

Se podría catalogar como el riesgo más prioritario del proyecto. El proyecto tiene que tener unos plazos de entrega específicos que tienen como fecha límite el 30/09/2013 en caso de la entrega del prototipo 1.0 y 23/10/2013 en caso del prototipo 2.0.

En todo momento, el equipo de dirección puede proponer el trabajar por objetivos ampliando el margen de 40 horas semanales. Por otro lado, se reducirá la optimización de funcionalidades en caso de que no se llegue a los plazos estipulados.

DESCONOCIMIENTO DE LAS TECNOLOGÍAS A UTILIZAR

El no tener experiencia previa a la hora de desarrollar el producto con las tecnologías expuestas, puede generar que las estimaciones iniciales no sean buenas y que esto lleve a un fracaso del proyecto.

Para poder hacer frente a este riesgo, se ha introducido una fase inicial de formación. Esto lo podemos ver en el apartado Aprendizaje.

CORRECTO FUNCIONAMIENTO DE LOS SERVIDORES

Uno de los riesgos más importantes del proyecto. Para poder paliar esta situación, se cuenta con varios servidores de respaldo, para que no se produzca en ningún momento una parada en la producción y consecución del proyecto.

PERDIDA DE PERSONAL CLAVE

En caso de que una persona del equipo del proyecto tenga que ausentarse en cualquier momento, se realizará una división de las tareas de esta persona para asignárselas al personal disponible. Esto es posible si se instaura una visión conjunta de lo que se está realizando en el proyecto.

CAMBIOS EN LAS PRIORIDADES

En caso de que se produzcan cambios en las prioridades del proyecto, el equipo de dirección se reunirá para realizar una pequeña replanificación de las tareas a realizar para poder hacer frente a estos cambios.

7.8. GESTIÓN DE ADQUISICIONES

A continuación, se detalla la lista de las necesarias adquisiciones. En este caso los encargados de seleccionar dichas adquisiciones han sido los miembros de dirección del proyecto.

Descripción	Detalle	Vendedor	Precio
Servidores	Servidores de desarrollo y producción	Amazon	Costo por uso
Iconos	Iconos utilizados en la aplicación	Axialis Software	60€
Core Framework	Core utilizado para generar la base de la aplicación	Geograma	

Tabla 17 Gestión adquisiciones

Hay que indicar, que a la hora de generar la aplicación web, la base de dicha aplicación, así como la gestión de idiomas, es responsabilidad del Core Framework, ya realizado anteriormente por la empresa Geograma.

Por otro lado, se han utilizado servidores alojados en Amazon, tanto para el desarrollo de la aplicación, como para la puesta en producción de dicha aplicación.

Por último, se han adquirido 2 paquetes de iconos que se han utilizado en el panel de administración.

8. CONCLUSIONES

En el siguiente apartado, se exponen las conclusiones más significativas del proyecto. Estas se han dividido en diferentes ámbitos que son los siguientes:

- **Conclusiones tecnológicas**: En este apartado se tratan los aspectos más relevantes de las tecnologías utilizadas.
- **Conclusiones relativas a las comunicaciones**: En este apartado se tratan aspectos relacionados con la comunicación entre los miembros del proyecto.
- **Conclusiones relativas al producto:** En este apartado se tratan aspectos relacionados con el producto.
- **Conclusiones relativas al trabajo en equipo:** En este apartado se tratan las aspectos más importantes dentro del trabajo en equipo.
- **Conclusiones relativas al proyecto:** En este apartado se tratan las conclusiones relativas a el proyecto.

8.1. CONCLUSIONES TECNOLÓGICAS

USO DEL FRAMEWORK SENCHA EXTJS

El uso de este framework a la hora de realizar aplicaciones que tengan un ámbito administrativo, ayuda notablemente al ahorro de diseño y especificación de elementos, y por lo tanto, al esfuerzo a realizar para generar interfaces gráficas.

El framework tiene una extensa API donde podemos encontrar cualquier duda que nos surja, así como foros de ayuda muy completos.

USO DE HIBERNATE

Esta herramienta nos ofrece variedad de ventajas a la hora de utilizar bases de datos. Te da una total independencia con el motor de base de datos ya que tenemos totalmente independiente la capa de datos con la lógica de negocio. Con cambiar una línea de un fichero de conexión ya tendremos la aplicación bajo otro motor de base de datos.

Otra de las grandes ventajas que nos ofrece, es el cacheo de datos. Esto hace que el primer acceso sea costoso, pero que los sucesivos accesos tengan un costo mínimo.

USO DE NOTEPAD++

La herramienta que personalmente, más me ha sorprendido y recomiendo. Se pueden destacar las siguientes características:

- Identifica los lenguajes de programación más habituales y gracias a ello ofrece una presentación ordenada y clara de código.
- Permite abrir archivos con cualquier extensión.
- Indica los números de línea.
- Permite trabajar con múltiples archivos abiertos.
- Tiene infinidad de plugins para añadirle diferentes funcionalidades.

8.2. CONCLUSIONES RELATIVAS A LAS COMUNICACIONES

USO DE SUBVERSION

El uso de esta herramienta, permite tener un control total a la hora de trabajar varias personas en los mismos proyectos. Mediante este control de versiones, podemos tener copias de respaldo continuas y compartir información de manera muy sencilla. A la hora de trabajo colaborativo se antoja indispensable.

Por otro lado, nos ofrece un historial de cambios realizados, por lo que rápidamente cada uno de los componentes del equipo puede observar las modificaciones realizadas por cada uno de los miembros del proyecto.

NORMATIVAS DE DESARROLLO

El utilizar normativas de desarrollo muy definidas, ayuda a que todos los componentes del equipo del proyecto sepan en cualquier momento identificar diferentes tipos de información:

- Al realizar los servicios, se sigue una estructura definida, por lo tanto cualquier componente del equipo del proyecto es capaz de ubicarse en servicios ajenos.
- La documentación del código ,como de los servicios mediante NaturalDocs, permite una facilidad mayor para la comprensión del código desarrollado.

ESTRUCTURAS DE CARPETAS

Otro de los aspectos más importantes dentro de un proyecto que componen varias personas. Al definir desde un inicio diferentes estructuras de carpetas para ubicar los archivos, genera que cualquier componente del equipo del proyecto, pueda acceder a cualquier ordenador en un momento dado, y saber rápidamente donde se encuentra la información que necesita.

Por otro lado, genera un ahorro de tiempo importante, al no tener que buscar los diferentes documentos, así como el código desarrollado.

8.3. CONCLUSIONES RELATIVAS AL PRODUCTO

IMPORTANCIA DE UN BUEN DISEÑO

A la hora de realizar una aplicación web, uno de los puntos más importantes es realizar un buen diseño de tanto las interfaces gráficas que la compongan, como las funcionalidades que tenga.

Si nos empleamos a fondo en esta fase, se verá recompensado el costo de tiempo notablemente a la hora de realizar el desarrollo, ya que nos podemos abstraer de tener que desarrollar, diseñar y corregir, generando una rueda que se repite continuamente.

8.4. CONCLUSIONES RELATIVAS AL TRABAJO EN EQUIPO

VISIÓN CONJUNTO DEL PROYECTO

Este aspecto es muy importante para mejorar altamente la eficiencia a la hora de realizar cualquier trabajo en equipo. Tener en todo momento una visión de lo que están realizando cada uno de los componentes del proyecto del equipo, permite que ante la necesidad de tener que abordar partes que en un principio no corresponden al trabajo asignado, puedan realizarse sin dificultades.

PREDISPOSICIÓN A AYUDAR

Cuando hablamos de proyectos en equipo, tener una predisposición a ayudar a los compañeros en todo momento ayuda mucho para resolver problemas puntuales. Es importante no reinventar la rueda, es decir, en caso de estar bloqueados en un determinado momento, pedir ayuda a un compañero puede repercutir en no tener que perder mucho tiempo en algo que el compañero pueda tardar menos de dos minutos en resolver.

8.5. CONCLUSIONES RELATIVAS AL PROYECTO

DOCUMENTACIÓN CONTINUA DEL TRABAJO REALIZADO

A la hora de tener que generar documentación sobre un proyecto, es de mucha utilidad ir generándola de manera progresiva y no dejarlo todo para el final. Esto mejora ampliamente la calidad de los documentos, así como la calidad de lo redactado.

9. BIBLIOGRAFÍA

Framework Ext Js. (s.f.). Recuperado el 8 de Octubre de 2013, de Wikipedia: http://es.wikipedia.org/wiki/Ext_JS

Hibernate. (s.f.). Recuperado el 8 de Octubre de 2013, de Wikipedia: http://es.wikipedia.org/wiki/Hibernate

Información Geograma. (s.f.). Recuperado el 8 de Octubre de 2013, de Geograma: http://www.geograma.com/

Notepad++. (s.f.). Recuperado el 11 de 10 de 2013, de Wikipedia: http://es.wikipedia.org/wiki/Notepad%2B%2B

OpenLayers. (s.f.). Recuperado el 8 de Octubre de 2013, de Wikipedia: http://es.wikipedia.org/wiki/OpenLayers

OpenStreetMap. (s.f.). Recuperado el 8 de Octubre de 2013, de Wikipedia: http://es.wikipedia.org/wiki/OpenStreetMap

PostgreSQL. (s.f.). Recuperado el 8 de Octubre de 2013, de Wikipedia: http://es.wikipedia.org/wiki/PostgreSQL

10. AGRADECIMIENTOS

Al finalizar un trabajo tan arduo como el desarrollo de un proyecto fin de carrera, al realizar un análisis objetivo del mismo, te lleva a la conclusión de que dicho proyecto no hubiera sido posible sin la participación de personas e instituciones que han facilitado las cosas para que este trabajo llegue a un feliz término.

Debo agradecer de manera especial y sincera al Profesor José Miguel Blanco por aceptarme para realizar este proyecto bajo su dirección. Su apoyo y confianza en mi trabajo y su capacidad para guiar mis ideas, no solo este año, sino también en años anteriores. Muchas gracias profesor.

Quiero agradecer extender un sincero agradecimiento a mis compañeros de trabajo, que me han estado dispuestos a ayudarme en todo momento. Me han brindado multitud de consejos y conocimientos que estoy seguro me ayudarán en el ámbito profesional así como personal.

Y, por supuesto el agradecimiento más profundo y sentido va para mi familia. Sin su apoyo , colaboración e inspiración habría sido imposible llevar a cabo esta dura empresa. Más cuando el único sustento económico ha venido de ellos. A mis padres Javier y Lourdes, por su apoyo constante, a mi hermano Erlantz y mi hermana Ania, por el constante ánimo transmitido, y como no, a mi recién nacida sobrina, que su sola presencia ya hace que el día a día sea mejor.!Por ellos y para ellos!

ANEXO I: MANUAL DE NATURAL DOCS

En el siguiente anexo, se detalla un pequeño manual de cómo utilizar la herramienta Natural Docs para generar documentación a la hora de tener que generar una API de los servicios realizados.

Para ello, mediante unos sencillos pasos, se podrá generar una documentación muy clara y concisa que servirá de ayuda para el manejo de los servicios realizados.

En el caso de este proyecto, todos los servicios realizados en cada una de las tres capas, SOAP, Servlet y Javascript, han sido documentados y se ha generado una API correspondiente a cada uno de dichas capas.

1. DOCUMENTACIÓN EN EL CÓDIGO

Comenzaremos detallando como documentar el código. Para ello, mostraremos como se ha realizado en cada una de las capas. Tanto en la capa SOAP, como en la capa Servlet, los métodos se documentarán en las interfaces que componen los servicios de dichas capas.

Comenzaremos mostrando la documentación realizada en la capa SOAP, para ello cogeremos como ejemplo el servicio SOAP UMS. Si analizamos el archivo IUms.java podremos ver lo siguiente:

```
public interface IUms {

    /*
      Function: getGroups

       Returns the group information associated with a particular account of the
       Platform GeoServices. Optionally, you can specify a series of filters on the
       groups to obtain, or if you want to get only a summary of the groups or any
       associated information.

      Parameters:

        username      - {String}  A valid user of the Plattform.
        token         - {String}  Associated with the user token to validate obtained
                                  by the method getToken.
        filter        - {String}  Filter on groups to obtain.
        extendedinfo  - {boolean} Indicates to get also the user information
                                  associated with each group

      Returns:

        A <GetGroupsResponse> structure

    */
    public GetGroupsResponse getGroups(String username, String token, String filter,
boolean extendedinfo);
```

Lo primero que tendremos que hacer es introducir comentarios siempre antes de los métodos. Los parámetros que deberemos rellenar son los siguientes:

- Function: Nombre función : En este apartado tendremos que introducir la descripción de la funcionalidad del método del servicio.
- Parameters: Aquí tendremos que especificar cada uno de los parámetros de entrada.
- Returns: Aquí tendremos que especificar lo que retorna la función.

Este procedimiento se tendrá que repetir en cada uno de los métodos que tenga el servicio realizado.

Seguiremos mostrando la documentación realizada en la capa Servlet. En este caso, el procedimiento será el mismo. Si analizamos el archivo IUms.java de la capa Servlet podremos ver lo siguiente:

```
public interface IUms {
    /*
      Function: request=getGroups
       Return the groups associated to a requested domain.Optionally, you can
       specify a series of filters, or request the info of the   users of each group

      Sample usage:
        (code)
http://ums servlet url/services/Ums?request=getGroups&username=user&key=theKey&filter
=theFilter&extendedInfo=isExtendedInfo&start=start&limit=limit&callback=callback
        (end)

       Parameters:

          request       - {String} Name of the requested service (getGroups)
          username      - {String} The email of a registered user for the product.
          key           - {String} A valid key for the requested service
          filter        - {String} An optional filter for the requested groups
          extendedInfo  - {Boolean} Indicates if in the request must appear extended
                              info with the associated users for each group
          start         - {int} Position of the first returned record
          limit         - {int} Position of the last returned record
          callback      - {String} Name of the function that process the response

       Returns {JSON}:

          (code)
          {
            "data": {
                "error": 0,
                "totalProperty": 2,
                "groups": [{
                    "users": [{
                        "email": "user1@geograma.com",
                        "name": "Name1",
                        "surname": "Surname1",
                        "role": "user"
                    }],
                    "description": "Default group",
                    "name": "default",
                    "numUsers": 1
                }
                }],
                "metadata": {
                    "Request time": "120 ms"
                }
            },
            "success": true
          }
          (end)
     */
    void getGroups(HttpServletRequest request, HttpServletResponse response, String
soapUrl, String username) throws JSONException, IOException;
```

Lo primero que tendremos que hacer es introducir comentarios siempre antes de los métodos. Los parámetros que deberemos rellenar son los siguientes:

- Function: Nombre función : En este apartado tendremos que introducir la descripción de la funcionalidad del método del servicio.

- Parameters: Aquí tendremos que especificar cada uno de los parámetros de entrada.
- Returns: Aquí tendremos que especificar lo que retorna la función.

Este procedimiento se tendrá que repetir en cada uno de los métodos que tenga el servicio realizado.

Por último mostraremos la documentación realizada en la capa Javascript. Este paso es análogo a lo realizado con la capa Servlet, ya que la documentación del código es idéntica, a diferencia de si cambia algún parámetro.

Una vez documentado el código, procederemos a mostrar como generar la documentación.

2.GENERACIÓN DE LA API

Para poder generar la API, lo primero que haremos será descargarnos desde la página de Natural Docs el software necesario. Para ello tendremos que realizar dos cosas:

- Instalación de active Perl
- Descarga de la última versión de Natural Docs

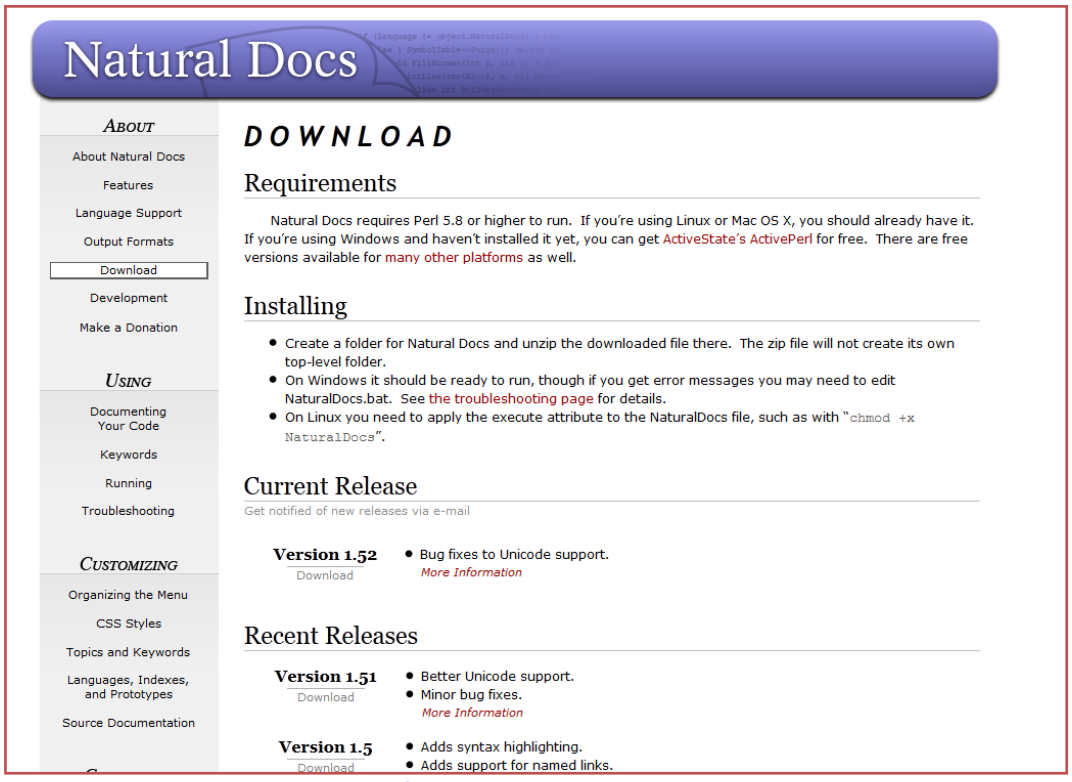


Ilustración 22 Web descarga Natural Docs

Una vez instalado Active Perl, descomprimimos la versión de Natural Docs en la localización donde tengamos ubicado el servicio a documentar. Tendremos que crear otra carpeta que se llame output en la misma localización.

Una vez realizado esto, tendremos que configurar Natural Docs con las rutas de los archivos que queremos documentar. Para ello accederemos a la carpeta **naturaldocs_project** y editaremos el script **ums.bat**.

```
NaturalDocs
    -i C:\ums\soap\service\src\service
    -o HTML C:\ums\soap\naturaldocs\output
    -p C:\ums\soap\naturaldocs\naturaldocs_project\project
```

Como podemos ver, tendremos que rellenar tres parámetros:

- -i : En esta línea tendremos que especificar donde se encuentra el archivo con la documentación.
- -o : En esta línea tendremos que especificar la ruta donde queremos que nos genere la API.
- -p : En esta línea tendremos que especificar la ruta donde hemos descomprimido la versión descargada de natural docs.

Una vez editado el script, lo ejecutaremos y en la carpeta output nos generará la documentación pertinente en forma de archivo HTML.

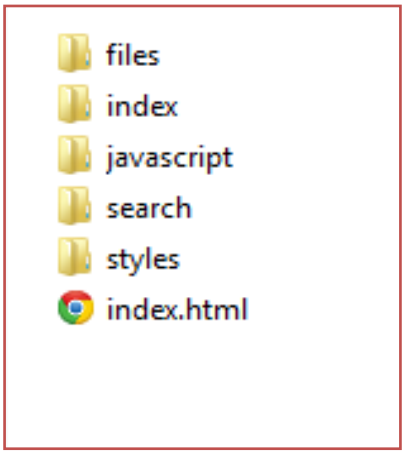
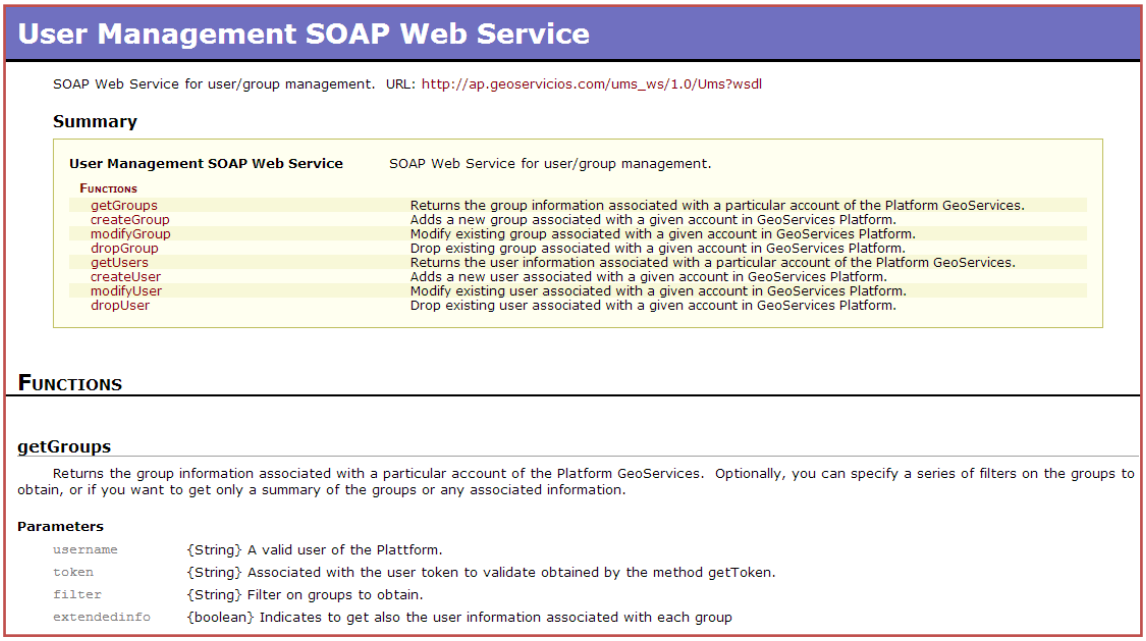


Ilustración 23 Estructura carpetas Natural Docs

3. RESULTADO FINAL

Una vez realizados todos los pasos, si accedemos a la carpeta output y ejecutamos el index.html, podremos ver la API generada. Esto lo vemos en la siguiente figura:



User Management SOAP Web Service

SOAP Web Service for user/group management. URL: http://ap.geoservicios.com/ums_ws/1.0/Ums?wsdl

Summary

User Management SOAP Web Service	SOAP Web Service for user/group management.
Functions	
getGroups	Returns the group information associated with a particular account of the Platform GeoServices.
createGroup	Adds a new group associated with a given account in GeoServices Platform.
modifyGroup	Modify existing group associated with a given account in GeoServices Platform.
dropGroup	Drop existing group associated with a given account in GeoServices Platform.
getUsers	Returns the user information associated with a particular account of the Platform GeoServices.
createUser	Adds a new user associated with a given account in GeoServices Platform.
modifyUser	Modify existing user associated with a given account in GeoServices Platform.
dropUser	Drop existing user associated with a given account in GeoServices Platform.

Functions

getGroups

Returns the group information associated with a particular account of the Platform GeoServices. Optionally, you can specify a series of filters on the groups to obtain, or if you want to get only a summary of the groups or any associated information.

Parameters

username	{String} A valid user of the Plattform.
token	{String} Associated with the user token to validate obtained by the method getToken.
filter	{String} Filter on groups to obtain.
extendedinfo	{boolean} Indicates to get also the user information associated with each group

Ilustración 24 Api Natural Docs generada

ANEXO II: NORMATIVA DE DESARROLLO SERVICIOS Y HIBERNATE

El objetivo del siguiente capítulo es especificar los pasos a seguir para desarrollar un nuevo servicio. Se detalla tanto la configuración del entorno de trabajo como las diferentes etapas para crear un servicio teniendo en cuenta la configuración de Hibernate.

1. CONFIGURACIÓN DEL ENTORNO DE TRABAJO

Los distintos proyectos que forman los servicios geográficos deben localizarse en nuestro PC en la siguiente ruta:

C:\softwarebase\services\applications\administrationpanel

El workspace de Eclipse asociado a este conjunto de servicios debe localizarse en nuestro PC en la siguiente ruta:

C:\workspaces\softwarebase\services\applications\administrationpanel

2. CREACIÓN DE UN NUEVO SERVICIO SOAP

Lo primero que haremos será crear la estructura de carpetas del servicio. En este caso vamos a crear un servicio llamado **lms_soap**. A continuación podemos ver la estructura de carpetas creada:

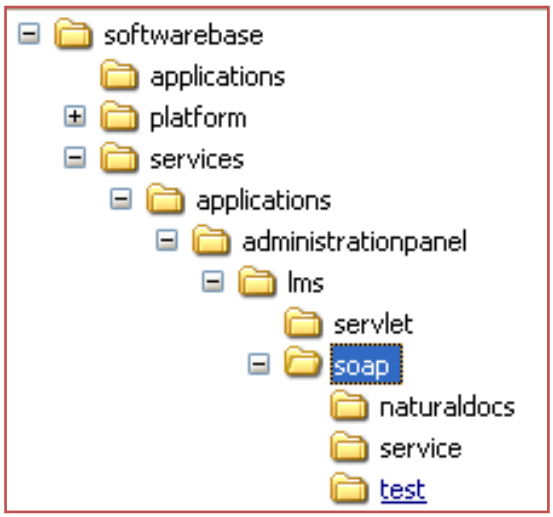


Ilustración 25 Estructura de carpetas servicio SOAP

Dentro del servicio soap, se han creado 3 carpetas:

- **NaturalDocs:** Esta carpeta la utilizaremos para realizar la documentación del servicio.
- **Service:** En esta carpeta se implementará el servicio en cuestión.
- **Test:** En esta carpeta se crearán los test necesarios para probar el servicio.

A continuación iniciaremos Eclipse. Lo primero será cambiar el workspace para que sea el siguiente:

C:\workspaces\softwarebase\services\applications\administrationpanel

Para ello pulsaremos en **File\Switch Workspace\Other** e introduciremos la ruta indicada anteriormente. Esto lo podemos ver en la siguiente figura:

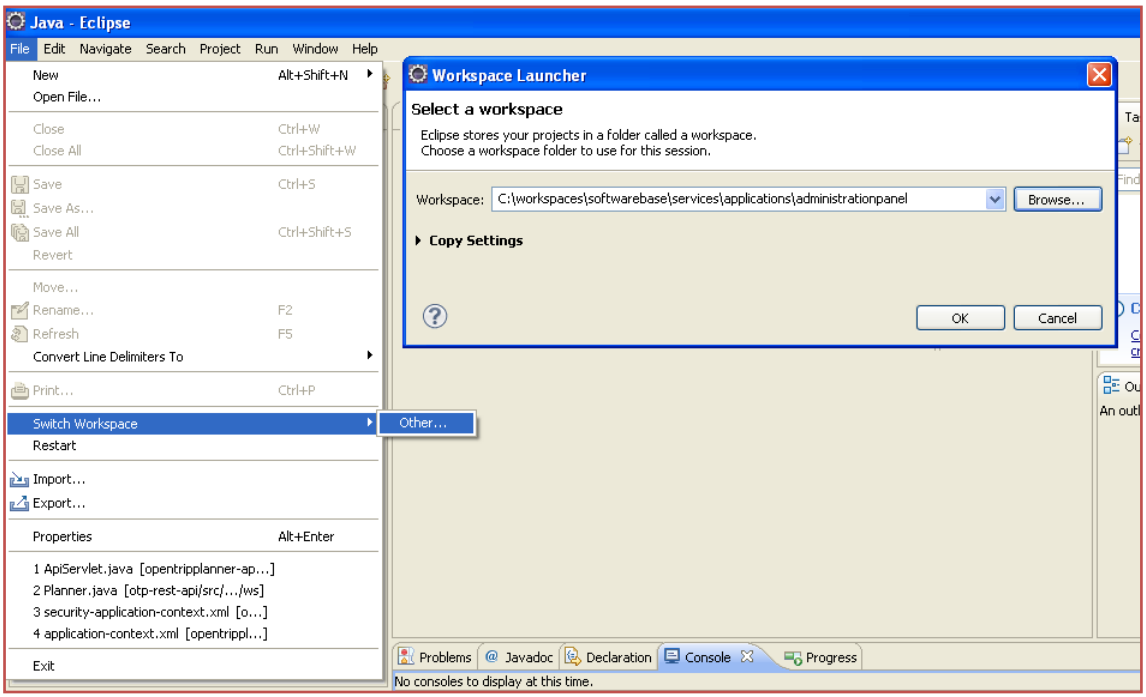


Ilustración 26 Workspace servicio SOAP

Una vez realizado esto procederemos a crear un nuevo proyecto. Para ello tendremos que pulsar **New\Dynamic Web Project**. Nos aparecerá una ventana con varias opciones que serán las siguientes:

- **Project Name:** lms_soap (Establecemos el nombre del servicio)
- **Location:** En este caso tenemos que especificar la carpeta que previamente hemos creado donde se almacenará el servicio. En este caso:

  C:\Softwarebase\services\applications\administrationpanel\lms\soap\service

- **Dynamic web module version:** 2.5 (Establecemos la versión a utilizar en 2.5)

Una vez introducidos los siguientes parámetros, pulsaremos en continuar. Esto último lo podemos ver en la siguiente figura:

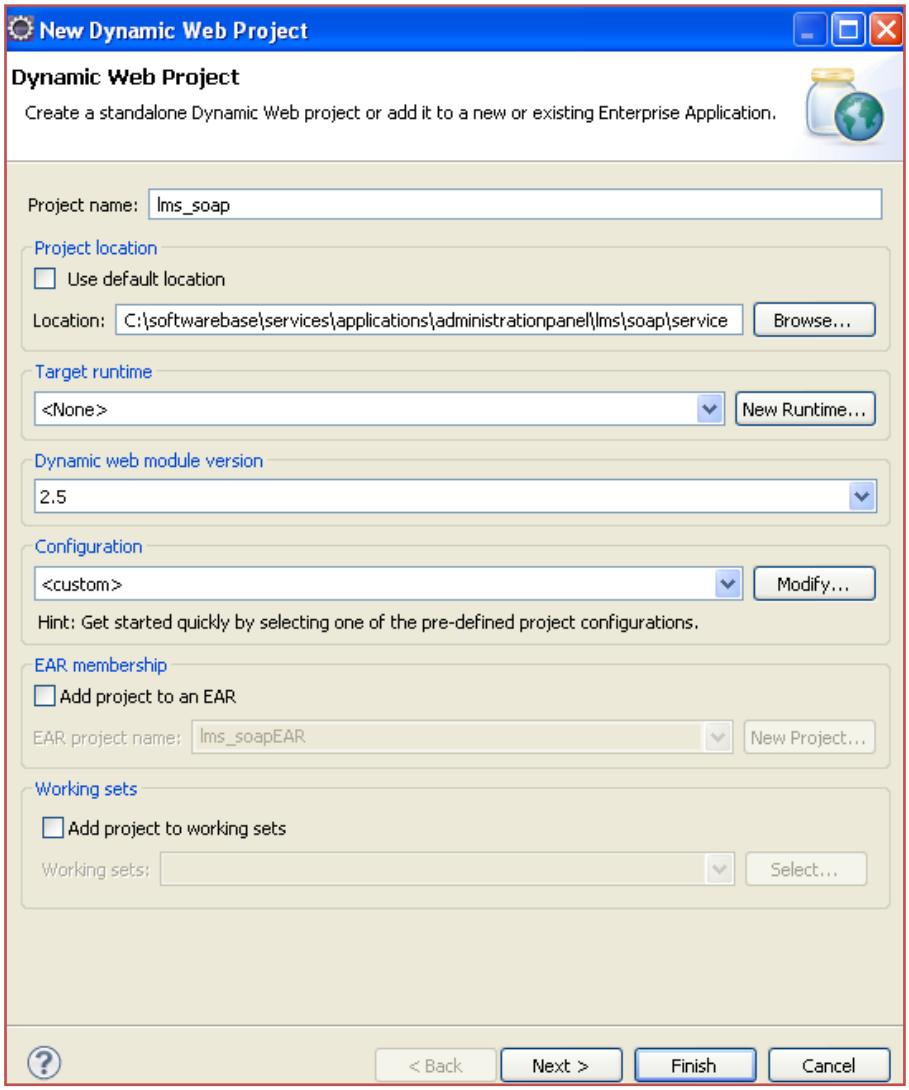


Ilustración 27 Creación proyecto servicio SOAP

Tras esto, tenemos que introducir las librerías que vamos a utilizar en nuestro proyecto. Para ello, accederemos a cualquier servicio que tengamos ya previamente creado, y accediendo a la siguiente ruta copiaremos las librerías ya existentes, y las pegaremos en l ruta del servicio a crear. Si no es así tendremos que agregar las librerías externas como la de login, Hibernate si lo utilizamos, etc..

Seguidamente procederemos a configurar el compilador de Eclipse y estableceremos la versión 1.6. Esto lo haremos accediendo de la siguiente manera: **Propiedades (de nuestro proyecto) \Java Compiler** y establecemos el 1.6. Esto lo podemos observar en la siguiente figura:

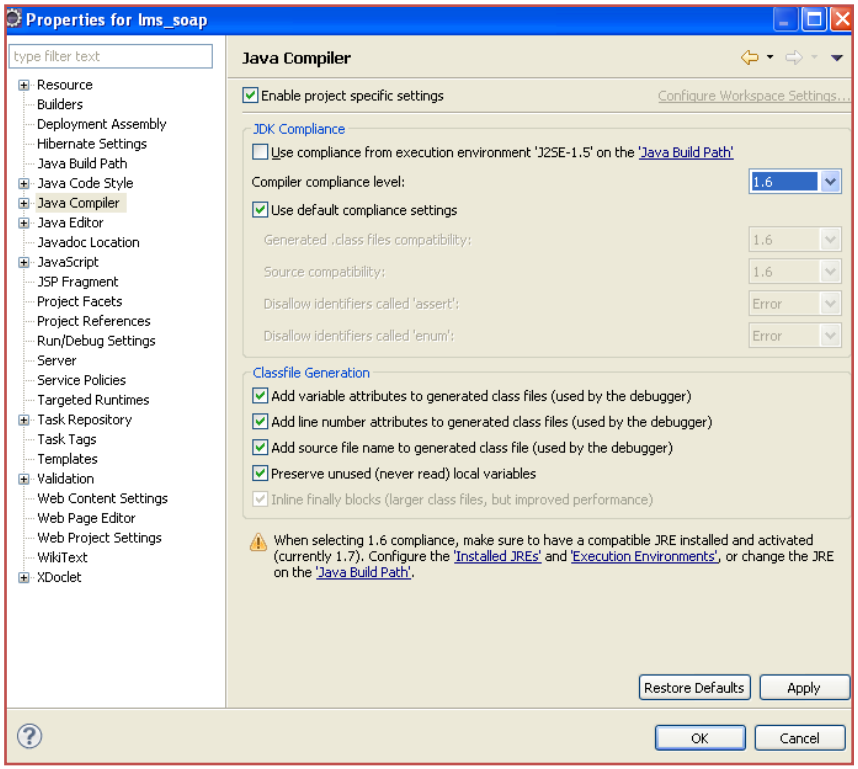


Ilustración 28 Configuración entorno desarrollo servicio SOAP

Aprovecharemos también para añadir el servidor apache tomcat al proyecto. Para ello pulsaremos en la pestaña **Servers\New server wizard.** Una vez realizado esto, seleccionaremos la versión de tomcat que vayamos a utilizar:

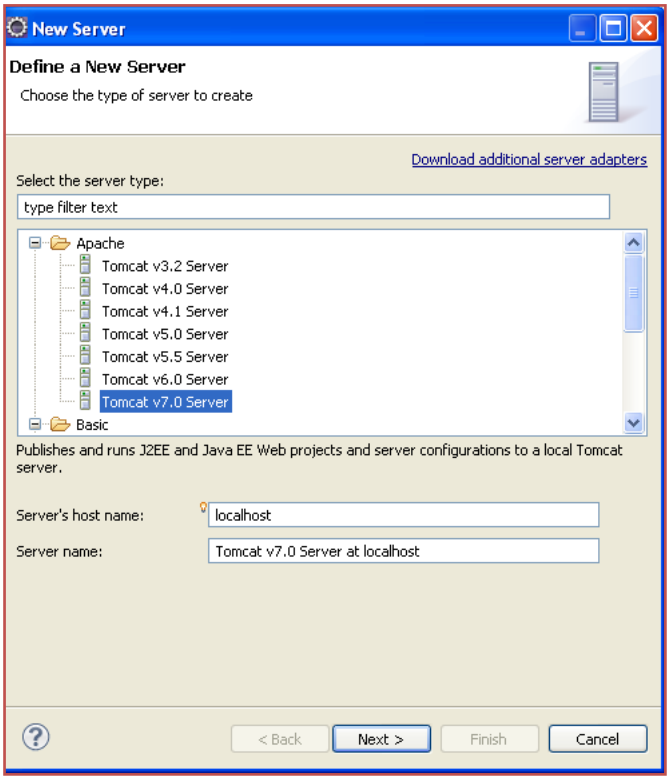


Ilustración 29 Servidor servicio SOAP

Y seguidamente, tendremos que insertar la ruta donde tengamos instalado el tomcat.

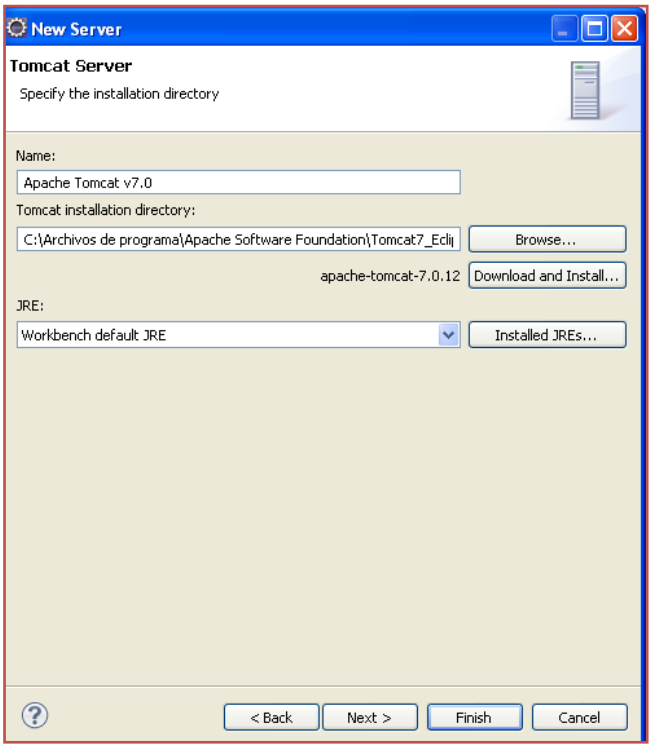


Ilustración 30 Servidor servicio SOAP 2

Tras esto, tendremos que crear una carpeta deploy en nuestro proyecto. Para ello accederemos a la carpeta service y crearemos dicha carpeta. En esta carpeta se crearán los archivos .War cuando queramos preparar el servicio para lanzarlo. La estructura que tenemos que tener es la siguiente:

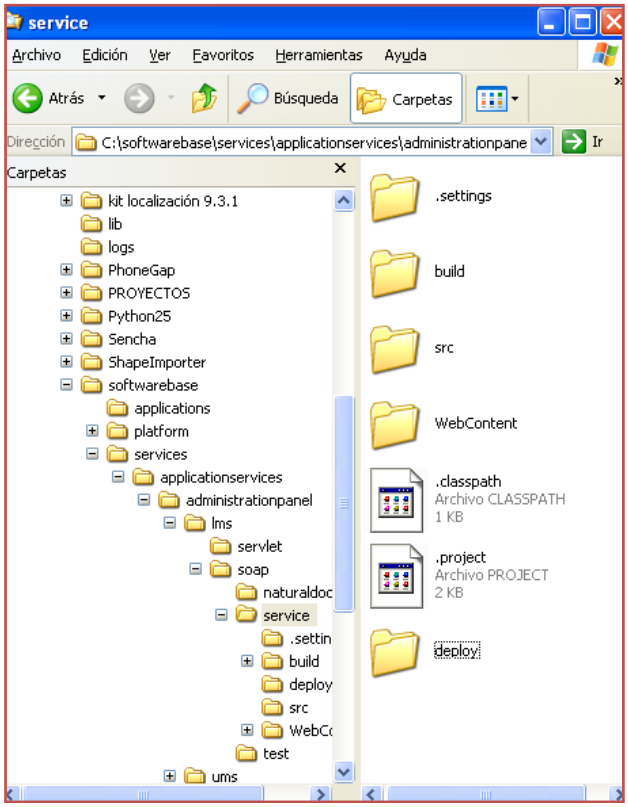


Ilustración 31 Estructura carpetas servicio SOAP 2

Seguidamente, dentro de **src** crearemos la siguiente estructura de carpetas:

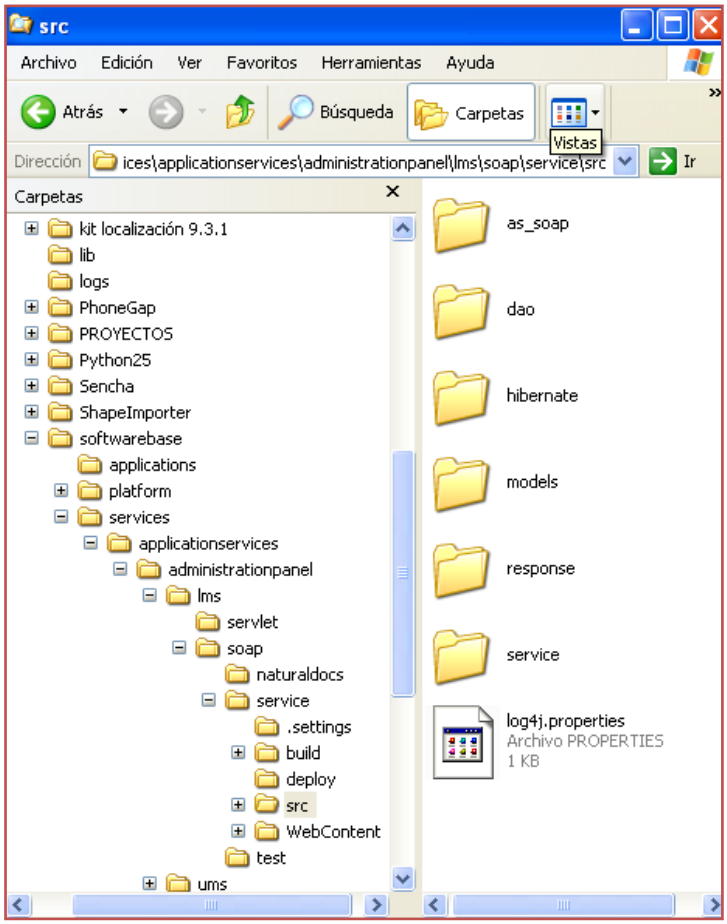


Ilustración 32 Estructura carpetas servicio SOAP 3

Analicemos cada una de las carpetas introducidas:

- **As_soap:** Este paquete se encargará de llamar al servicio de autenticación ya existente.
- **Dao:** Este paquete se encargará de realizar la conexión y consultas a la base de datos.
- **Hibernate:** En este paquete se encontrarán los archivos necesarios para la configuración y utilización de Hibernate.
- **Models:** En este paquete se mapearán las tablas que seleccionemos a la hora de trabajar con Hibernate.
- **Response:** Este paquete contiene los parámetros a devolver por el servicio que se va a crear.
- **Service:** En este paquete tendremos la lógica del servicio.

Por último el archivo **log4j.properties** contendrá la información necesaria para utilizar el log. Este archivo contendrá lo siguiente:

```
log4j.appender.file=org.apache.log4j.RollingFileAppender
log4j.appender.file.File=${catalina.base}/logs/lms_soap.log
log4j.appender.file.append=true
log4j.appender.file.DatePattern='.'yyyy-MM-dd
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d (%5p) %-5l - <%m>%n
log4j.rootLogger=error, file


#httpclient logging properties
log4j.logger.httpclient.wire=ERROR

#Hibernate logging proeprties
log4j.logger.org.hibernate=ERROR
log4j.logger.org.hibernate.sql=ERROR
log4j.logger.org.hibernate.hql=ERROR
log4j.category.org.hibernate.dialect=ERROR

#Axis2 logging properties
log4j.logger.org.apache.axis2.enterprise=ERROR
log4j.logger.org.apache.axis2=ERROR
log4j.logger.org.apache.axiom=ERROR
log4j.logger.de.hunsicker.jalopy.io=ERROR
log4j.logger.org.apache.commons.httpclient=ERROR
log4j.logger.com.mchange=ERROR
```

Mediante este archivo de configuración, podemos controlar el Log. Hay que tener en cuenta que en caso de insertar alguna librería externa nueva, habría que comprobar si utilizan el Log y añadirlo en este archivo al igual que con Httpclient, Hibernate y Axis2.

Es importante que cambiemos el nombre de la siguiente línea estableciendo el nombre de nuestro servicio:

log4j.appender.file.File=${catalina.base}/logs/lms_soap.log

Comenzaremos rellenando el paquete services. En este paquete crearemos una interfaz y una clase.

Interfaz:

```
package service;


import los paquetes necesarios;

public interface IDms {

    /*
      Comentarios para  generar api con Natural Docs
    */
    public GetStoreResponse getStore(String username, String token, String filter);

}
```

Clase:

```
package service;


import los paquetes necesarios;

public class Dms {

    private Logger logger = Logger.getLogger(Dms.class);

    private Properties prop = new Properties();

    public GetStoreResponse getStore(String username, String token, String filter){
        String objId = "getStore";
        long startMillis = System.currentTimeMillis();
        long endMillis = 0;
        logger.info("Inicio método: "+objId+"-"+username+"-"+token+"-"+filter);
        GetStoreResponse response = new GetStoreResponse();
        try{
            //Load service properties
            prop.load(this.getClass().getResourceAsStream("soap_service.properties"));
            //Check method parameters
            IDao dao = new Dao();
            response = dao.getStore(filter);
            endMillis = System.currentTimeMillis();
            return response;

        }catch(Exception oo){
            return response;
        }
    }
}
```

Seguiremos completando el paquete response. Para ello crearemos una clase con cada uno de los métodos que vayamos a crear en el servicio. La estructura será la siguiente.

```
package response;


import java.io.Serializable;

public class GetStoreResponse implements Serializable{
    /**
     * Indicates if the method has been executed without exceptions
     */
    private boolean success = false;
    /**
     * Indicates the error code when something goes bad. If everthing is correct
     * the value is 0
     */
    private int error = 0;
    /**
     * Metadata of the response
     */
    private Metadata[] metadata = null;
    /**
     * Number of elements
     */
    private int totalProperties = 0;
    /**
     * Todos los getter y setter que necesitemos en la respuesta
     */
}
```

De momento, en Eclipse tendríamos la siguiente estructura, con los nombres de los métodos que hayamos creado (la estructura es orientativa).

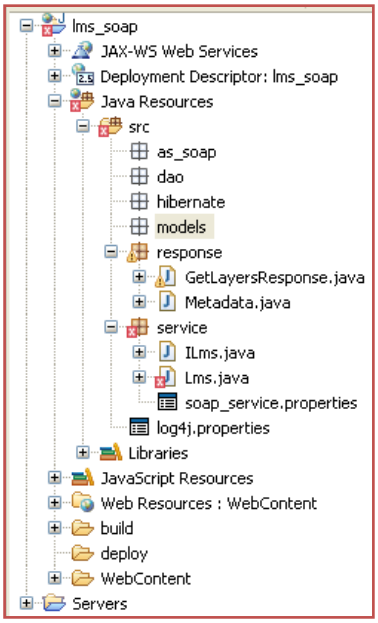


Ilustración 33 Estructura carpetas eclipse servicio SOAP

Seguidamente, realizaremos dos configuraciones en eclipse para que podamos manejar Apache Axis2. Para ello tendremos que realizar los siguientes dos pasos:

Accederemos a **Window\Preferences\Web services\Axis2Preferences**. Aquí tendremos que introducir la localización de Apache Axis2. Utilizaremos la versión 1.6.2.

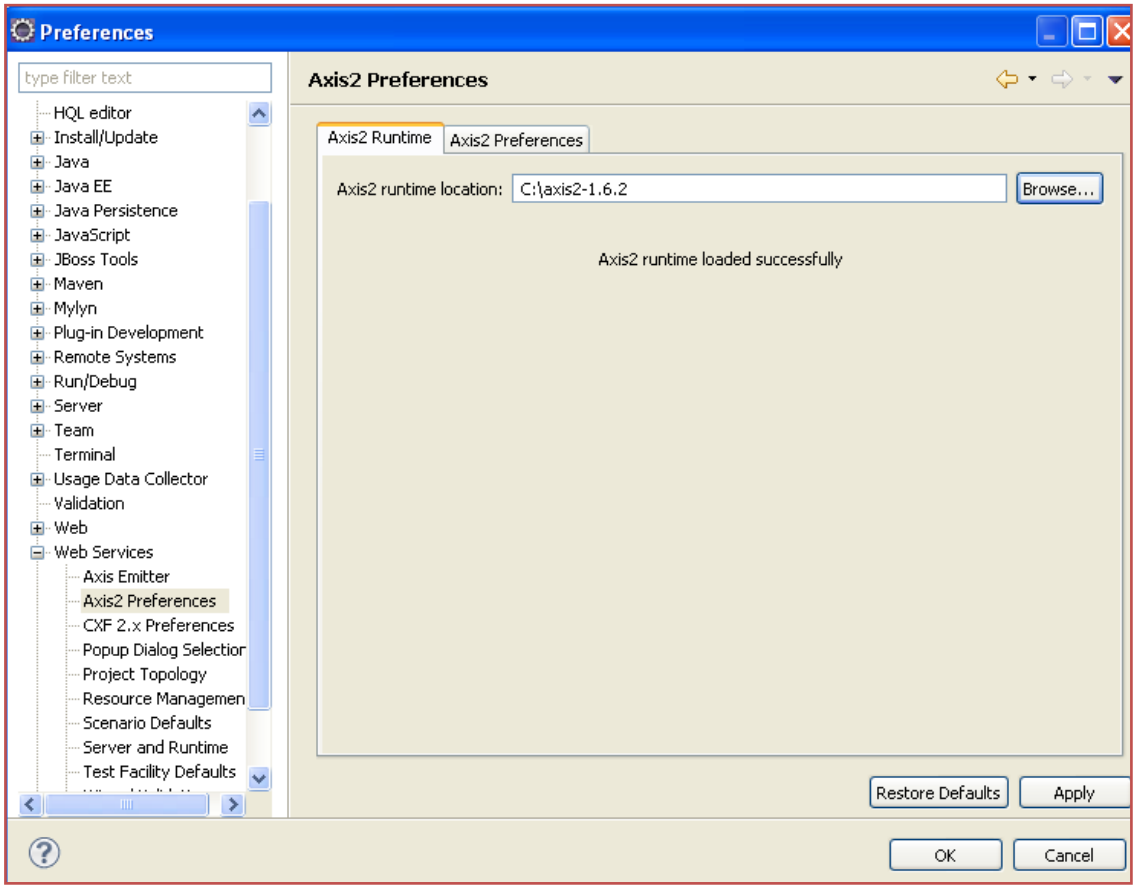


Ilustración 34 Configuración Axis2 servicio SOAP

También tendremos que añadir las librerías de tomcat. Para ello accederemos en nuestro proyecto a **Propiedades\Java Build Path\Libraries** y pulsaremos en añadir librería. Seleccionaremos **Server Runtime**.

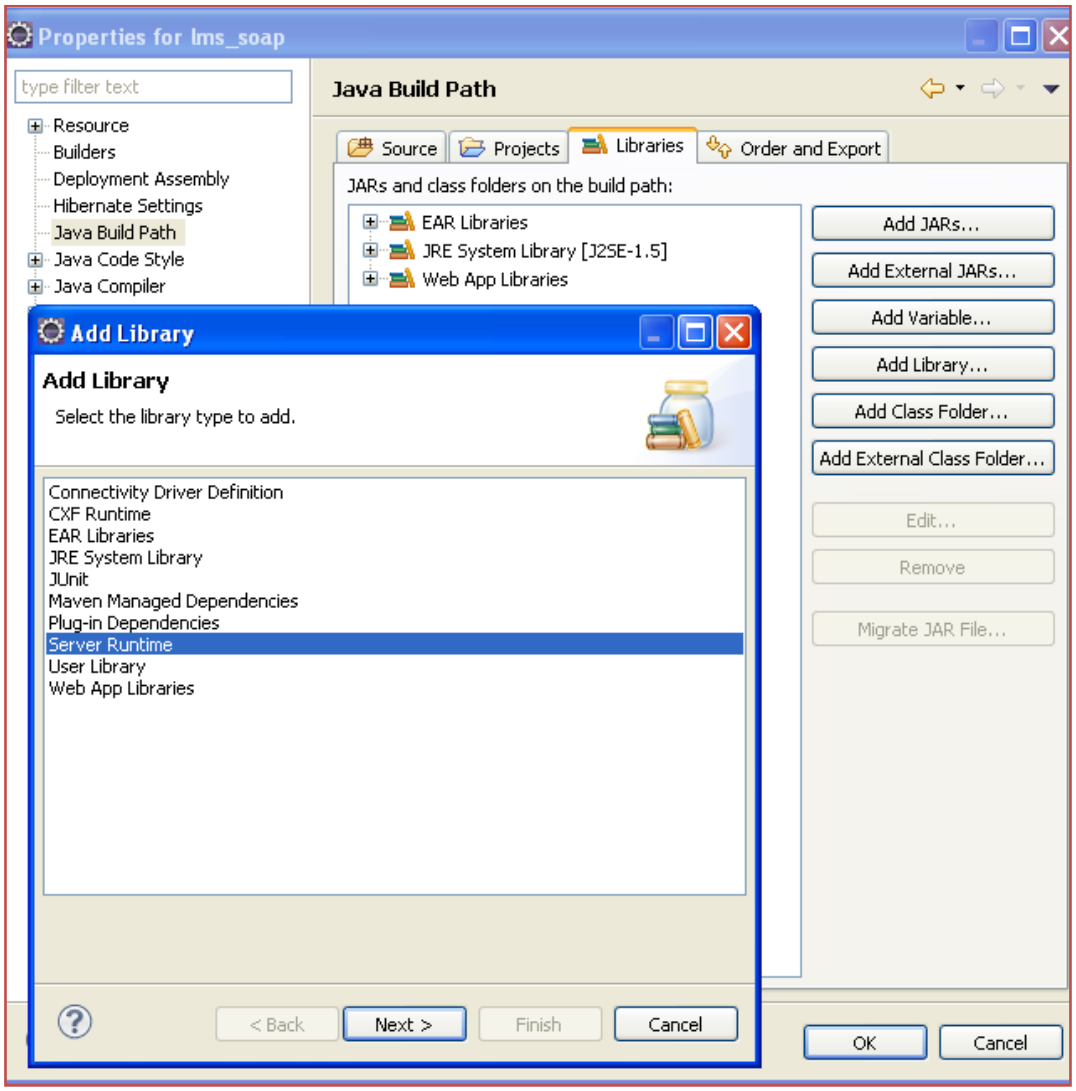


Ilustración 35 Configuración Tomcat servicio SOAP

A continuación procederemos a implementar el servicio web de autenticación. Para ello pulsaremos encima del paquete as_soap y pulsaremos en **New\Web Service Client**. En la siguiente ventana, tendremos que introducir la url donde está alojado el servicio web, en este caso:

http://100.100.100.165:8080/as_soap_2_0/services/As?wsdl

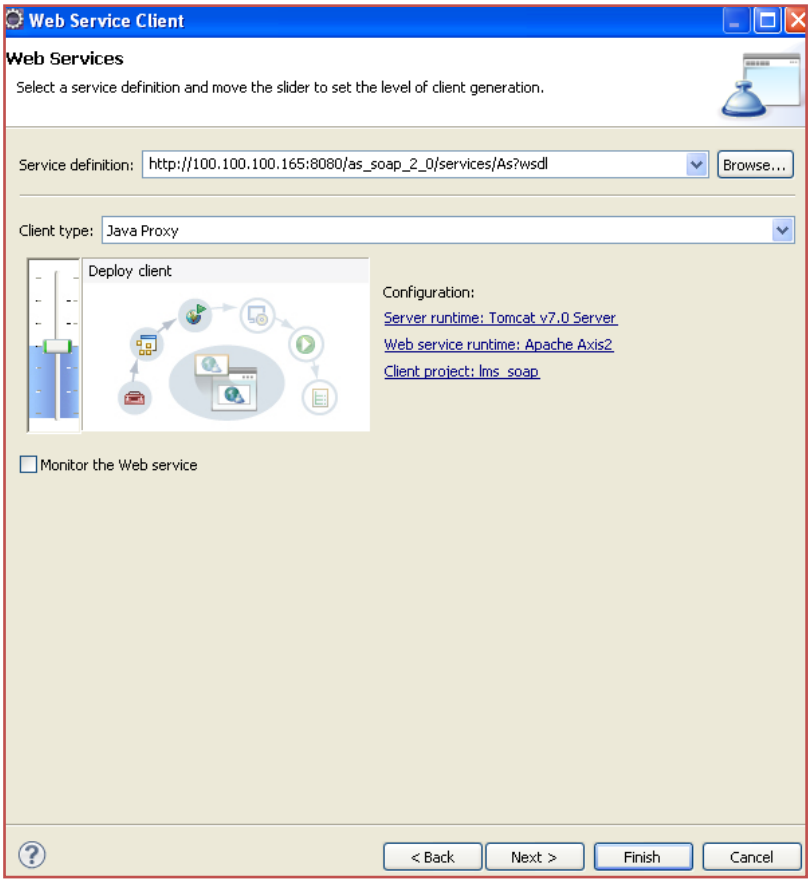


Ilustración 36 Creación web service client servicio SOAP

Seguidamente, pulsaremos en **Web service runtime** y seleccionaremos Apache Axis2

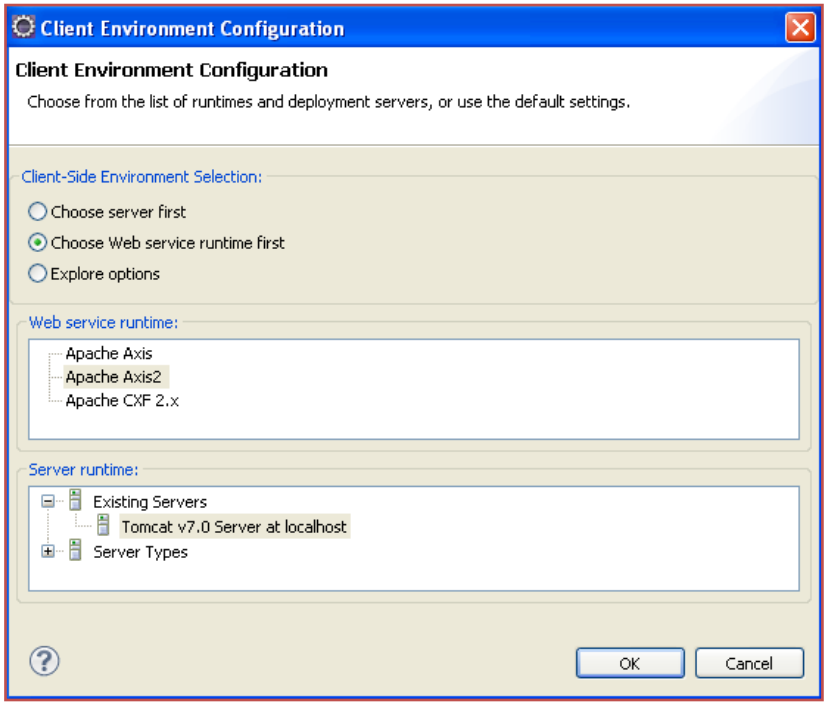


Ilustración 37 Creación web service client servicio SOAP 2

En la siguiente ventana podremos ver la información del servicio que estamos importando. Es muy importante que indiquemos el paquete as_soap como Custom package name:

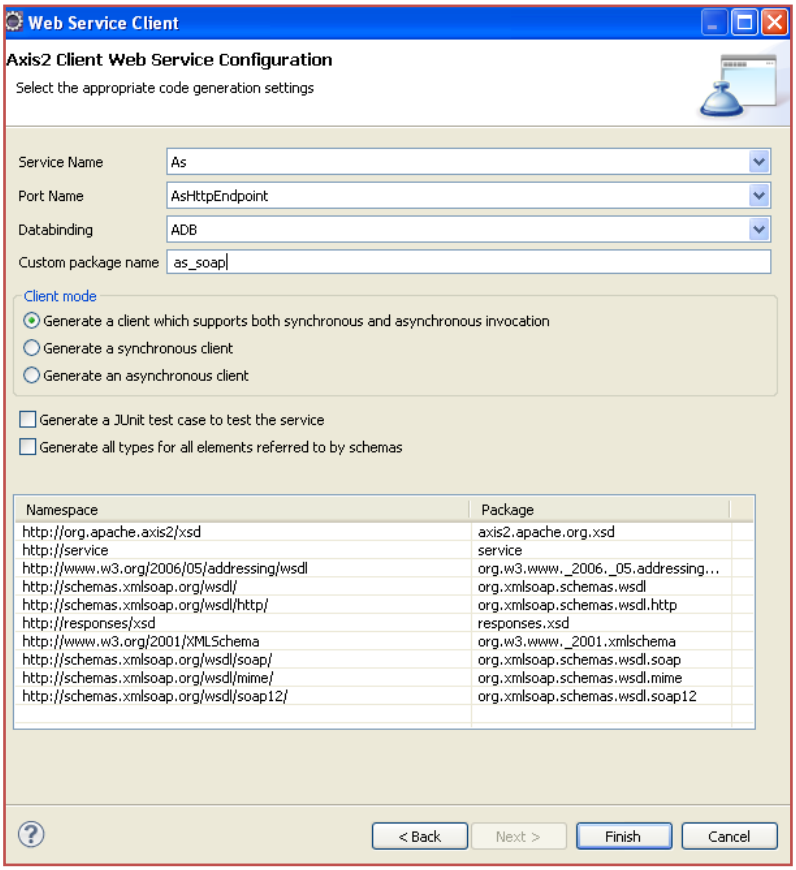


Ilustración 38 Creación web service client servicio SOAP 3

A continuación, procederemos a configurar la parte de Hibernate. Para ello lo primero que tendremos que hacer es descargarnos el pluging para eclipse. Esto lo podemos hacer en **Help\Install New Software** e insertando la siguiente url:

http://download.jboss.org/jbosstools/updates/stable/(versión de eclipse)/
*version de eclipse: juno, helios…

Una vez tengamos instalado este pluging, tendremos que abrir la perspectiva de Hibernate. Para ello pulsaremos en **Window\Open perspective** y seleccionaremos Hibernate.

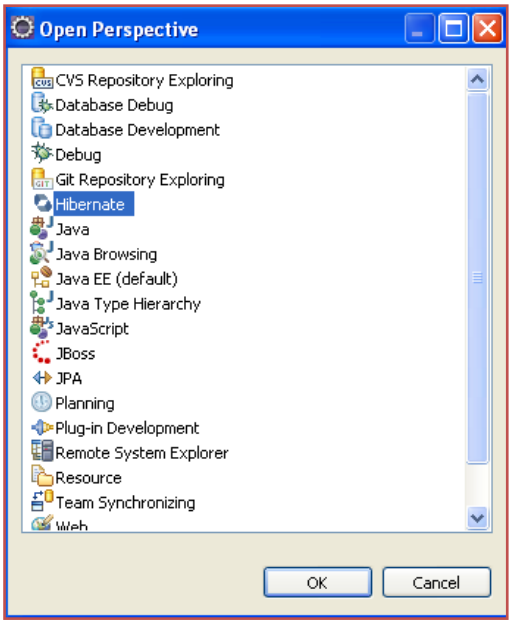


Ilustración 39 Perspectiva Hibernate servicio SOAP

Tras esto, procederemos a crear un archivo de configuración de Hibernate. Para ello pulsaremos en el paquete Hibernate de nuestro proyecto y entramos en **New\Hibernate Configuration File**.

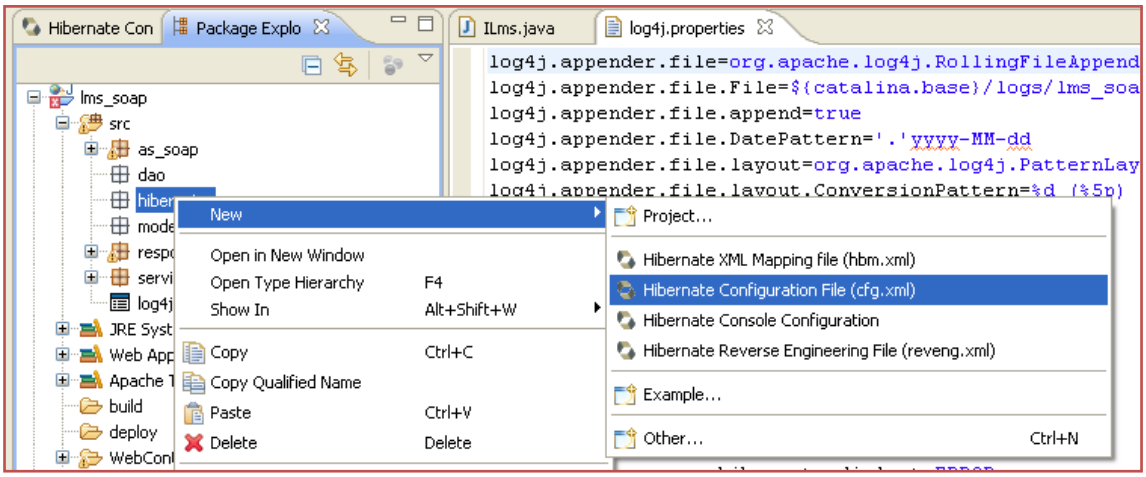


Ilustración 40 Archivo de configuración Hibernate servicio SOAP

Estableceremos el nombre del archivo siguiendo la nomenclatura Hibernate (tipo de base de datos).cfg.xml (Por ejemplo: hibernatePostgres.cfg.xml)

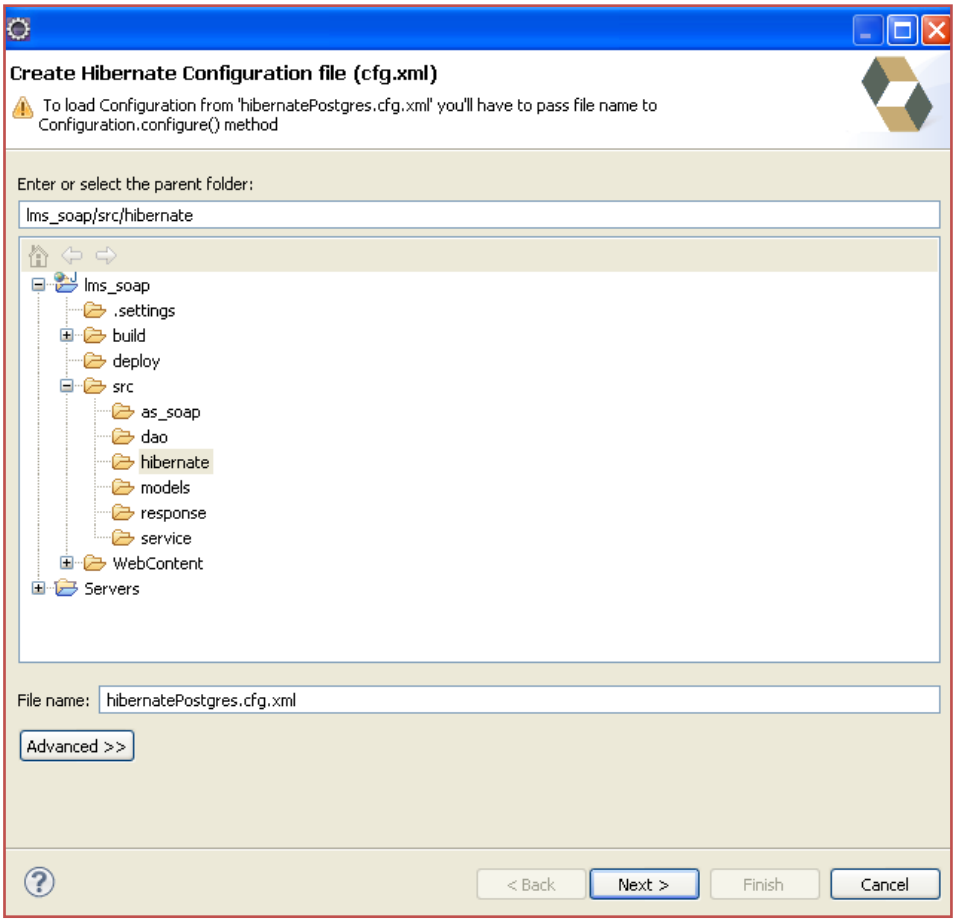


Ilustración 41 Archivo de configuración Hibernate servicio SOAP 2

Seguidamente tendremos que rellenar los datos del archivo de configuración. Para ello accederemos a otro archivo de configuración de otro servicio e iremos rellenando nuestro archivo con dicha información:

```
<hibernate-configuration>
 <session-factory>
  <property name="hibernate.connection.driver_class">org.postgresql.Driver</property>
  <property name="hibernate.connection.password">A-guinea</property>
  <property name="hibernate.connection.url">jdbc:postgresql://100.100.100.105:5433/geoserviciosv2</property>
  <property name="hibernate.connection.username">postgres</property>
  <property name="hibernate.default_catalog">geoserviciosv2</property>
  <property name="hibernate.default_schema">public</property>
  <property name="hibernate.dialect">org.hibernate.dialect.PostgreSQLDialect</property>
  <property name="current_session_context_class">thread</property>
  <property name="hibernate.show_sql">false</property>
  <property name="hibernate.format_sql">false</property>
  <property name="use_sql_comments">false</property>
  <property name="hibernate.c3p0.min_size">5</property>
  <property name="hibernate.c3p0.max_size">10</property>
  <property name="hibernate.c3p0.timeout">1000</property>
  <property name="hibernate.c3p0.max_statements">50</property>
  <property name="hibernate.c3p0.idle_test_period">10000</property>
 </session-factory>
</hibernate-configuration>
```

En eclipse:

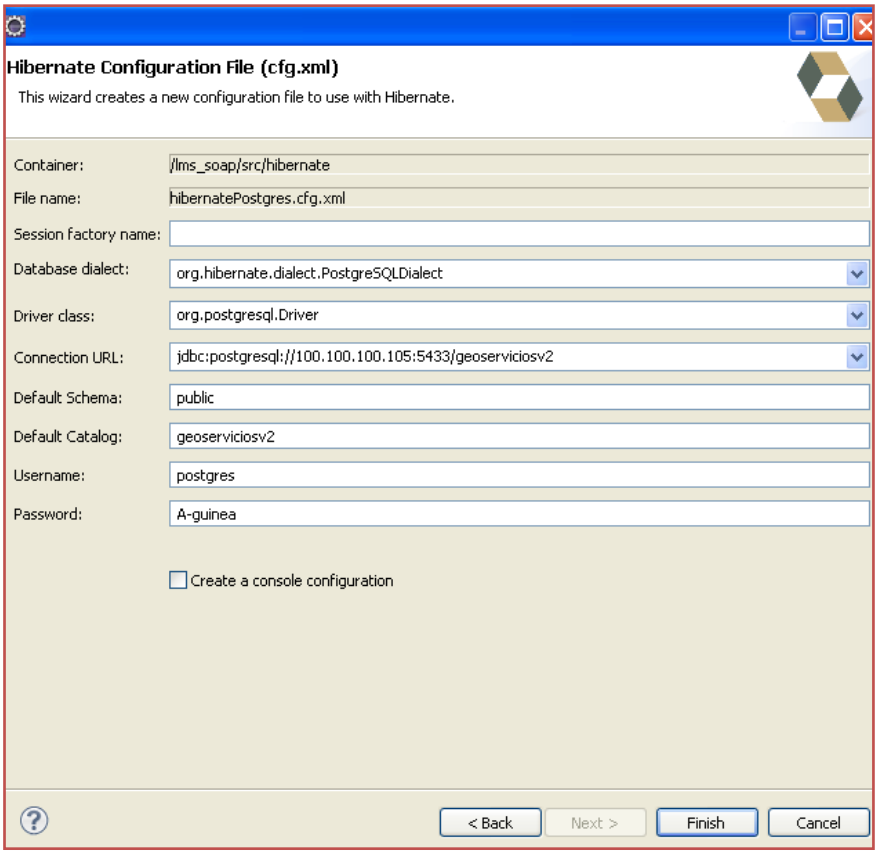


Ilustración 42 Archivo de configuración Hibernate servicio SOAP 3

A continuación lo que haremos será abrir el archivo hibernatePostgres.cfg.xml de nuestro servicio y copiar los campos que nos faltan. Añadiremos los campos necesarios para gestionar un pull de conexiones. Los campos que tendremos que añadir son los siguientes:

```
<property name="current_session_context_class">thread</property>
<property name="hibernate.show_sql">false</property>
<property name="hibernate.format_sql">false</property>
<property name="use_sql_comments">false</property>
<property name="hibernate.c3p0.min_size">5</property>
<property name="hibernate.c3p0.max_size">10</property>
<property name="hibernate.c3p0.timeout">1000</property>
<property name="hibernate.c3p0.max_statements">50</property>
<property name="hibernate.c3p0.idle_test_period">10000</property>
```

En este archive de configuración, nos queda insertar un campo. El campo a insertar es la propiedad mapping. Esta propiedad depende de moddels, por lo tanto, primero tendremos que configurar dicho paquete antes de añadirlo.

Seguidamente, pasaremos a la perspectiva de Hibernate. Con el botón derecho seleccionaremos **Add Configuration**. Rellenaremos la información que nos pide de la siguiente manera:

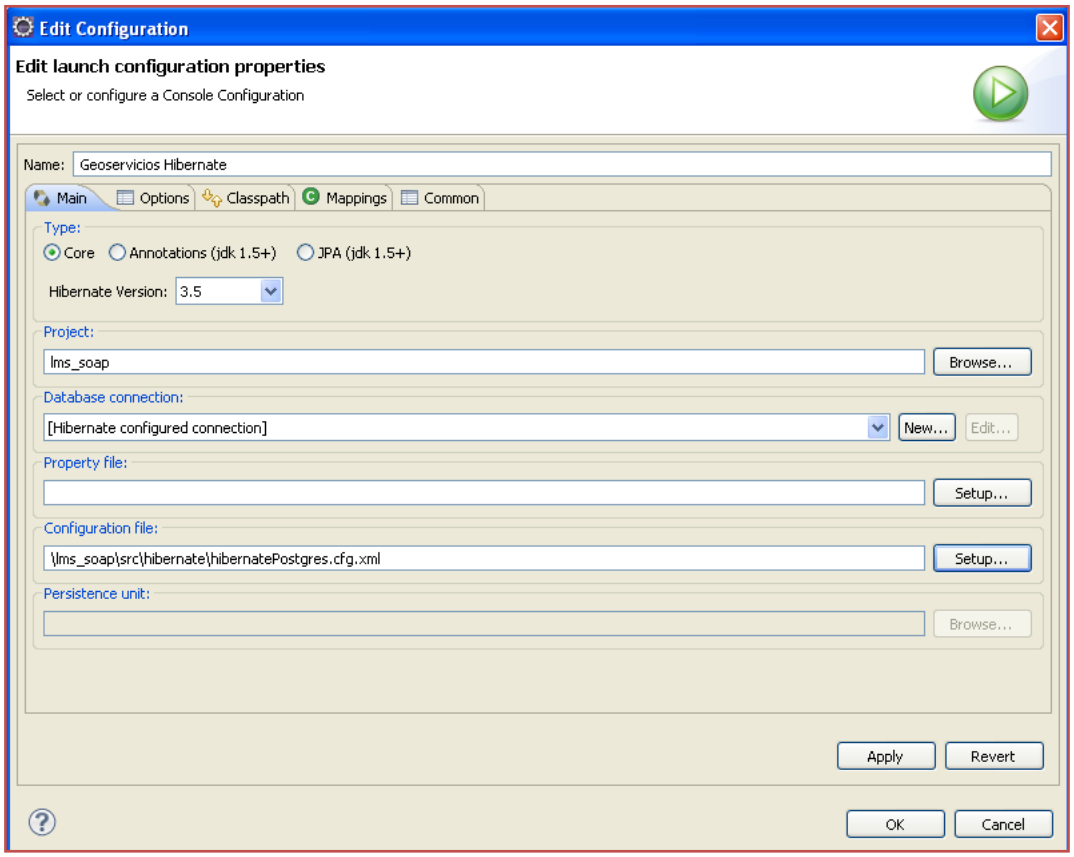


Ilustración 43 Archivo de configuración Hibernate servicio SOAP 4

Los campos a rellenar serán los siguientes:

- **Name→** Le damos un nombre a la configuración.
- **Project→** Seleccionamos nuestro proyecto.
- **Configuration File→** Seleccionamos el archivo de configuración anteriormente creado.

De esta manera veremos como en la perspectiva de Hibernate tenemos acceso a la base de datos y tablas seleccionadas.

A continuación crearemos el archivo de configuración HibernateRevenge.xml. En este archivo tendremos que indicar cuáles son las tablas que queremos mapear. Creamos el archivo en el paquete Hibernate de nuestro proyecto y lo rellenamos introduciendo los siguientes parámetros:

```
<hibernate-reverse-engineering>

    <table-filter match-name="gsgroup"/>

</hibernate-reverse-engineering>
```

De esta manera estamos indicando que queremos mapear la tabla gsgroup.

Una vez configurada la conexión y seleccionadas las tablas que queremos mapear es el momento de que Hibernate genere las clases Java para cada una de las tablas.

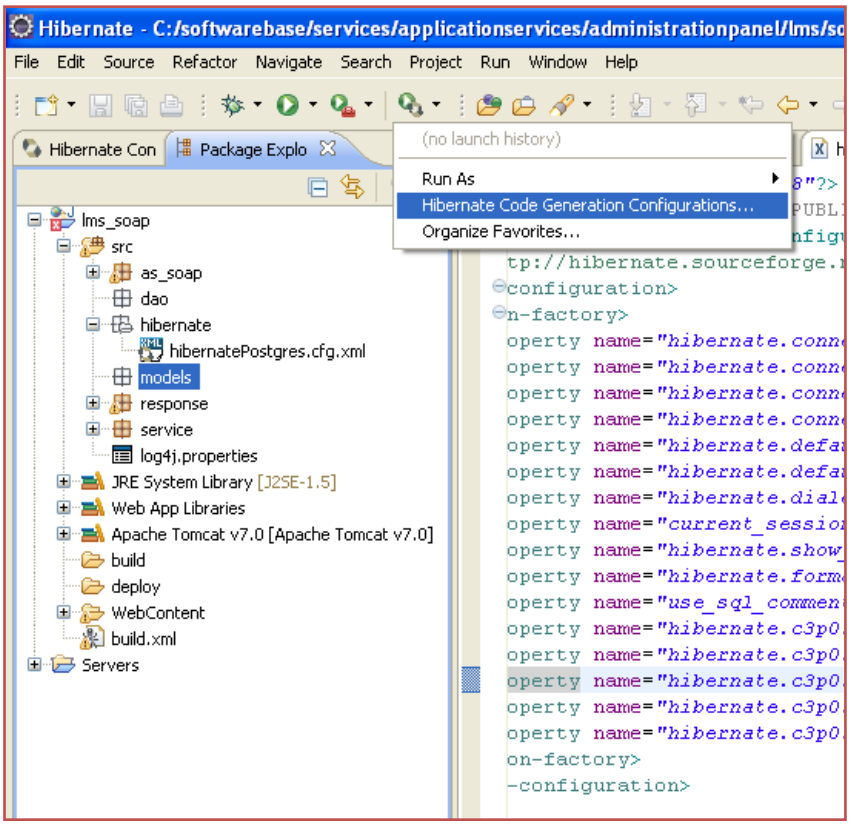


Ilustración 44 Archivo de configuración Hibernate servicio SOAP 5

Para ello desplegaremos el menú marcado en la imagen y seleccionaremos la opción **Hibernate Code Generation Configurations**.

Tal y como muestra la siguiente imagen deberemos rellenar los siguientes parámetros:

- **Name**
- **Console Configuration→** Fichero de configuración
- **Output directory→**(Por defecto)
- **Package→** Paquete donde se van a generar los modelos y ficheros de mapeo
- **Reveng.xml→** Fichero creado anteriormente donde se han seleccionado las tablas a mapear.

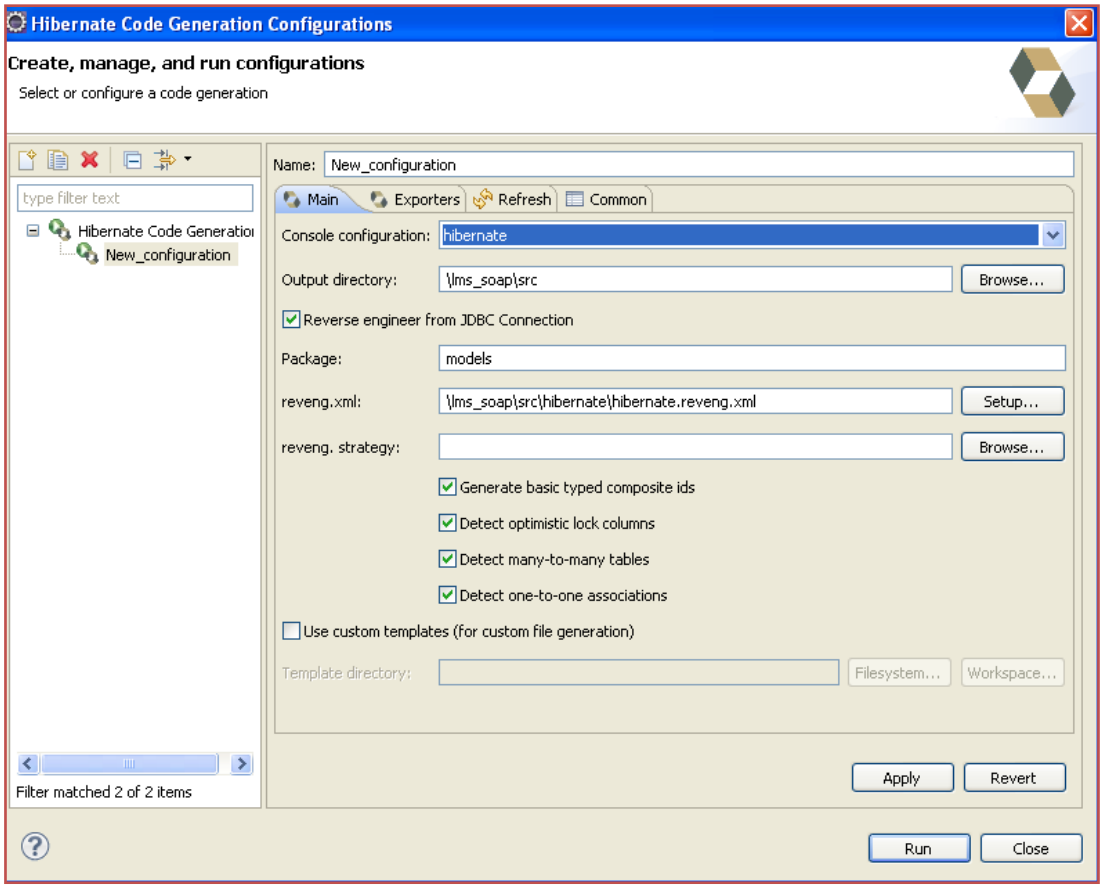


Ilustración 45 Archivo de configuración Hibernate servicio SOAP 6

Y en la pestaña **Exporters** indicamos qué queremos que se genere. En nuestro caso las clases java y los archivos xml con el mapeo.

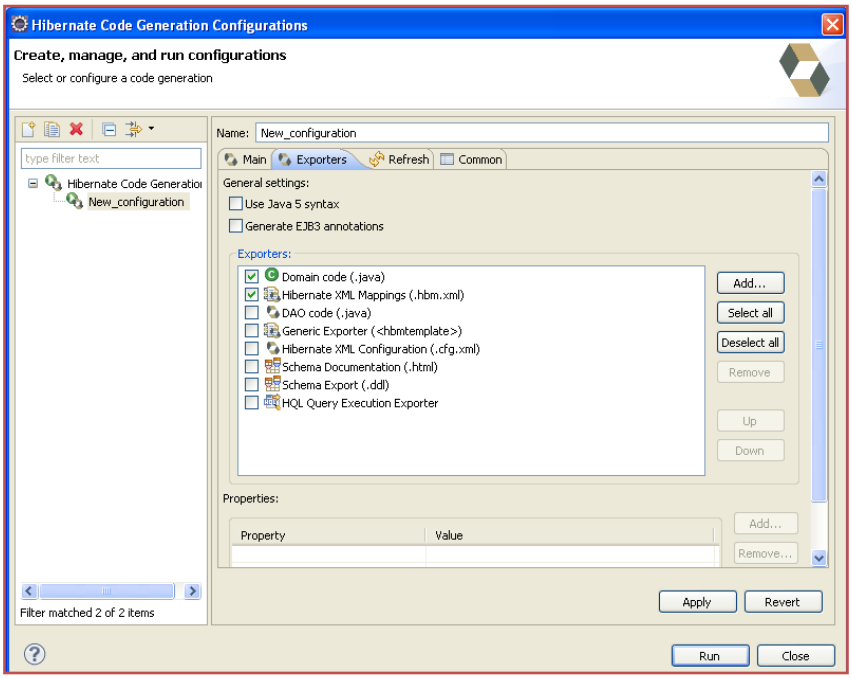


Ilustración 46 Archivo de configuración Hibernate servicio SOAP 7

Una vez pulsado Run, veremos como en el paquete models se han creado las clases necesarias para poder gestionar la tabla que hemos mapeado:

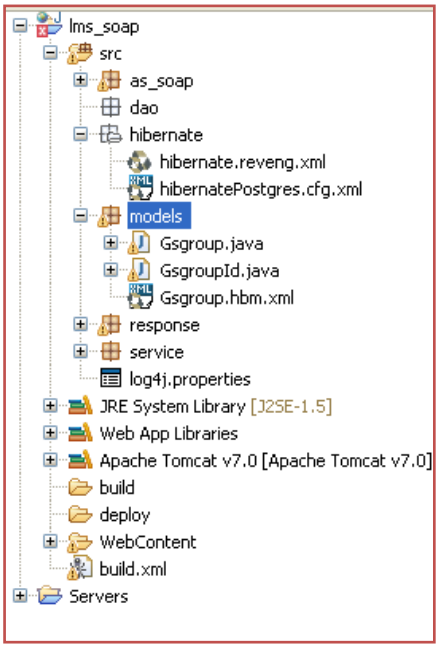


Ilustración 47 Estructura carpetas Hibernate servicio SOAP

A continuación, es muy importante introducir el parámetro mapping en el archivo de configuración hibernatePostgres.cfg.xml. Recordemos que este parámetro nos indicara las tablas que queremos mapear. En nuestro caso añadiríamos la siguiente línea:

```
<mapping resource="models/Gsgroup.hbm.xml"/>
```

Por último realizaremos un pequeño ejemplo para ver cómo podemos acceder a la tabla que hemos mapeado. Para ello en la perspectiva de Hibernate, pulsaremos en el botón HQL

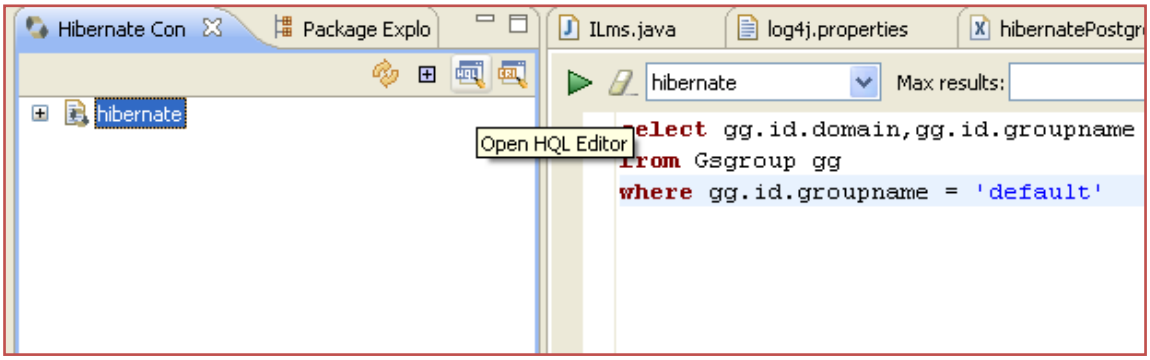


Ilustración 48 Ejemplo consulta Hibernate servicio SOAP

Como podemos ver si insertamos la consulta que aparece en la imagen:

```
select gg.id.domain,gg.id.groupname
from Gsgroup gg
where gg.id.groupname = 'default'
```

Al ejecutar dicha consulta nos devuelve los parámetros de la tabla que hemos seleccionado.

3. CREACIÓN DE UN NUEVO SERVICIO SERVLET

Lo primero que haremos será crear un nuevo dynamic web project. Para ello lo que haremos será pulsar en **New Project** y seleccionarlo en la siguiente ventana:

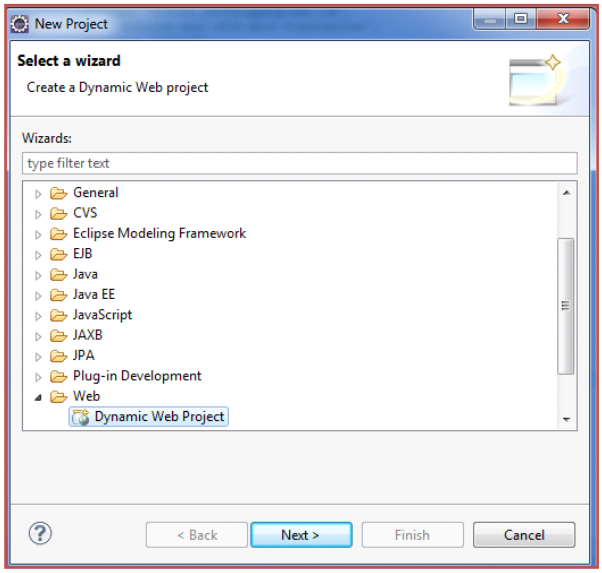


Ilustración 49 Creación proyecto servicio SERVLET

A continuación establecemos el nombre del proyecto. También tenemos que indicar la versión Dynamic web module con la que trabajamos que en este caso es la 2.5

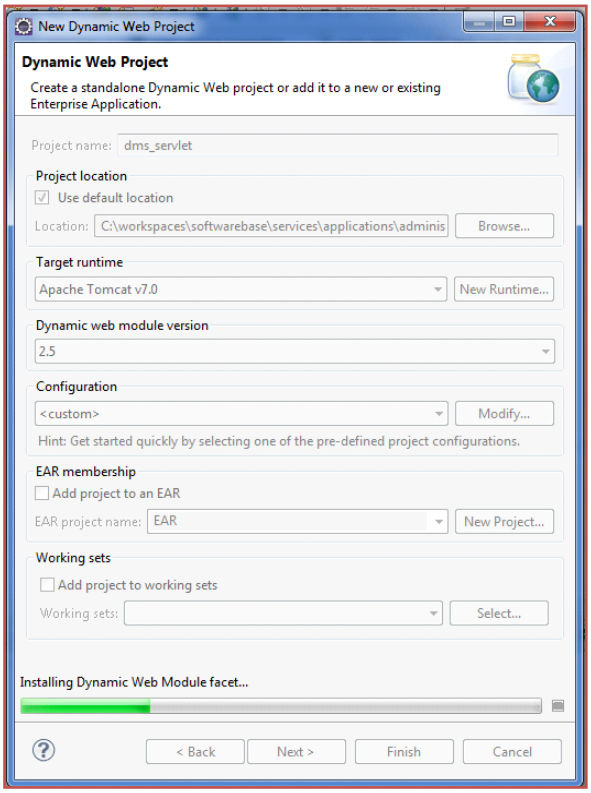


Ilustración 50 Creación proyecto servicio SERVLET 2

Una vez creado el proyecto tendremos que crear tres carpetas dentro de la ubicación src:

- **servicios_soap** -->Aquí tendremos que indicar la capa soap con la que se vaya a comunicar el servicio Servlet (tantos paquetes como soaps queramos comunicar)

- **service** -->Aquí tendremos la lógica del Servlet

- **request**--> en este paquete introduciremos los métodos del servicio web que vayamos a implementar.

En nuestro caso vamos a comunicar el servicio con dos servicios soaps diferentes, por lo tanto crearemos dos paquetes diferentes con los siguientes nombres:

- **as_soap**
- **dms_soap**

Crearemos el primero de ello. Para ello necesitaremos las urls de los servicios soap a utilizar. En nuestro caso serán las siguientes:

- http://100.100.100.165:8080/as_soap_2_0/services/As?wsdl

- http://100.100.100.165:8080/dms_soap/services/Dms?wsdl

Pulsamos encima del paquete creado en **New/Other**

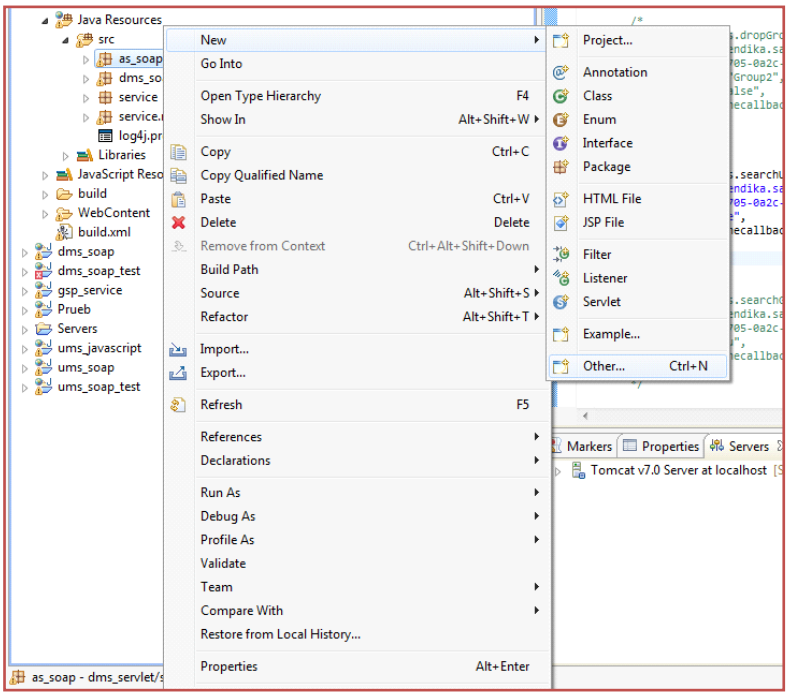


Ilustración 51 Creación web service servicio SERVLET

En la ventana emergente seleccionaremos **Web service client**. En la pantalla que veremos a continuación tendremos que establecer los siguientes parámetros:

- Seleccionar la opción de Apache Axis 2

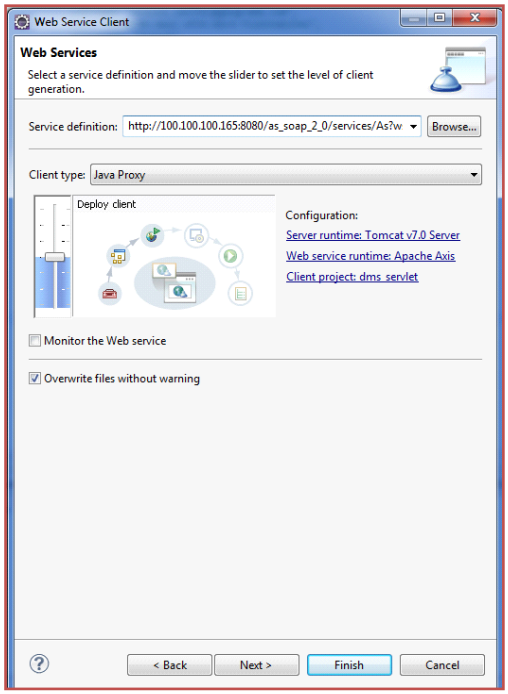


Ilustración 52 Creación web service servicio SERVLET 2

A continuación pulsaremos en next e indicaremos en que paquete queremos que nos cree el proxy al soap. En este caso indicaremos el paquete **as_soap**.

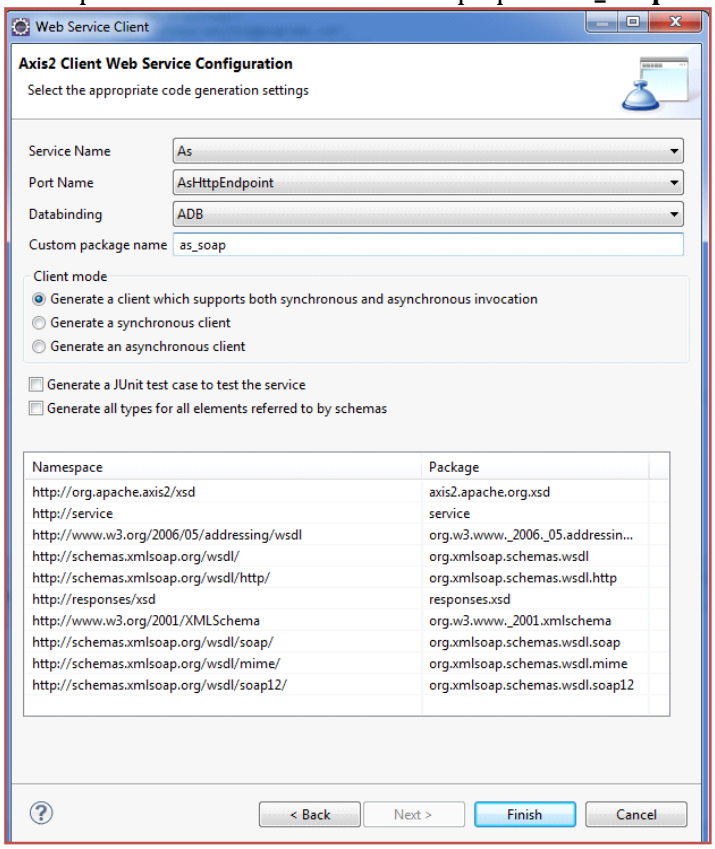


Ilustración 53 Creación web service servicio SERVLET 3

Realizaremos el paso análogo con el servicio web dms_soap.

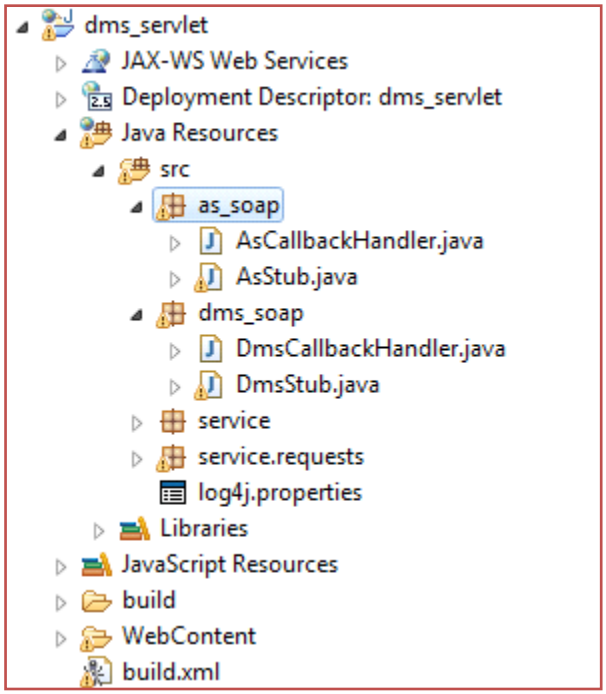


Ilustración 54 Estructura carptetas servicio SERVLET

Una vez tengamos los dos proxies creados, lo que haremos será completar el paquete service. Como ya hemos comentado, dentro de este paquete tendremos la lógica del Servlet. Dicha lógica la compondrán una clase y una interfaz. Estas clases serán:

- DMS
- IDMS

Estas clases tendrán el siguiente contenido:

IDMS:

```
package service;

import java.io.IOException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.json.JSONException;

public interface IDms {
    /*
      Function: request=getGetDataStores

        Sample usage:
        (code)
        (end)

       Parameters:

       Returns {JSON}:

       (end)
    */
    void getDataStores(HttpServletRequest request, HttpServletResponse response,
                String soapUrl) throws JSONException, IOException;
}
```

DMS:

```
package service;

public class Dms extends HttpServlet implements IDms {
    private static final long serialVersionUID = 1L;
    private Logger logger = Logger.getLogger(Dms.class);
    private Properties prop = new Properties();
    private String soapUrl = "";
    private String asUrl = "";
    private String token = "";

    public Dms() {
        super();
        try {
            prop.load(this.getClass().getResourceAsStream("soap_service.properties"));
            soapUrl = prop.getProperty("dms_service");
            asUrl = prop.getProperty("as_service");
            token = prop.getProperty("superToken");
        } catch (IOException e) {
            logger.error("Error reading sas service properties: " +e.getMessage());
        }
     }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
        logger.debug("doGet");
        try {
            proccessResponse(request, response);
        } catch (JSONException e) {
            logger.error("Error: "+e.getMessage());
        }
    }
       protected void doPost(HttpServletRequest request, HttpServletResponse response)
               throws ServletException, IOException {
               response.setHeader("Access-Control-Allow-Origin",
               request.getHeader("Origin"));
               response.addHeader("Access-Control-Allow-Methods",
               request.getHeader("Access-Control-Request-Method"));
               response.addHeader("Access-Control-Allow-Headers",
               request.getHeader("Access-Control-Request-Headers"));
               logger.debug("doPost");
               try {
                       proccessResponse(request, response);
               } catch (JSONException e) {
               logger.error("Error: "+e.getMessage());
               }
       }

       protected void doOptions(HttpServletRequest request, HttpServletResponse
               response){
               response.setHeader("Access-Control-Allow-Origin",
               request.getHeader("Origin"));
               response.addHeader("Access-Control-Allow-Methods",
               request.getHeader("Access-Control-Request-Method"));
               response.addHeader("Access-Control-Allow-Headers",
               request.getHeader("Access-Control-Request-Headers"));
               return;
     }

       private void proccessResponse(HttpServletRequest request, HttpServletResponse
                       response) throws IOException, JSONException{
               response.setContentType("application/json");
               String host = request.getHeader("REFERER");
               if(host == null){
                       host = request.getServerName();
               }else{
                       host = host.split("/")[2];
               }
               String ip = request.getHeader("X-Forwarded-For");
               if (ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip)) {
                       ip = request.getRemoteAddr();
               }
       }
```

También tendremos que crear un fichero de tipo properties, donde indicaremos la ruta donde tenemos desplegados los servicios soap pertinentes. Este archivo contendrá lo siguiente:

```
dms_service=http://localhost:8080/dms_soap/services/Dms
as_service=http://100.100.100.165:8080/as_soap_2_0/services/As
superToken=xxx-xxx-xxx
```

Tras esto, procederemos a rellenar el paquete request. Como prueba crearemos el siguiente método

- CreateDataStore

El contenido será el siguiente:

```
package service.requests;

public class GetDataStores {

    String CONTENT_TYPE = "application/json";
    private Logger logger = Logger.getLogger(GetDataStores.class);

    public GetDataStores(){
    }

    public void processRequest(HttpServletRequest request, HttpServletResponse
response, String soapUrl, String token, String username) throws JSONException,
IOException{

        logger.info("Starting method");
        response.setContentType(CONTENT_TYPE);
        logger.debug("Getting parameters from the request");
        String callback = request.getParameter("callback");
        String filter = request.getParameter("filter");
        String extendedInfoStr =  request.getParameter("extendedInfo");
        boolean extendedInfo = false;
        if(extendedInfoStr != null &&
extendedInfoStr.equalsIgnoreCase("true")){extendedInfo = true;}
    }

}
```

Por último tendremos que añadir las librerías de Hibernate y de Apache Axis2 para que el servicio sea compatible con la capa soap creada.

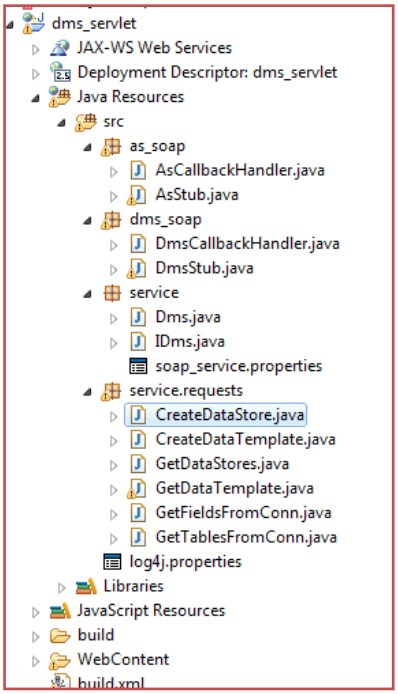


Ilustración 55 Estructura carpetas servicio SERVLET 2

4. CREACIÓN DE UN NUEVO SERVICIO JAVASCRIPT

Lo primero que haremos será crear un nuevo **dynamic web project**. Para ello lo que haremos será pulsar en **New project** y seleccionarlo en la siguiente ventana:

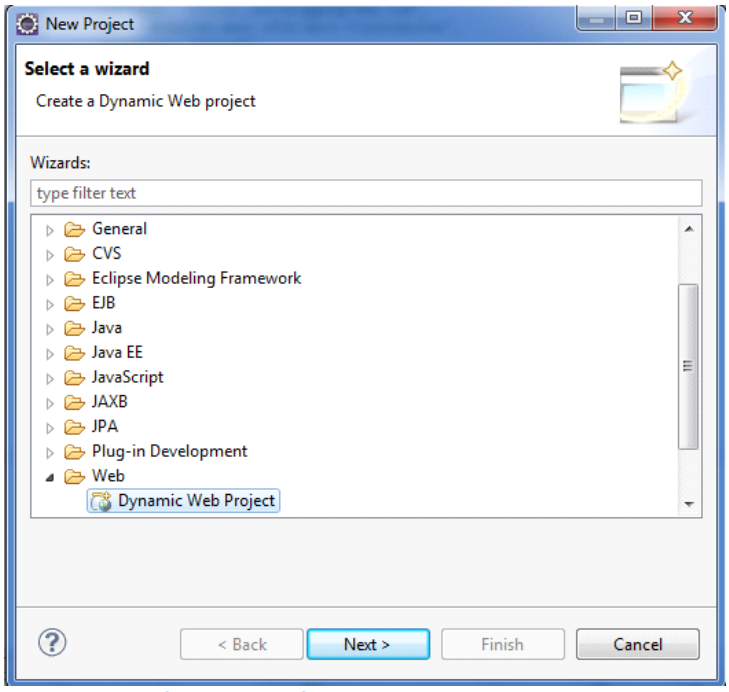


Ilustración 56 Creación proyecto servicio JAVASCRIPT

A continuación establecemos el nombre del proyecto. También tenemos que indicar la versión **Dynamic web module** con la que trabajamos que en este caso es la 2.5

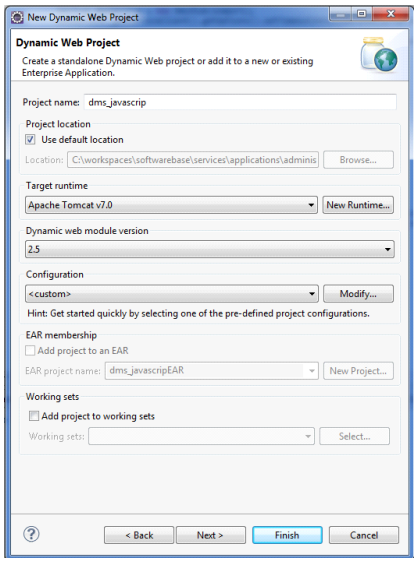


Ilustración 57 Creación proyecto servicio JAVASCRIPT 2

Tras esto, crearemos la carpeta js dentro de la ubicación webcontent. Dentro de esta carpeta crearemos un archivo javascript llamado:

- ums_api.js

Este archivo contendrá lo siguiente:

```
if(GEOSERVICIOS == null){
    var GEOSERVICIOS = {};
}
GEOSERVICIOS.dms = {

    //Private, the url of the gs Servlet
    baseurl: "http://127.0.0.1:8080/dms_servlet/services/Dms",

    /*
      Function: request=getDataStores

       Parameters:

       Returns {JSON}:
        (code)
        (code)
     */

    getDataStores: function(options){
        var oThis = this;
        var request = "getDataStores";
        var username = options.username;
        var key = options.key;
        var filter = options.filter;
        var extendedInfo = options.extendedInfo;
        if(options.method == "POST"){
            var params = request="+request+"&username="+username+"&key="+key+"
               &filter="+filter+"&extendedInfo="+extendedInfo;
            this.sendPost(params, options);
        }
        else{
            var random = Math.floor((Math.random()*100000)+1);
            var idRequest = "sas_" + random ;
            url = this.baseurl+"?callback=GEOSERVICIOS.dms."+idRequest+"
               &request="+request+"&username="+username+
               "&key="+key+"&filter="+filter+"&extendedInfo="+extendedInfo;
                GEOSERVICIOS.dms.endRequest(response,idRequest,options);
            };
            this.sendRequest(url,idRequest);
        }
    },
```

Dentro de este script, crearemos todos los métodos que contenga el servicio creado.

También crearemos un index y un test para realizar las comprobaciones de buen funcionamiento del servicio. Hay que recordar que será necesario incluir las bibliotecas necesarias para que el servicio funcione correctamente.

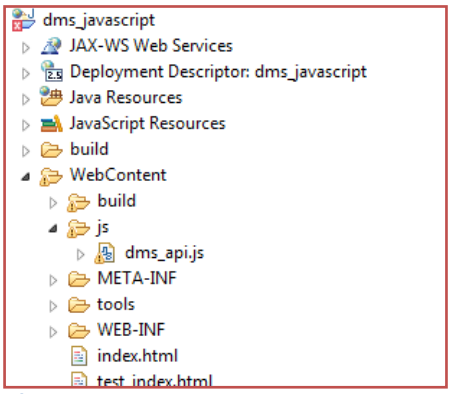


Ilustración 58 Estructura carpetas servicio JAVASCRIPT

ANEXO III: COMPONENTES MÓDULO DE USUARIO Y GRUPOS

En el siguiente Anexo, se detalla el diseño y especificación de cada uno de los componentes del módulo de usuarios y grupos. En dicho anexo, se detalla tanto la interfaz gráfica, los casos de uso como los diagramas de secuencia de cada uno de los componentes.

Los componentes del módulo de usuarios y grupos que se especifican, son los siguientes:

Componentes relativos a los usuarios:

- ListUserPanel
- EditUserPanel
- ProfileUserPanel
- AddUserPanel

Componentes relativos a grupos:

- ListGroupPanel
- EditGroupPanel
- ProfileGroupPanel
- AddGroupPanel

Cada uno de los componentes estará asociado a ciertos servicios que se irán identificando al realizar los casos de uso pertinentes. Dichos servicios están especificados en el Anexo IV: Servicios módulo de usuarios y grupos.

1. COMPONENTE LISTUSERPANEL

El siguiente componente, se encargará de mostrar el listado de usuarios relacionados con una determinada cuenta de la Plataforma GeoServicios Online, y la funcionalidad de búsqueda en dicho listado. Por otro lado también se encarga del borrado, acceso al perfil y edición de usuarios.

1.1. DISEÑO INTERFAZ GRÁFICA

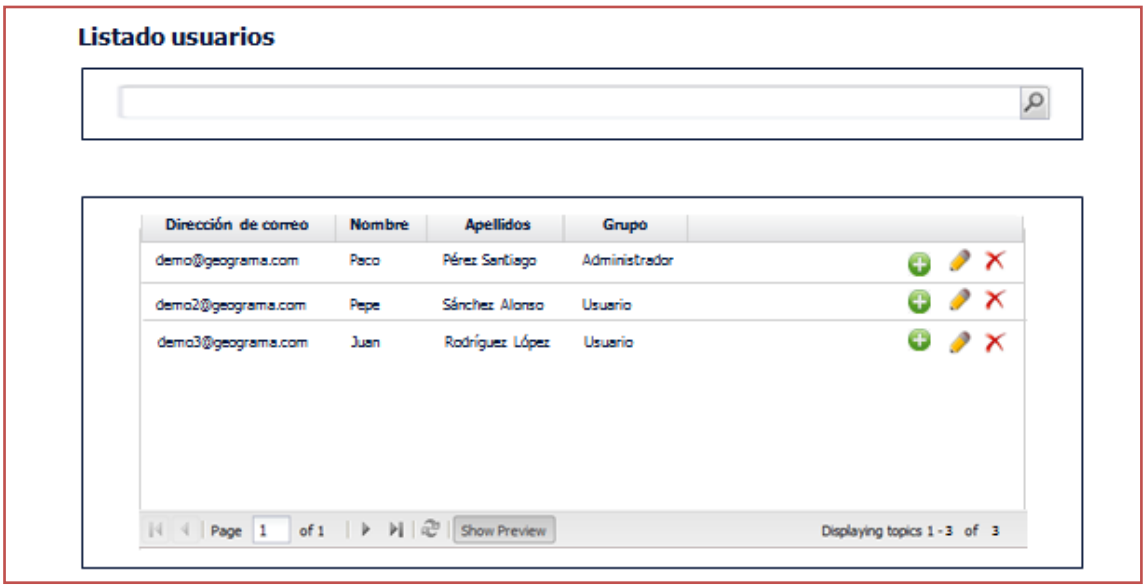


Ilustración 59 Diseño interfaz ListUserPanel

1.2. CASO DE USO

Las funcionalidades identificados en el siguiente componente son las que siguen:

- **Pedir lista de usuarios**
- **Borrar usuario**
- **Buscar usuario**
- **Ver perfil**
- **Editar usuario**

El diagrama de dicho caso de uso será el siguiente:

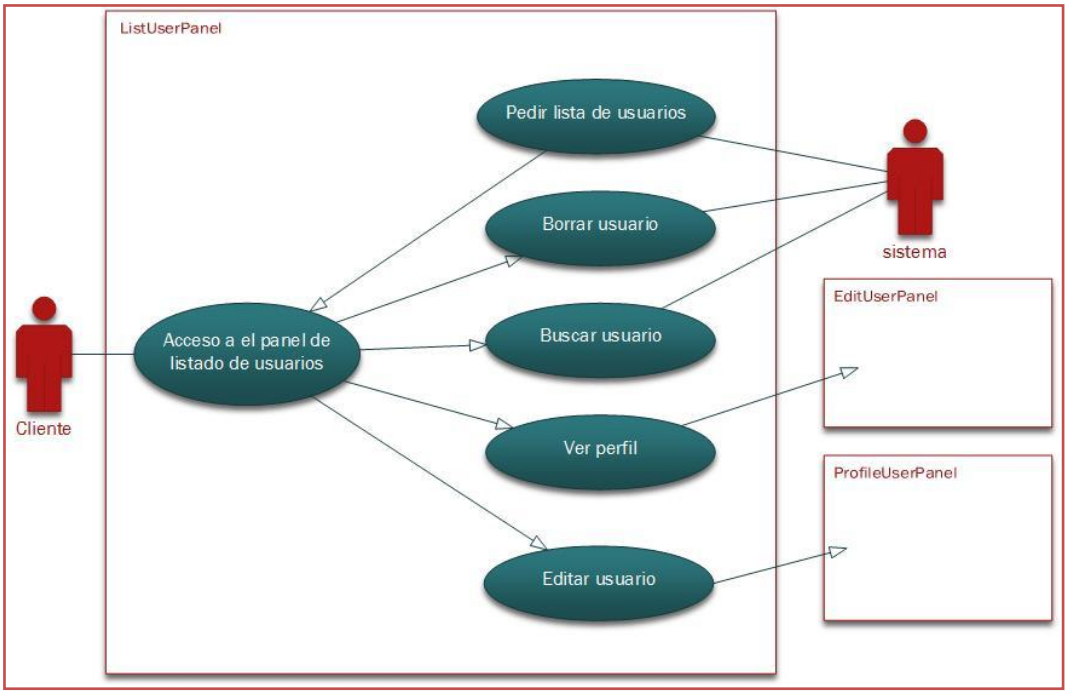


Ilustración 60 Caso de uso ListUserPanel

Pedir lista de usuarios: El sistema se encargará de devolver una lista de usuarios y mostrarla en un listado. Mediante esta funcionalidad identificamos el siguiente servicio:

- getUsers

Borrar usuario: Esta funcionalidad, se encargará del borrado de usuario. Mediante esta funcionalidad identificamos el siguiente servicio:

- dropUser

Buscar usuario: Esta funcionalidad, se encargará de la búsqueda de usuarios dentro del listado devuelto por el sistema. Mediante esta funcionalidad identificamos el siguiente servicio:

- searchUser

Ver perfil: Esta funcionalidad, se encargará del acceso al componente ProfileUserPanel, el cual se encargará de mostrar los datos de los usuarios.

Editar usuario: Esta funcionalidad , se encargará del acceso al componente EditUserPanel, el cual se encargará de la edición de usuarios.

1.3. DIAGRAMA DE SECUENCIA

Mediante el análisis realizado mediante el caso de uso e identificación de funcionalidades, procederemos a crear el diagrama de secuencia que será el siguiente.

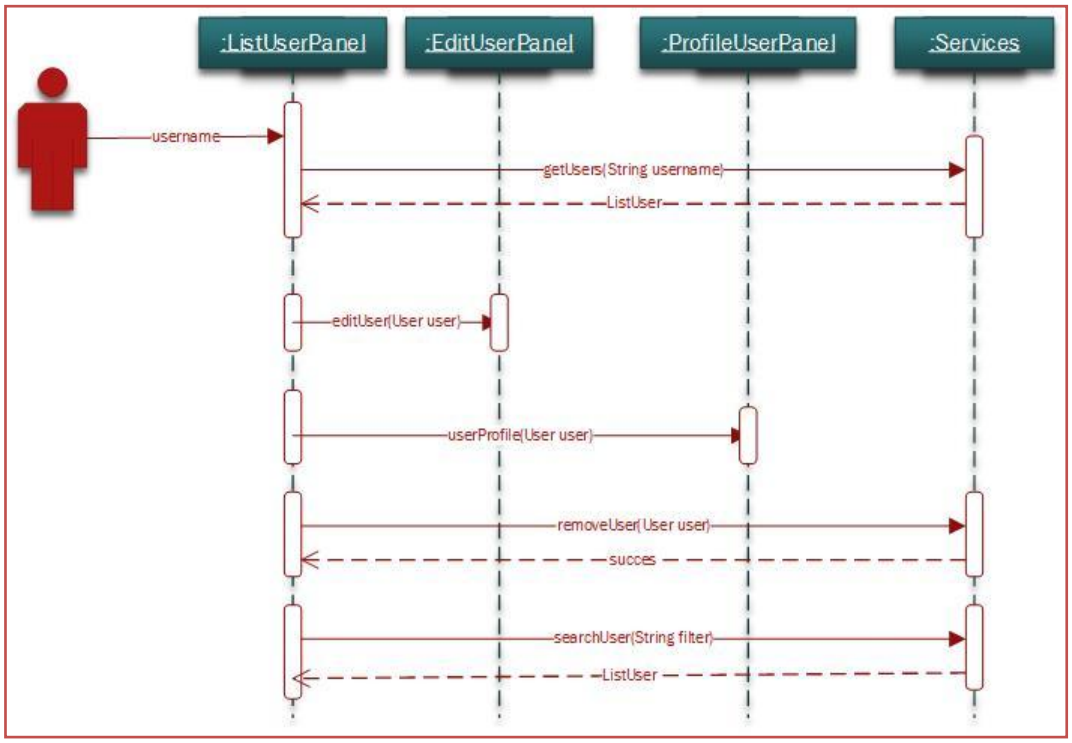


Ilustración 61Diagrama de secuencia ListUserPanel

1.4. RESULTADO FINAL

El resultado final del componente una vez integrado, es el que se puede observar a continuación:

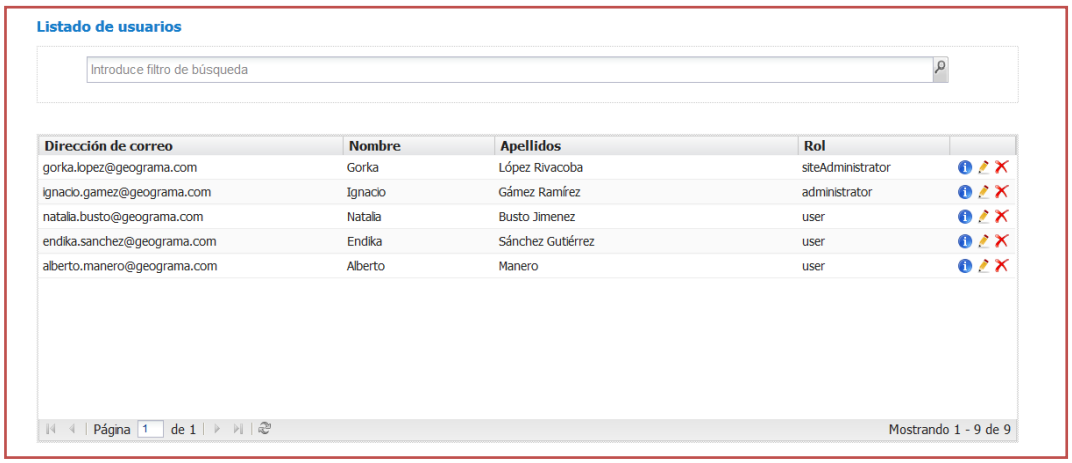


Ilustración 62 Resultado final ListUserPanel

2. COMPONENTE EDITUSERPANEL

El siguiente componente, se encargará de edición de un determinado usuario relacionado con una cuenta de la Plataforma GeoServicios Online.

2.1. DISEÑO INTERFAZ GRÁFICA

Ilustración 63 Diseño interfaz EditUserPanel

2.2. CASO DE USO

Las funcionalidades identificados en el siguiente componente son las que siguen:

- **Recuperar datos de usuario a editar**
- **Obtener grupos disponibles**
- **Guardar usuario**

El diagrama de dicho caso de uso será el siguiente:

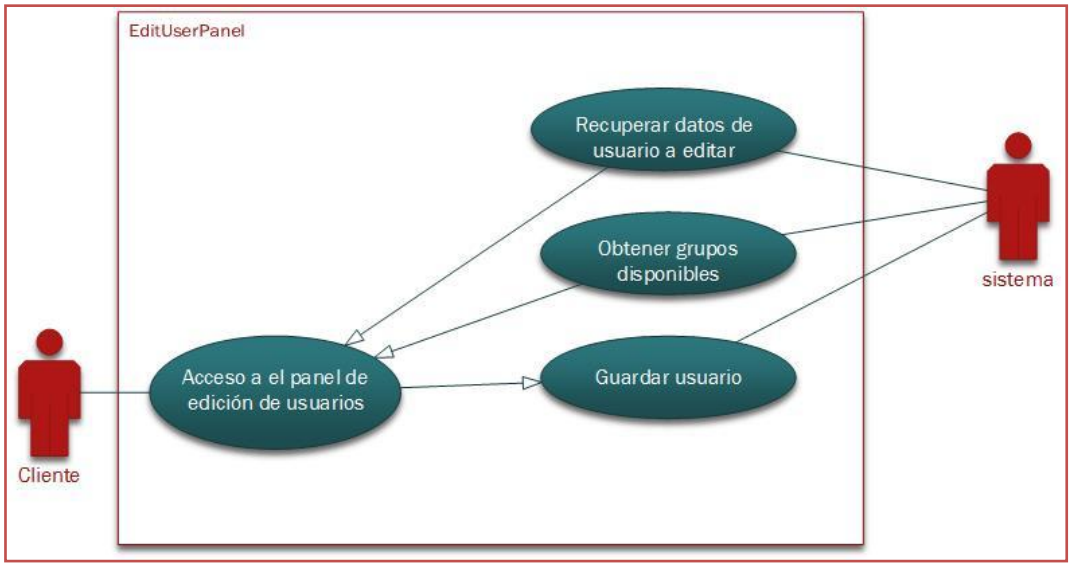


Ilustración 64 Caso de uso EditUserPanel

Recuperar datos de usuario a editar: El componente ListUserPanel se encargará de pasar la información relacionada al usuario seleccionada al componente EditUserPanel- Dicho componente se encargará de mostrar los datos obtenidos.

Obtener grupos disponibles: Esta funcionalidad, se encargará de obtener el listado de grupos disponibles relacionados con la cuenta de usuario de la Plataforma GeoServicios Online. Mediante esta funcionalidad identificamos el siguiente servicio:

- getGroups

Guardar usuario: Esta funcionalidad, se encargará de la modificación del usuario seleccionado. Mediante esta funcionalidad identificamos el siguiente servicio:

- modifyUser

2.3. DIAGRAMA DE SECUENCIA

Mediante el análisis realizado mediante el caso de uso e identificación de funcionalidades, procederemos a crear el diagrama de secuencia que será el siguiente.

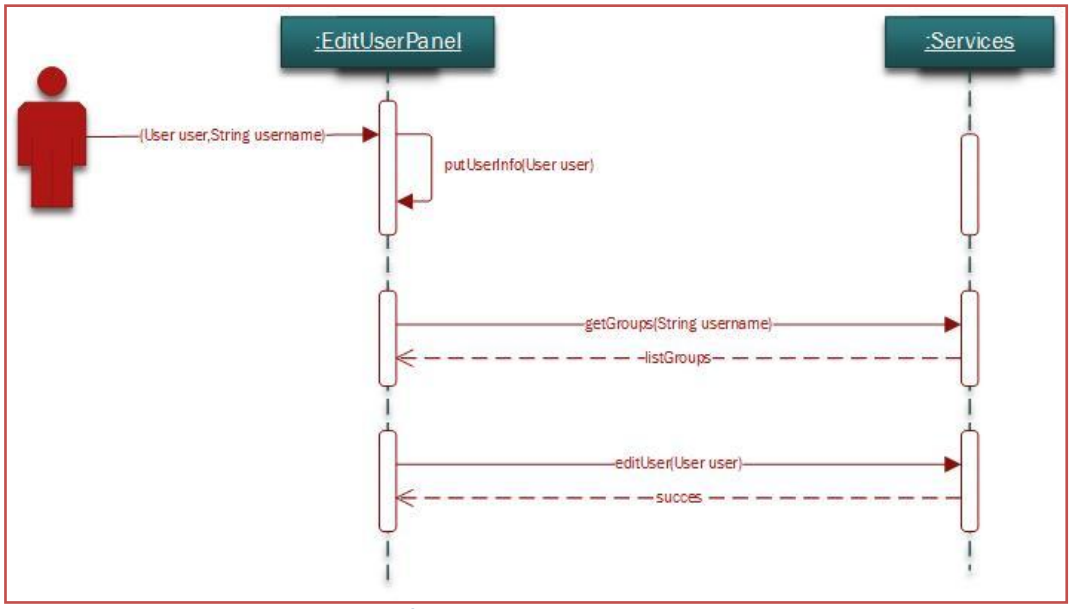


Ilustración 65 Caso de uso EditUserPanel

2.4. *RESULTADO FINAL*

El resultado final del componente una vez integrado, es el que se puede observar a continuación:

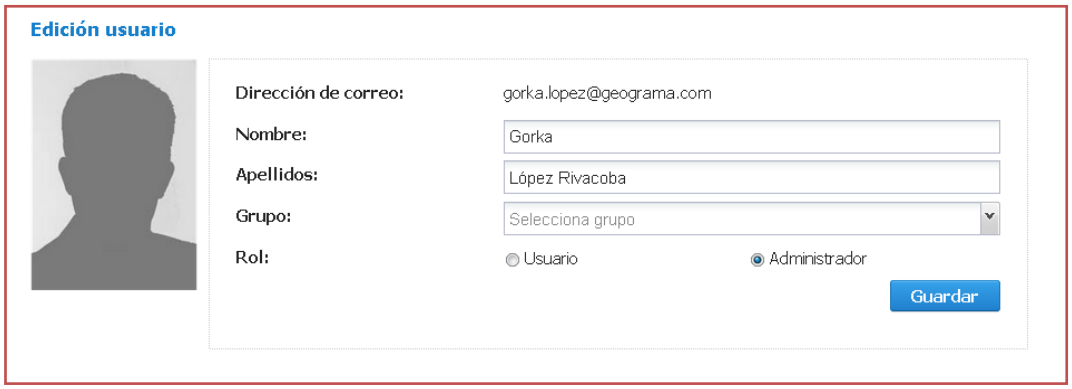


Ilustración 66 Resultado final EditUserPanel

3. COMPONENTE PROFILEUSERPANEL

El siguiente componente, se encargará de mostrar los datos de un determinado usuario relacionado con una cuenta de la Plataforma GeoServicios Online.

3.1. DISEÑO INTERFAZ GRÁFICA

Ilustración 67 Diseño interfaz ProfileUserPanel

3.2. CASO DE USO

Las funcionalidades identificados en el siguiente componente son las que siguen:

- **Recuperar datos de usuario a editar**
- **Editar usuario**

El diagrama de dicho caso de uso será el siguiente:

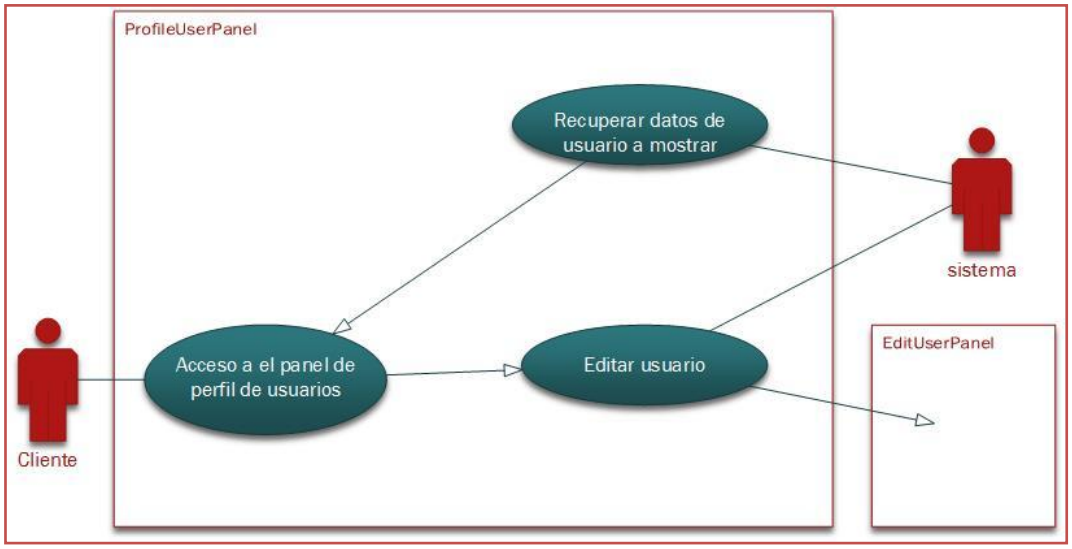


Ilustración 68 Caso de uso ProfileUserPanel

Recuperar datos de usuario a mostrar: El componente ListUserPanel se encargará de pasar la información relacionada al usuario seleccionada al componente ProfileUserPanel. Dicho componente se encargará de mostrar los datos obtenidos.

Editar usuario: Esta funcionalidad , se encargará del acceso al componente EditUserPanel, el cual se encargará de la edición de usuarios.

3.3. DIAGRAMA DE SECUENCIA

Mediante el análisis realizado mediante el caso de uso e identificación de funcionalidades, procederemos a crear el diagrama de secuencia que será el siguiente.

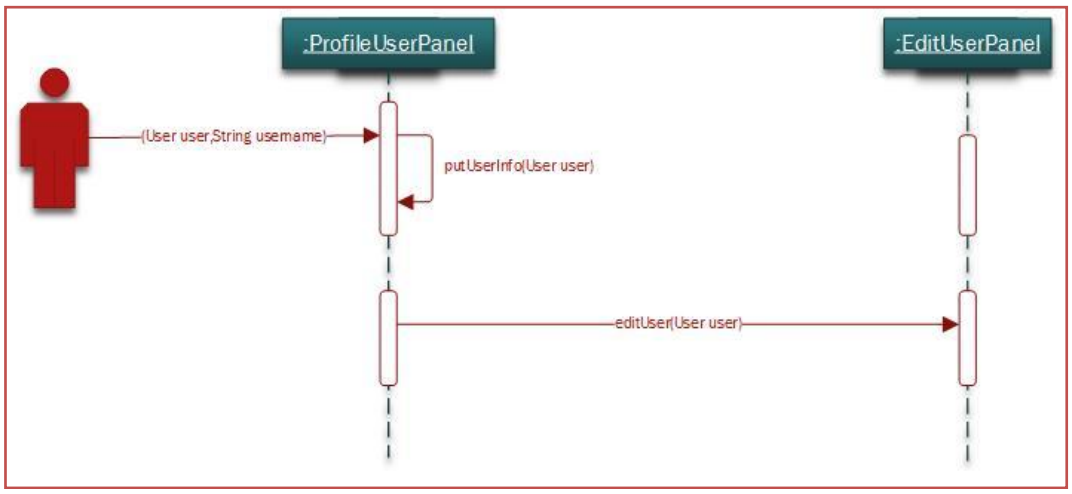


Ilustración 69 Diagrama de secuencia ProfileUserPanel

3.4. RESULTADO FINAL

El resultado final del componente una vez integrado, es el que se puede observar a continuación:

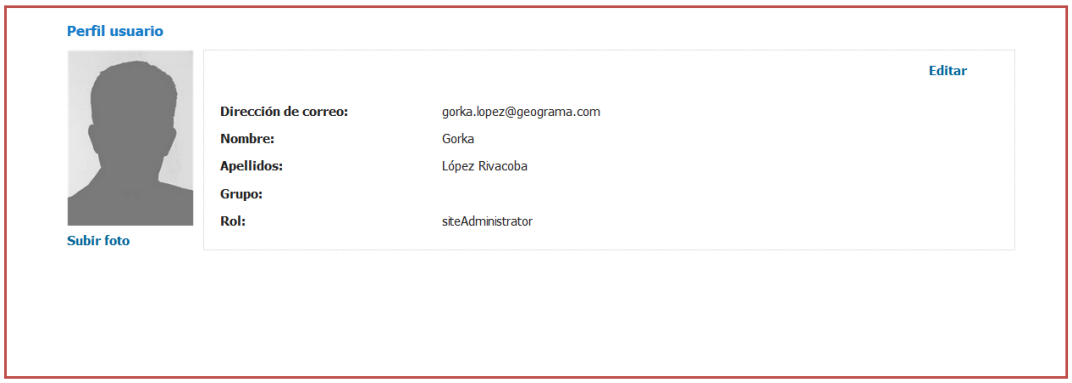


Ilustración 70 Resultado final ProfileUserPanel

4. COMPONENTE ADDUSERPANEL

El siguiente componente, se encargará de la creación de un nuevo usuario relacionado con una cuenta de la Plataforma GeoServicios Online.

4.1. DISEÑO INTERFAZ GRÁFICA

Ilustración 71 Diseño interfaz AddUserPanel

4.2. CASO DE USO

Las funcionalidades identificados en el siguiente componente son las que siguen:

- **Obtener grupos disponibles**
- **Guardar usuario**

El diagrama de dicho caso de uso será el siguiente:

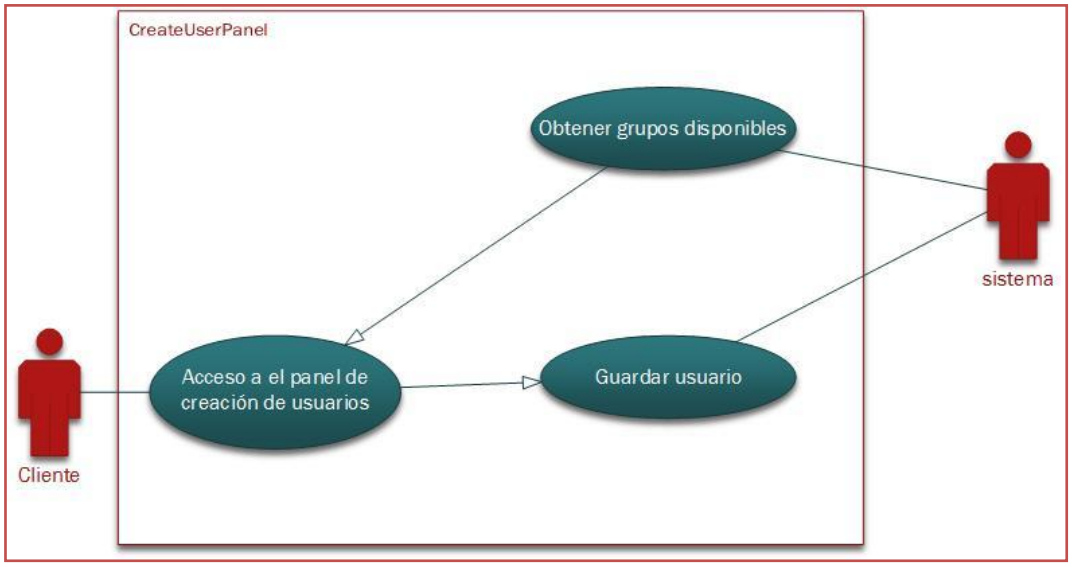


Ilustración 72 Caso de uso AddUserPanel

Obtener grupos disponibles: Esta funcionalidad, se encargará de obtener el listado de grupos disponibles relacionados con la cuenta de usuario de la Plataforma GeoServicios Online. Mediante esta funcionalidad identificamos el siguiente servicio:

- getGroups

Guardar usuario: Esta funcionalidad, se encargará de almacenar la información del nuevo usuario. Mediante esta funcionalidad identificamos el siguiente servicio:

- createUser

4.3. DIAGRAMA DE SECUENCIA

Mediante el análisis realizado mediante el caso de uso e identificación de funcionalidades, procederemos a crear el diagrama de secuencia que será el siguiente.

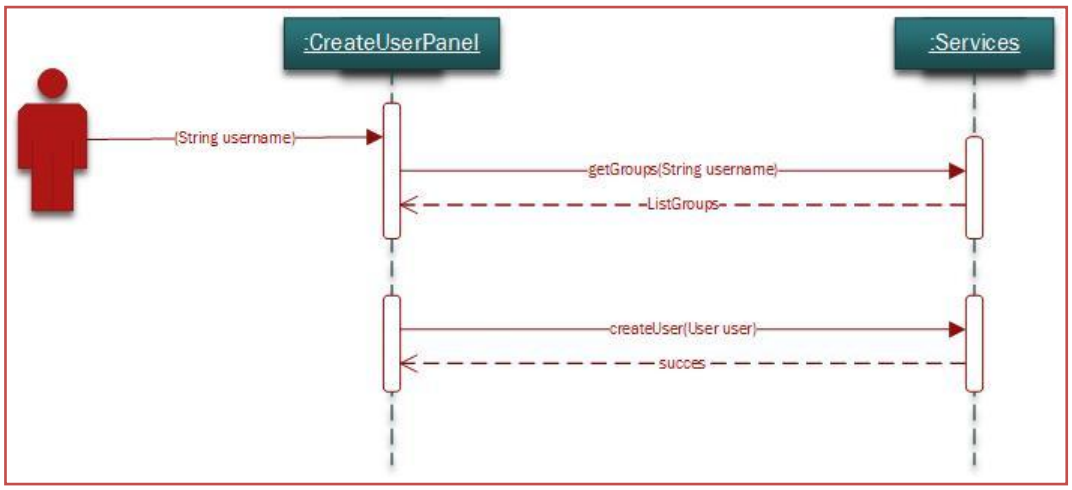


Ilustración 73 Diagrama de secuencia AddUserPanel

4.4. RESULTADO FINAL

El resultado final del componente una vez integrado, es el que se puede observar a continuación:

Ilustración 74 Resultado final AddUserPanel

5. COMPONENTE LISTGROUPPANEL

El siguiente componente, se encargará de mostrar el listado de grupos relacionados con una determinada cuenta de la Plataforma GeoServicios Online, y la funcionalidad de búsqueda en dicho listado. Por otro lado también se encarga del borrado, acceso al perfil y edición de grupos.

5.1. DISEÑO INTERFAZ GRÁFICA

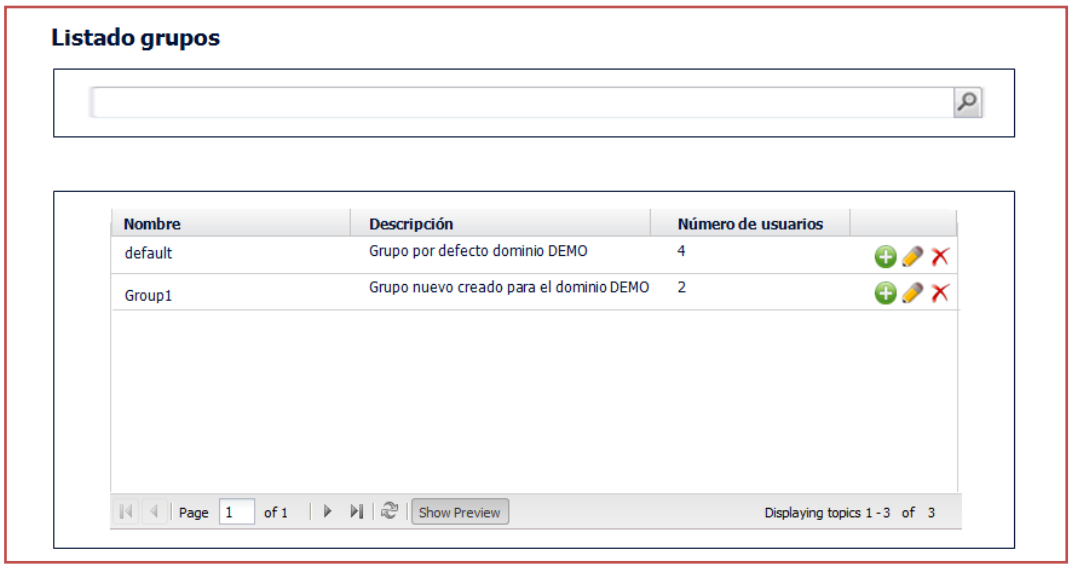


Ilustración 75 Diseño interfaz ListGroupPanel

5.2. CASO DE USO

Las funcionalidades identificados en el siguiente componente son las que siguen:

- **Pedir lista de grupos**
- **Borrar grupo**
- **Buscar grupo**
- **Ver perfil**
- **Editar grupo**

El diagrama de dicho caso de uso será el siguiente:

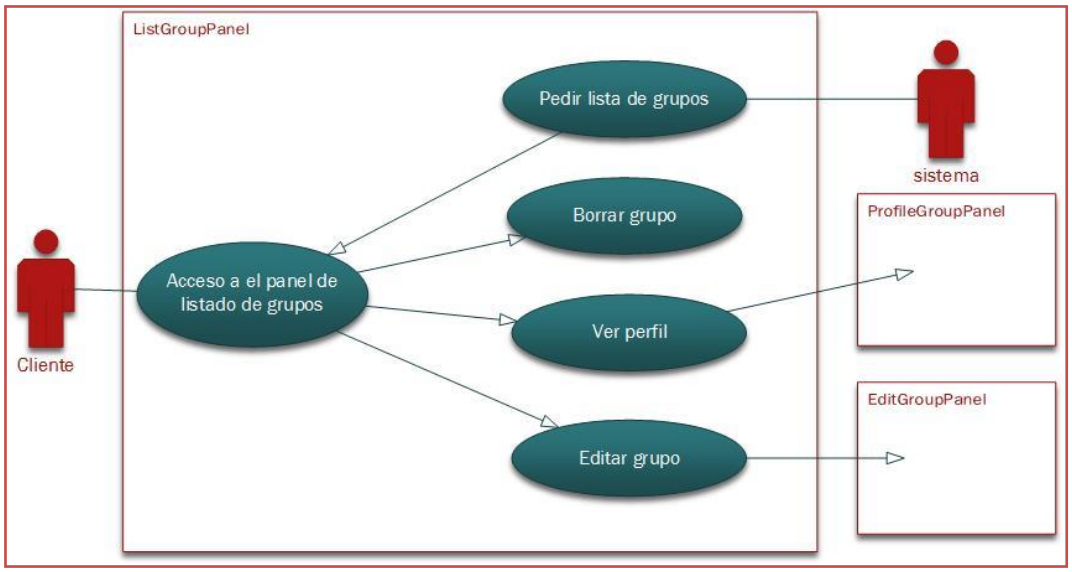


Ilustración 76 Caso de uso ListGroupPanel

Pedir lista de grupos: El sistema se encargará de devolver una lista de grupos y mostrarla en un listado. Mediante esta funcionalidad identificamos el siguiente servicio:

- getGroups

Borrar grupo: Esta funcionalidad, se encargará del borrado de grupos. Mediante esta funcionalidad identificamos el siguiente servicio:

- dropGroup

Buscar grupo: Esta funcionalidad, se encargará de la búsqueda de grupos dentro del listado devuelto por el sistema. Mediante esta funcionalidad identificamos el siguiente servicio:

- searchGroup

Ver perfil: Esta funcionalidad, se encargará del acceso al componente ProfileGroupPanel, el cual se encargará de mostrar los datos de los grupos.

Editar grupo: Esta funcionalidad , se encargará del acceso al componente EditGroupPanel, el cual se encargará de la edición de grupos.

5.3. DIAGRAMA DE SECUENCIA

Mediante el análisis realizado mediante el caso de uso e identificación de funcionalidades, procederemos a crear el diagrama de secuencia que será el siguiente.

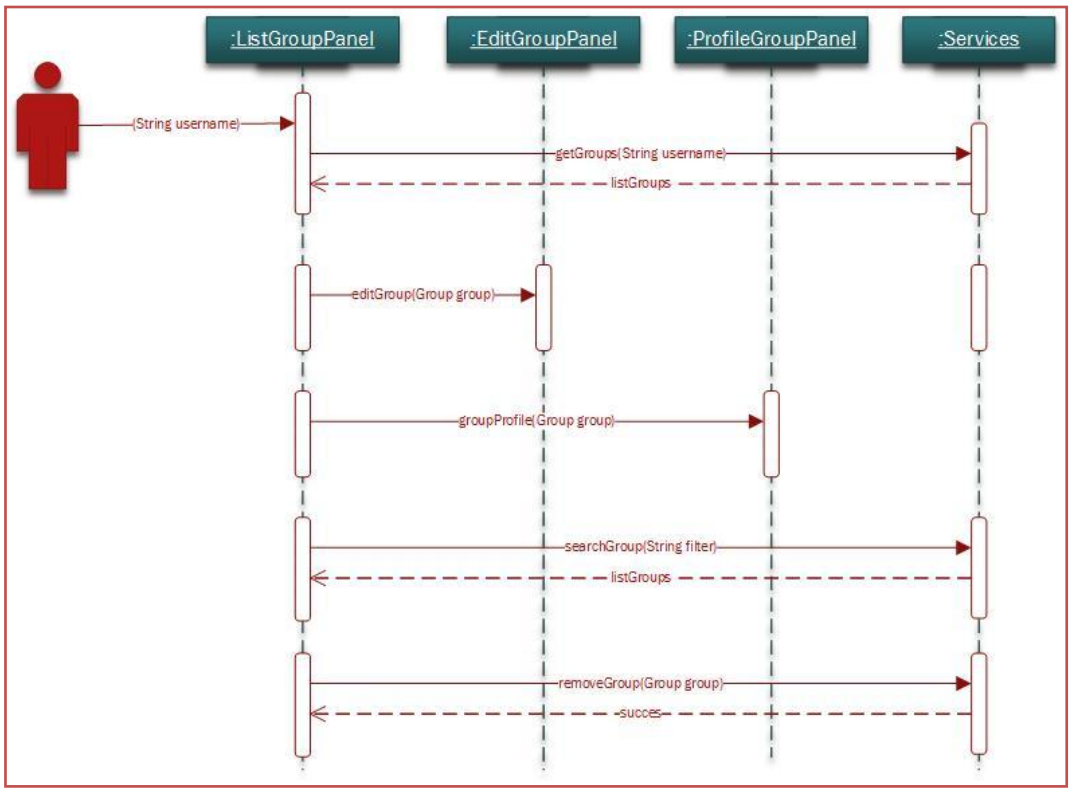


Ilustración 77 Diagrama de secuencia ListGroupPanel

5.4. RESULTADO FINAL

El resultado final del componente una vez integrado, es el que se puede observar a continuación:

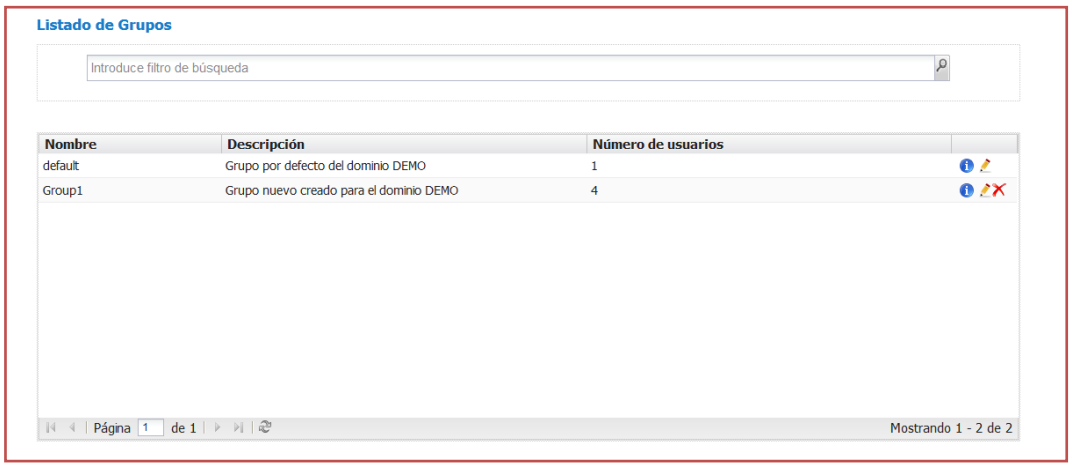


Ilustración 78 Resultado final ListGroupPanel

6. COMPONENTE EDITGROUPPANEL

El siguiente componente, se encargará de edición de un determinado grupo relacionado con una cuenta de la Plataforma GeoServicios Online.

6.1. DISEÑO INTERFAZ GRÁFICA

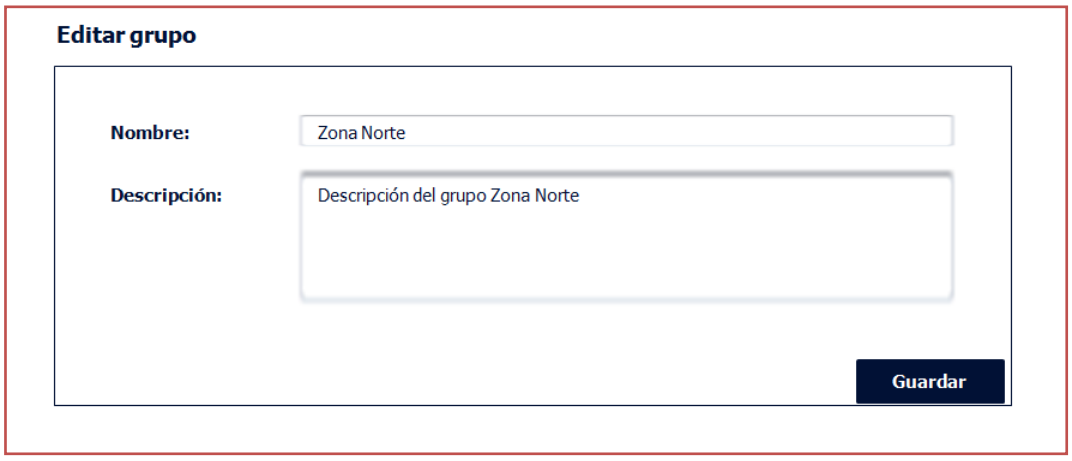


Ilustración 79 Diseño interfaz EditGroupPanel

6.2. CASO DE USO

Las funcionalidades identificados en el siguiente componente son las que siguen:

- **Recuperar datos de grupo a editar**
- **Guardar grupo**

El diagrama de dicho caso de uso será el siguiente:

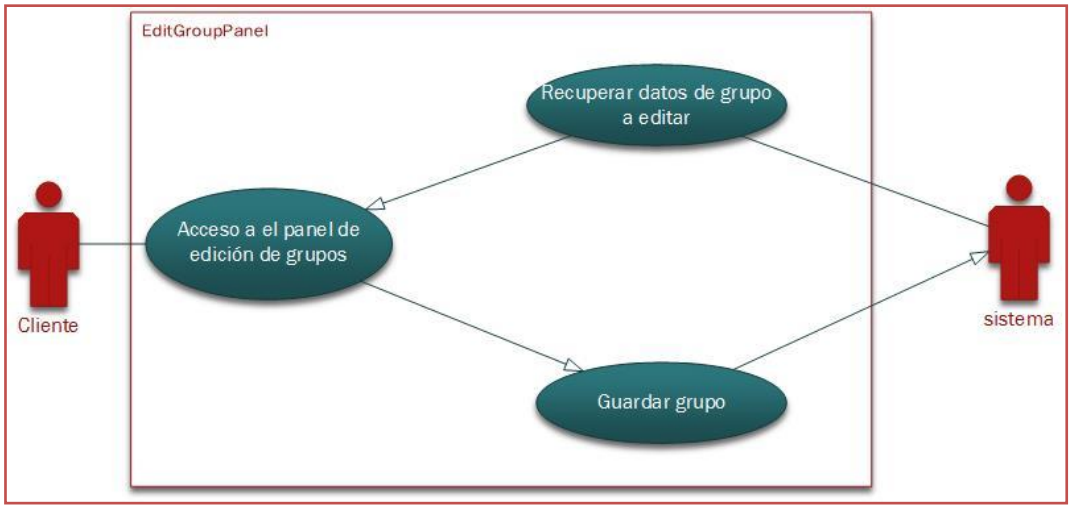


Ilustración 80 Caso de uso EditGroupPanel

Recuperar datos de grupo a editar: El componente ListGroupPanel se encargará de pasar la información relacionada al grupo seleccionado al componente EditGroupPanel- Dicho componente se encargará de mostrar los datos obtenidos.

Guardar grupo: Esta funcionalidad, se encargará de la modificación del grupo seleccionado. Mediante esta funcionalidad identificamos el siguiente servicio:

- modifyGroup

6.3. DIAGRAMA DE SECUENCIA

Mediante el análisis realizado mediante el caso de uso e identificación de funcionalidades, procederemos a crear el diagrama de secuencia que será el siguiente.

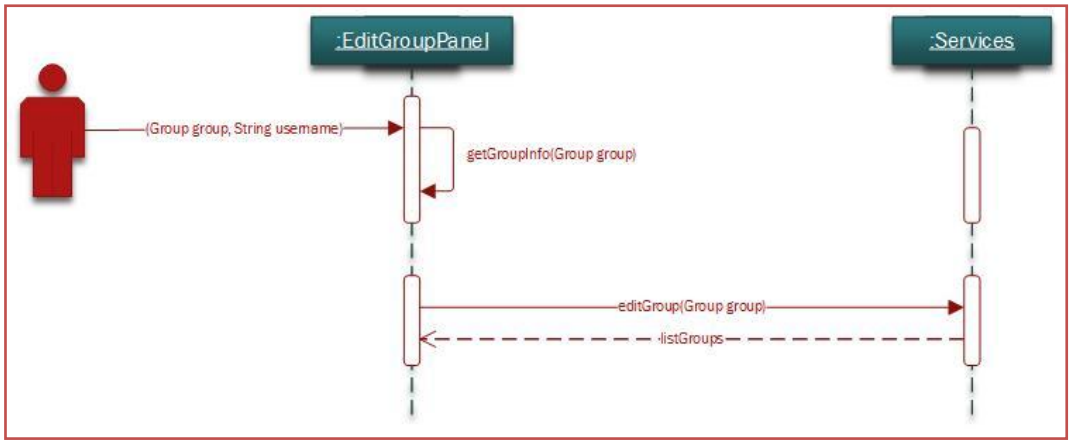


Ilustración 81 Diagrama de secuencia EditGroupPanel

6.4. RESULTADO FINAL

El resultado final del componente una vez integrado, es el que se puede observar a continuación:

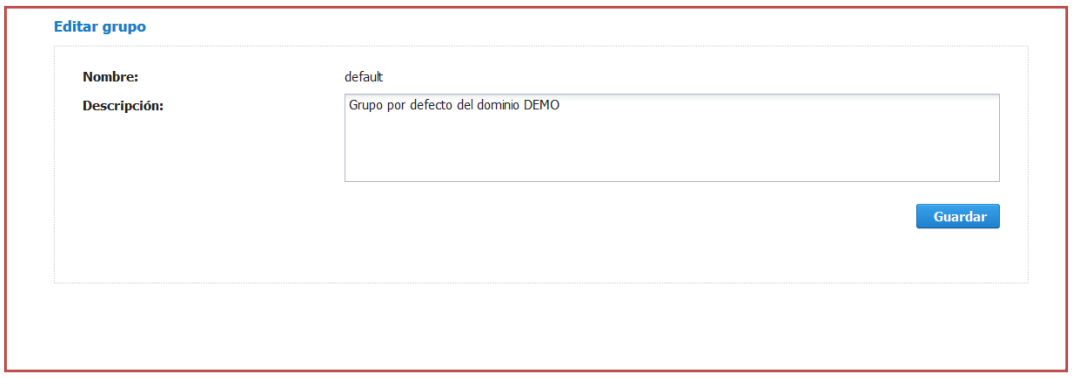


Ilustración 82 Resultado final EditGroupPanel

7. COMPONENTE PROFILEGROUPPANEL

El siguiente componente, se encargará de mostrar los datos de un determinado grupo relacionado con una cuenta de la Plataforma GeoServicios Online.

7.1. DISEÑO INTERFAZ GRÁFICA

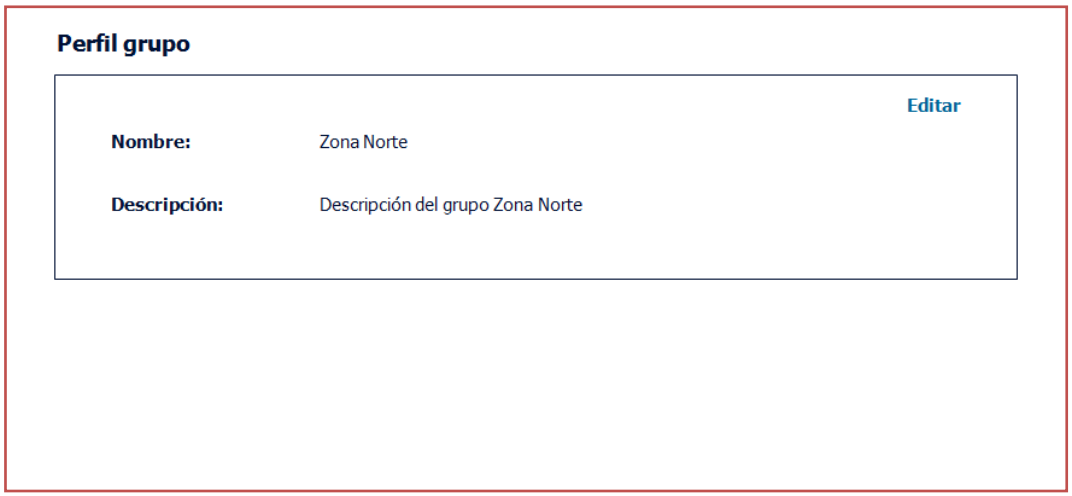


Ilustración 83 Diseño interfaz ProfileGroupPanel

7.2. CASO DE USO

Las funcionalidades identificados en el siguiente componente son las que siguen:

- **Recuperar datos de grupo a editar**
- **Editar grupo**

El diagrama de dicho caso de uso será el siguiente:

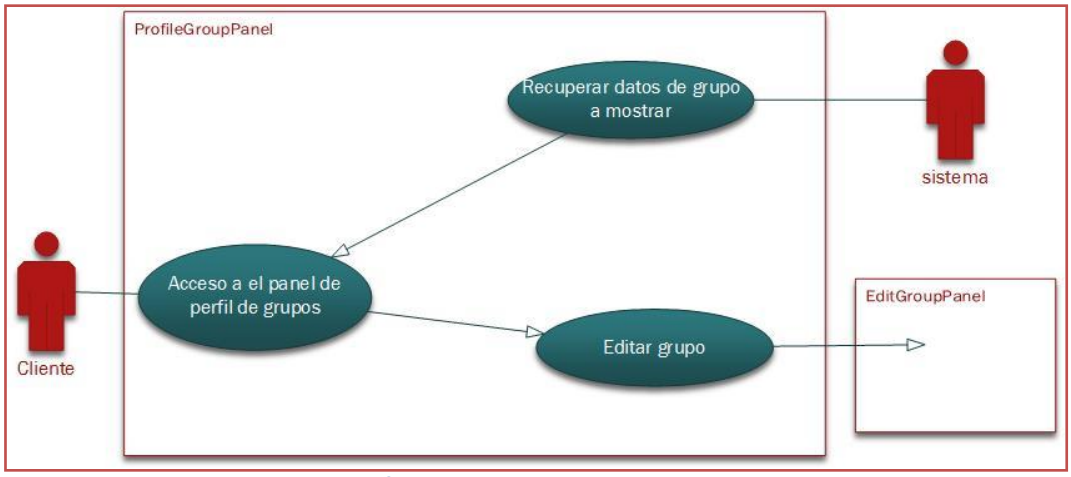


Ilustración 84 Caso de uso ProfileGroupPanel

Recuperar datos de grupo a mostrar: El componente ListGroupPanel se encargará de pasar la información relacionada al grupo seleccionada al componente ProfileGroupPanel. Dicho componente se encargará de mostrar los datos obtenidos.

Editar grupo: Esta funcionalidad , se encargará del acceso al componente EditGroupPanel, el cual se encargará de la edición de grupos.

7.3. DIAGRAMA DE SECUENCIA

Mediante el análisis realizado mediante el caso de uso e identificación de funcionalidades, procederemos a crear el diagrama de secuencia que será el siguiente.

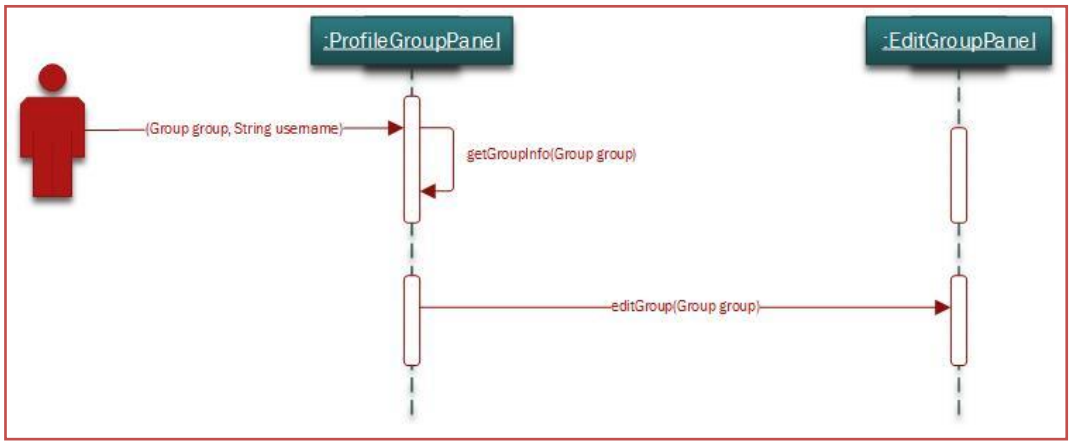


Ilustración 85 Diagrama de secuencia ProfileGroupPanel

7.4. RESULTADO FINAL

El resultado final del componente una vez integrado, es el que se puede observar a continuación:

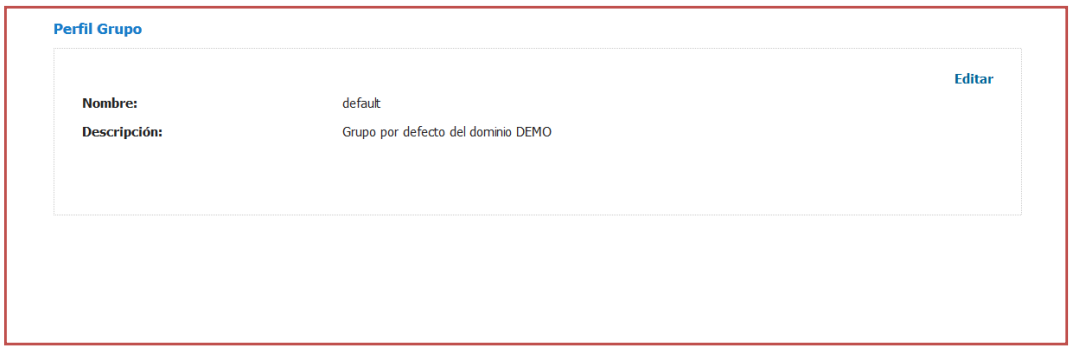


Ilustración 86 Resultado final ProfileGroupPanel

8. COMPONENTE ADDGROUPPANEL

El siguiente componente, se encargará de la creación de un nuevo grupo relacionado con una cuenta de la Plataforma GeoServicios Online.

8.1. DISEÑO INTERFAZ GRÁFICA

Ilustración 87 Diseño interfaz AddGroupPanel

8.2. CASO DE USO

Las funcionalidades identificados en el siguiente componente son las que siguen:

- **Guardar grupo**

El diagrama de dicho caso de uso será el siguiente:

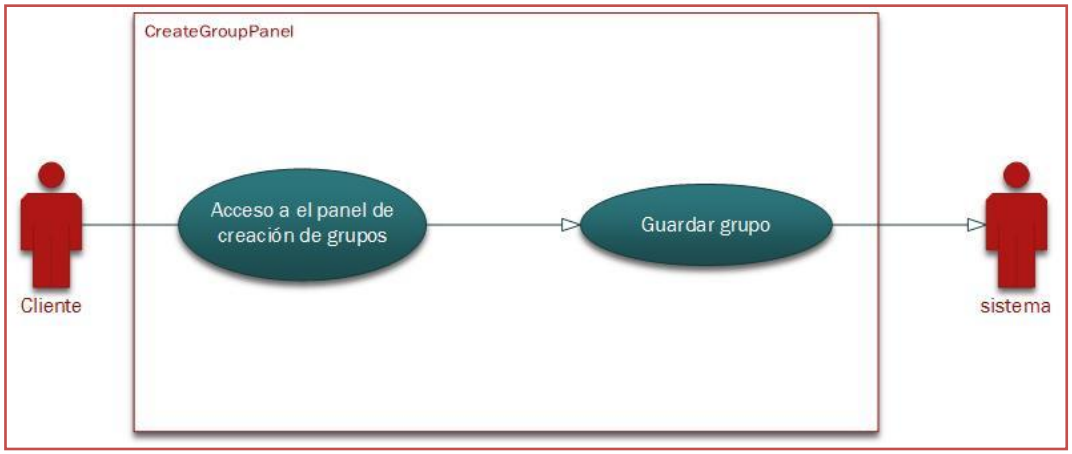


Ilustración 88 Caso de uso CreateGroupPanel

Guardar grupo: Esta funcionalidad, se encargará de almacenar la información del nuevo grupo. Mediante esta funcionalidad identificamos el siguiente servicio:

- createGroup

8.3. DIAGRAMA DE SECUENCIA

Mediante el análisis realizado mediante el caso de uso e identificación de funcionalidades, procederemos a crear el diagrama de secuencia que será el siguiente.

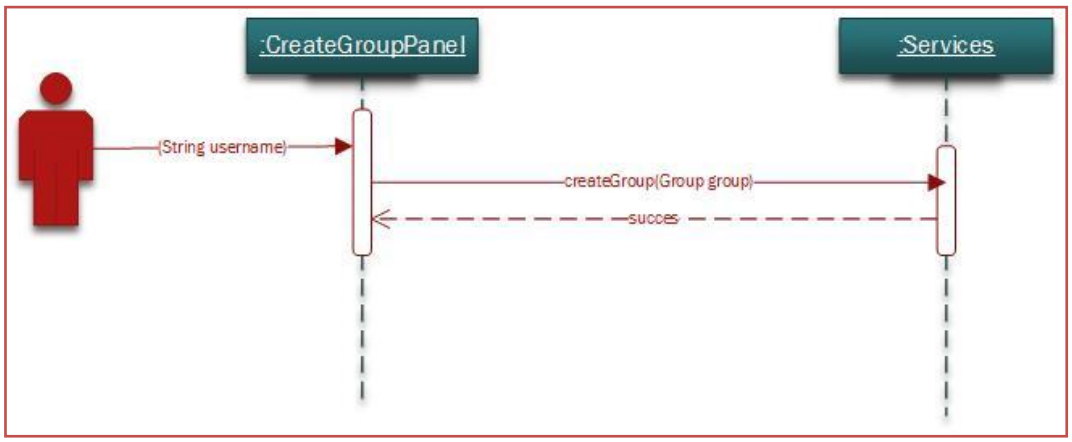


Ilustración 89 Diagrama de secuencia AddGroupPanel

8.4. RESULTADO FINAL

El resultado final del componente una vez integrado, es el que se puede observar a continuación:

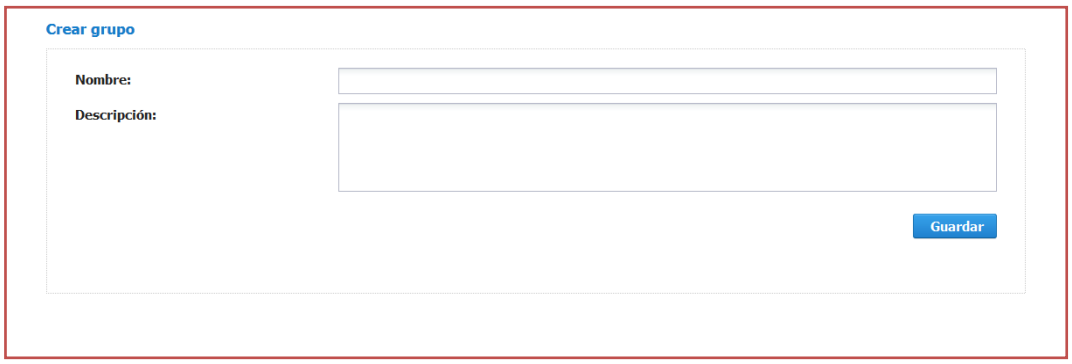


Ilustración 90 Resultado final AddGroupPanel

ANEXO IV: SERVICIOS MÓDULO DE USUARIO Y GRUPOS

El objetivo del presente capítulo es definir los métodos u operaciones que conforman la lógica necesaria del módulo de gestión de usuarios/grupos (User Management Service, de aquí en adelante, UMS) de la Plataforma GeoServicios Online.

La plataforma GeoServicios Online se compone de un conjunto de servicios web que encapsulan la lógica de la misma.

Estos servicios, orientados a ser integrados en los diferentes productos desarrollados por Geograma, pero también abiertos a posibles integraciones de terceros, están preparados para ser consumidos mediante:

- **Servicios web SOAP**, orientados a ser consumidos por aplicaciones de escritorio mediante protocolo SOAP
- **Servlets**, orientados a ser consumidos en formato JSON por aplicaciones web
- **API Javascript**, orientado a ser consumidos por aplicaciones web o móviles con soporte Javascript

1. SERVICIOS WEB SOAP

1.1. GETGROUPS

Devuelve la información de grupos asociados a una determinada cuenta de la Plataforma GeoServicios. Opcionalmente, se pueden indicar una serie de filtros sobre los grupos a obtener, o si se desea obtener sólo un resumen de los grupos o toda su información asociada.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
filter	String	(Opcional) Filtro sobre los grupos a obtener (1)
extendedInfo	Boolean	Indica si se desea obtener también la información de los usuarios asociados a cada grupo

Tabla 18 Parámetros de entrada GetGroups SOAP

(1) El formato del filtro asociado a la petición deberá ser de la siguiente forma (JSON)

```
{
	"filters": [{
		"filterType": "relation",
		"filterBody": {
			"type": "OR",
			"filters": [{
				"filterType": "simple",
				"filterBody": {
					"field": "groupname",
					"operator": "=",
					"type": "string",
					"value": "default"
				}
			},
			{
				"filterType": "relation",
				"filterBody": {
					"type": "AND",
					"filters": [{
						"filterType": "simple",
						"filterBody": {
							"field": "groupname",
							"operator": "=",
							"type": "string",
							"value": "Group1"
						}
					},
					{
						"filterType": "simple",
						"filterBody": {
							"field": "description",
							"operator": "=",
							"type": "string",
							"value": "Grupo nuevo creado para el dominio DEMO"
						}
					}]
				}
			}]
		}
	}]
}
```

Posibles valores del atributo 'type' del objeto 'relation': 'AND', 'OR'
Posibles valores del atributo 'field': "groupname", "description"
Posibles valores del atributo 'operator': "=", "!=", "IN", "NOT IN"

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -16: Error en el filtro enviado

- **totalProperties**: Número de elementos (grupos) devueltos por el servicio

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

- **groups**: (array) Listado de grupos (cero o varios) asociados al dominio y filtros pedidos.
  - name: Nombre del grupo
  - description: Descripción del grupo
  - numUsers: Número de usuarios asociados
  - users: (array) Listado de usuarios (cero o varios) pertenecientes al grupo
    - email: Email del usuario
    - name: Nombre del usuario
    - surname: Apellidos del usuario
    - role: Rol del usuario

1.2. CREATEGROUP

Permite añadir un nuevo grupo asociado a una determinada cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
groupname	String	Nombre del grupo a crear
description	String	(Opcional) Descripción asociada al grupo a crear

Tabla 19 Parámetros de entrada AddGroup SOAP

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -18: Grupo ya existente

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

1.3. MODIFYGROUP

Permite modificar un grupo ya existente asociado a una determinada cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
groupname	String	Nombre del grupo a modificar
description	String	Descripción asociada al grupo a modificar

Tabla 20 Parámetros de entrada ModifyGroup SOAP

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -19: Grupo no existente

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

1.4. DROPGROUP

Permite eliminar un grupo existente asociado a una determinada cuenta de la Plataforma GeoServicios. Adicionalmente, permite eliminar los usuarios asociados a este grupo o asociar estos usuarios al grupo por defecto.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
groupname	String	Nombre del grupo a borrar
cascade	boolean	Indica si los usuarios asociados a este grupo se deben eliminar (true) o no

Tabla 21 Parámetros de entrada DropGroup SOAP

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -19: Grupo no existente

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

1.5. SEARCHGROUP

Permite buscar grupos a partir de una o varias palabras de filtrado asociados a una determinada cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
filter	String	Filtro a buscar en la tabla de grupos

Tabla 22 Parámetros de entrada SearchGroup SOAP

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado

- **totalProperties**: Número de elementos (grupos) devueltos por el servicio

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

- **groups**: (array) Listado de grupos (cero o varios) asociados al filtro enviado.
  - name: Nombre del grupo
  - description: Descripción del grupo
  - numUsers: Número de usuarios asociados

1.6. GETUSERS

Devuelve la información de los usuarios asociados a una determinada cuenta de la Plataforma GeoServicios. Opcionalmente, se pueden indicar una serie de filtros sobre los usuarios a obtener.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
filter	String	(Opcional) Filtro sobre los usuarios a obtener (1)

Tabla 23 Parámetros de entrada GetUsers SOAP

(1) El formato del filtro asociado a la petición deberá ser de la siguiente forma (JSON)

```
{
	"filters": [{
		"filterType": "relation",
		"filterBody": {
			"type": "OR",
			"filters": [{
				"filterType": "simple",
				"filterBody": {
					"field": "groupname",
					"operator": "=",
					"type": "string",
					"value": "default"
				}
			},
			{
				"filterType": "relation",
				"filterBody": {
					"type": "AND",
					"filters": [{
						"filterType": "simple",
						"filterBody": {
							"field": "groupname",
							"operator": "=",
							"type": "string",
							"value": "Group1"
						}
					},
					{
						"filterType": "simple",
						"filterBody": {
							"field": "description",
							"operator": "=",
							"type": "string",
							"value": "Grupo nuevo creado para el dominio DEMO"
						}
					}]
				}
			}]
		}
	}]
}
```

Posibles valores del atributo 'type' del objeto 'relation': 'AND', 'OR'
Posibles valores del atributo 'field': "groupname", "description"
Posibles valores del atributo 'operator': "=", "!=", "IN", "NOT IN"

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -16: Error en el filtro enviado

- **totalProperties**: Número de elementos (usuarios) devueltos por el servicio

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

- **users**: (array) Listado de usuarios (cero o varios) asociados al dominio y filtros pedidos.
  - email: Email del usuario
  - name: Nombre del usuario
  - surname: Apellidos del usuario
  - role: Rol del usuario
  - groupname: Nombre del grupo del usuario

1.7. CREATEUSER

Permite añadir un nuevo usuario asociado a una determinada cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
mail	String	Email del usuario a crear
name	String	(Opcional) Nombre del usuario a crear (por defecto, null)
surname	String	(Opcional) Apellidos del usuario a crear (por defecto, null)
groupname	String	(Opcional) Grupo del usuario a crear (por defecto, 'default')

Parámetro	Tipo de Dato	Descripción
role	String	(Opcional) Rol del usuario a crear (por defecto, 'user'
password	String	Contraseña del usuario a crear

Tabla 24 Parámetros de entrada CreateUser SOAP

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -18: Usuario ya existente y referencias no validas

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

1.8 MODIFYUSER

Permite modificar un usuario ya existente asociado a una determinada cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
mail	String	Email del usuario a modificar
name	String	(Opcional) Nombre del usuario a modificar (por defecto, null)
surname	String	(Opcional) Apellidos del usuario a modificar (por defecto, null)
groupname	String	(Opcional) Grupo del usuario a modificar (por defecto, 'default')
role	String	(Opcional) Rol del usuario a modificar (por defecto, 'user'
password	String	Contraseña del usuario a modificar

Tabla 25 Parámetros de entrada ModifyUser SOAP

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -19: Usuario no existente o grupo no existente

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

1.9 DROPUSER

Permite eliminar un usuario existente asociado a una determinada cuenta de la Plataforma GeoServicios.

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -19: Usuario no existente

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

1.10 SEARCHUSER

Permite buscar usuarios a partir de una o varias palabras de filtrado asociados a una determinada cuenta de la Plataforma GeoServicios

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
filter	String	Filtro a buscar en la tabla de usuarios

Tabla 26 Parámetros de entrada SearchUser SOAP

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado

- **totalProperties**: Número de elementos (usuarios) devueltos por el servicio

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

- **users**: (array) Listado de usuarios (cero o varios) asociados al dominio y filtros pedidos.
  - email: Email del usuario
  - name: Nombre del usuario
  - surname: Apellidos del usuario
  - role: Rol del usuario
  - groupname: Nombre del grupo del usuario

2. SERVICIOS WEB SERVLET

2.1. GETGROUPS

Devuelve la información de grupos asociados a una determinada cuenta de la Plataforma GeoServicios. Opcionalmente, se pueden indicar una serie de filtros sobre los grupos a obtener, o si se desea obtener sólo un resumen de los grupos o toda su información asociada.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
request	String	Tipo de petición. Para este método, el valor de este parámetro deberá ser 'getGroups'
start	Int	(Opcional) Orden del grupo a partir del cual obtener el listado (para paginación)
limit	Int	Límite máximo de resultados a devolver por el servicio (para paginación)
username	String	Nombre de usuario a validar
key	String	Clave asociada al usuario y con permiso para la IP y dominio desde que se envía la petición
callback	String	(Opcional) Función de callback que procesará la petición en el cliente
filter	String	Filtro sobre los grupos a obtener (1)
extendedInfo	Boolean	Indica si se desea obtener también la información de los usuarios asociados a cada grupo

Tabla 27 Parámetros de entrada GetGroups SERVLET

(1) El formato del filtro asociado a la petición deberá ser de la siguiente forma (JSON)

```
{
	"filters": [{
		"filterType": "relation",
		"filterBody": {
			"type": "OR",
			"filters": [{
				"filterType": "simple",
				"filterBody": {
					"field": "groupname",
					"operator": "=",
					"type": "string",
					"value": "default"
				}
			},
			{
				"filterType": "relation",
				"filterBody": {
					"type": "AND",
					"filters": [{
						"filterType": "simple",
						"filterBody": {
							"field": "groupname",
							"operator": "=",
							"type": "string",
							"value": "Group1"
						}
					},
					{
						"filterType": "simple",
						"filterBody": {
							"field": "description",
							"operator": "=",
							"type": "string",
							"value": "Grupo nuevo creado para el dominio DEMO"
						}
					}]
				}
			}]
		}
	}]
}
```

Posibles valores del atributo 'type' del objeto 'relation': 'AND', 'OR'
Posibles valores del atributo 'field': "groupname", "description"
Posibles valores del atributo 'operator': "=", "!=", "IN", "NOT IN"

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o la key indicada no sea válida
  - true: En caso de que todo haya ido bien

- **data**
  - **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
    - -1: Excepción genérica del método
    - -2: Parámetros no válidos
    - -3: Testigo no válido para el usuario y/o producto indicado
    - -6: El parámetro obligatorio 'request' es nulo
    - -7: El parámetro obligatorio 'request' no es válido
    - -16: Error en el filtro enviado
  - **totalProperties**: Número de elementos (grupos) devueltos por el servicio
  - **groups**: (array) Listado de grupos (cero o varios) asociados al dominio y filtros pedidos.
    - name: Nombre del grupo
    - description: Descripción del grupo
    - numUsers: Número de usuarios asociados
    - users: (array) Listado de usuarios (cero o varios) pertenecientes al grupo
      - email: Email del usuario
      - name: Nombre del usuario
      - surname: Apellidos del usuario
      - role: Rol del usuario

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

2.2 CREATEGROUP

Permite añadir un nuevo grupo asociado a una determinada cuenta de la Plataforma GeoServicios.

Parámetro	Tipo de Dato	Descripción
request	String	Tipo de petición. Para este método, el valor de este parámetro deberá ser 'createGroup'
username	String	Nombre de usuario a validar
key	String	Clave asociada al usuario y con permiso para la IP y dominio desde que se envía la petición
groupname	String	Nombre del grupo a crear
description	String	Descripción asociada al grupo a crear

Parámetro	Tipo de Dato	Descripción
callback	String	(Opcional) Función de callback que procesará la petición en el cliente

Tabla 28 Parámetros de entrada CreateGroup SERVLET

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o la key indicada no sea válida
  - true: En caso de que todo haya ido bien

- **data**
  - error: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
    - -1: Excepción genérica del método
    - -2: Parámetros no válidos
    - -3: Testigo no válido para el usuario y/o producto indicado
    - -6: El parámetro obligatorio 'request' es nulo
    - -7: El parámetro obligatorio 'request' no es válido
    - -18: Grupo ya existente

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

2.3 MODIFYGROUP

Permite modificar un grupo ya existente asociado a una determinada cuenta de la Plataforma GeoServicios

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
request	String	Tipo de petición. Para este método, el valor de este parámetro deberá ser 'modifyGroup'
username	String	Nombre de usuario a validar
key	String	Clave asociada al usuario y con permiso para la IP y dominio desde que se envía la petición
groupname	String	Nombre del grupo a modificar
description	String	Descripción asociada al grupo a modificar
callback	String	(Opcional) Función de callback que procesará la petición en el cliente

Tabla 29 Parámetros de entrada ModifyGroup SERVLET

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o la key indicada no sea válida
  - true: En caso de que todo haya ido bien

- **data**
  - error: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
    - -1: Excepción genérica del método
    - -2: Parámetros no válidos
    - -3: Testigo no válido para el usuario y/o producto indicado
    - -6: El parámetro obligatorio 'request' es nulo
    - -7: El parámetro obligatorio 'request' no es válido
    - -19: Grupo no existente

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

2.4 DROPGROUP

Permite eliminar un grupo existente asociado a una determinada cuenta de la Plataforma GeoServicios. Adicionalmente, permite eliminar los usuarios asociados a este grupo o asociar estos usuarios al grupo por defecto.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
request	String	Tipo de petición. Para este método, el valor de este parámetro deberá ser 'modifyGroup'
username	String	Nombre de usuario a validar
key	String	Clave asociada al usuario y con permiso para la IP y dominio desde que se envía la petición
groupname	String	Nombre del grupo a modificar
cascade	Boolean	Indica si los usuarios asociados a este grupo se deben eliminar (true) o se deben asociar al grupo por defecto (false)
callback	String	(Opcional) Función de callback que procesará la petición en el cliente

Tabla 30 Parámetros de entrada DropGroup SERVLET

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o la key indicada no sea válida
  - true: En caso de que todo haya ido bien

- **data**
  - error: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
    - -1: Excepción genérica del método
    - -2: Parámetros no válidos
    - -3: Testigo no válido para el usuario y/o producto indicado
    - -6: El parámetro obligatorio 'request' es nulo
    - -7: El parámetro obligatorio 'request' no es válido
    - -19: Grupo no existente

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

2.5 GETUSERS

Devuelve la información de los usuarios asociados a una determinada cuenta de la Plataforma GeoServicios. Opcionalmente, se pueden indicar una serie de filtros sobre los usuarios a obtener.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
request	String	Tipo de petición. Para este método, el valor de este parámetro deberá ser 'getUsers'
start	Int	(Opcional) Orden del grupo a partir del cual obtener el listado (para paginación)
limit	Int	(Opcional) Límite máximo de resultados a devolver por el servicio (para paginación)
username	String	Nombre de usuario a validar
key	String	Clave asociada al usuario y con permiso para la IP y dominio desde que se envía la petición
callback	String	(Opcional) Función de callback que procesará la petición en el cliente
filter	String	Filtro sobre los usuarios a obtener (1)

Tabla 31 Parámetros de entrada GetUsers SERVLET

(1) El formato del filtro asociado a la petición deberá ser de la siguiente forma (JSON)

```
{
	"filters": [{
		"filterType": "relation",
		"filterBody": {
			"type": "OR",
			"filters": [{
				"filterType": "simple",
				"filterBody": {
					"field": "groupname",
					"operator": "=",
					"type": "string",
					"value": "default"
				}
			},
			{
				"filterType": "relation",
				"filterBody": {
					"type": "AND",
					"filters": [{
						"filterType": "simple",
						"filterBody": {
							"field": "groupname",
							"operator": "=",
							"type": "string",
							"value": "Group1"
						}
					},
					{
						"filterType": "simple",
						"filterBody": {
							"field": "description",
							"operator": "=",
							"type": "string",
							"value": "Grupo nuevo creado para el dominio DEMO"
						}
					}]
				}
			}]
		}
	}]
}
```

Posibles valores del atributo 'type' del objeto 'relation': 'AND', 'OR'
Posibles valores del atributo 'field': "groupname", "description"
Posibles valores del atributo 'operator': "=", "!=", "IN", "NOT IN"

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **data**
  - error: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
    - -1: Excepción genérica del método
    - -2: Parámetros no válidos
    - -3: Testigo no válido para el usuario y/o producto indicado
    - -6: El parámetro obligatorio 'request' es nulo
    - -7: El parámetro obligatorio 'request' no es válido
    - -16: Error en el filtro enviado
  - totalProperties: Número de elementos (usuarios) devueltos por el servicio

- users: (array) Listado de usuarios (cero o varios) asociados al dominio y filtros pedidos.
  - email: Email del usuario
  - name: Nombre del usuario
  - surname: Apellidos del usuario
  - role: Rol del usuario
  - groupname: Nombre del grupo del usuario

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

2.6 CREATEUSER

Permite añadir un nuevo usuario asociado a una determinada cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
request	String	Tipo de petición. Para este método, el valor de este parámetro deberá ser 'createUser'
username	String	Nombre de usuario a validar
key	String	Clave asociada al usuario y con permiso para la IP y dominio desde que se envía la petición
mail	String	Email del usuario a crear
name	String	(Opcional) Nombre del usuario a crear (por defecto, null)
surname	String	(Opcional) Apellidos del usuario a crear (por defecto, null)
groupname	String	(Opcional) Grupo del usuario a crear (por defecto, 'default')
role	String	(Opcional) Rol del usuario a crear (por defecto, 'user'
password	String	Contraseña del usuario a crear
callback	String	(Opcional) Función de callback que procesará la petición en el cliente

Tabla 32 Parámetros de entrada CreateUser SERVLET

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **data**
  - error: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:

  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -6: El parámetro obligatorio 'request' es nulo
  - -7: El parámetro obligatorio 'request' no es válido
  - -18: Usuario ya existente o relación con la tabla de grupos/roles violada

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

2.7 MODIFYUSER

Permite modificar un usuario ya existente asociado a una determinada cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
request	String	Tipo de petición. Para este método, el valor de este parámetro deberá ser 'modifyUser'
username	String	Nombre de usuario a validar
key	String	Clave asociada al usuario y con permiso para la IP y dominio desde que se envía la petición
mail	String	Email del usuario a modificar
name	String	(Opcional) Nombre del usuario a modificar (por defecto, null)
surname	String	(Opcional) Apellidos del usuario a modificar (por defecto, null)
groupname	String	(Opcional) Grupo del usuario a modificar (por defecto, 'default')
role	String	(Opcional) Rol del usuario a modificar (por defecto, 'user'
password	String	Contraseña del usuario a modificar
callback	String	(Opcional) Función de callback que procesará la petición en el cliente

Tabla 33 Parámetros de entrada ModifyUser SERVLET

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -6: El parámetro obligatorio 'request' es nulo
  - -7: El parámetro obligatorio 'request' no es válido
  - -19: Usuario no existente

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

2.8 DROPUSER

Permite eliminar un usuario existente asociado a una determinada cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
request	String	Tipo de petición. Para este método, el valor de este parámetro deberá ser 'dropUser'
username	String	Nombre de usuario a validar
key	String	Clave asociada al usuario y con permiso para la IP y dominio desde que se envía la petición
mail	String	Email del usuario a eliminar
callback	String	(Opcional) Función de callback que procesará la petición en el cliente

Tabla 34 Parámetros de entrada DropUser SERVLET

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -6: El parámetro obligatorio 'request' es nulo
  - -7: El parámetro obligatorio 'request' no es válido
  - -19: Usuario no existente

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

2.9 SEARCHUSERS

Permite buscar usuarios a partir de una o varias palabras de filtrado asociados a una determinada cuenta de la Plataforma GeoServicios

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
request	String	Tipo de petición. Para este método, el valor de este parámetro deberá ser 'searchUsers'
start	Int	(Opcional) Orden del grupo a partir del cual obtener el listado (para paginación)
limit	Int	(Opcional) Límite máximo de resultados a devolver por el servicio (para paginación)
username	String	Nombre de usuario a validar
key	String	Clave asociada al usuario y con permiso para la IP y dominio desde que se envía la petición
filter	String	Filtro a buscar en la tabla de usuarios

Tabla 35 Parámetros de entrada SearchUsers SERVLET

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -6: El parámetro obligatorio 'request' es nulo
  - -7: El parámetro obligatorio 'request' no es válido
  - -16: Error en el filtro enviado

- **totalProperties**: Número de elementos (usuarios) devueltos por el servicio

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

- **users**: (array) Listado de usuarios (cero o varios) asociados al dominio y filtros pedidos.
  - email: Email del usuario
  - name: Nombre del usuario
  - surname: Apellidos del usuario
  - role: Rol del usuario
  - groupname: Nombre del grupo del usuario

2.10 SEARCHGROUPS

Permite buscar grupos a partir de una o varias palabras de filtrado asociados a una determinada cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
request	String	Tipo de petición. Para este método, el valor de este parámetro deberá ser 'searchGroups'
start	Int	(Opcional) Orden del grupo a partir del cual obtener el listado (para paginación)
limit	Int	(Opcional) Límite máximo de resultados a devolver por el servicio (para paginación)
username	String	Nombre de usuario a validar
key	String	Clave asociada al usuario y con permiso para la IP y dominio desde que se envía la petición
filter	String	Filtro a buscar en la tabla de grupos
callback	String	(Opcional) Función de callback que procesará la petición en el cliente

Tabla 36 Parámetros de entrada SearchGroups SERVLET

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o la key indicada no sea válida
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -6: El parámetro obligatorio 'request' es nulo
  - -7: El parámetro obligatorio 'request' no es válido
  - -16: Error en el filtro enviado

- **totalProperties**: Número de elementos (grupos) devueltos por el servicio

- **groups**: (array) Listado de grupos (cero o varios) asociados al dominio y filtros pedidos.
  - name: Nombre del grupo
  - description: Descripción del grupo

3. SERVICIOS WEB JAVASCRIPT

La especificación de los métodos de la capa de servicios Javascript, es análoga a la capa Servlet, con la diferencia de que todos los métodos añadirán el siguiente parámetro:

- method: "GET" o "POST"
- No se introducen los parámetros **start** y **limit** en los métodos **getUsers** y **getGroups**.

Por lo tanto, en este apartado, se expondrá un ejemplo de utilización de cada uno de los métodos para observar cual debe de ser su comportamiento.

3.1 GETGROUPS

Devuelve la información de grupos asociados a una determinada cuenta de la Plataforma GeoServicios. Opcionalmente, se pueden indicar una serie de filtros sobre los grupos a obtener, o si se desea obtener sólo un resumen de los grupos o toda su información asociada.

Ejemplo:

```
//Instantiate the API service - "GEOSERVICIOS.ums"
        GEOSERVICIOS.ums.getGroups({
            username: "theUser",
            key: "theKey",
            filter: "",
            extendedInfo: true,
            callback: thecallback
        });

        function thecallback(response){
            //Do your work here
        }
```

3.2 CREATEGROUP

Permite añadir un nuevo grupo asociado a una determinada cuenta de la Plataforma GeoServicios.

Ejemplo:

```
//Instantiate the API service - "GEOSERVICIOS.ums"
        GEOSERVICIOS.ums.createGroup({
            method: "POST",
            username: "theUser",
            key: "theKey",
            groupName: "DemoGroup",
            description: "DemoGroup description",
            callback: thecallback
        });

        function thecallback(response){
            //Do your work here
        }
```

3.3 MODIFYGROUP

Permite modificar un grupo ya existente asociado a una determinada cuenta de la Plataforma GeoServicios.

Ejemplo:

```
//Instantiate the API service - "GEOSERVICIOS.ums"
        GEOSERVICIOS.ums.modifyGroup({
            method: "POST",
            username: "theUser",
            key: "theKey",
            groupName: "DemoGroup",
            description: "DemoGroup description",
            callback: thecallback
        });

        function thecallback(response){
            //Do your work here
        }
```

3.4 DROPGROUP

Permite eliminar un grupo existente asociado a una determinada cuenta de la Plataforma GeoServicios. Adicionalmente, permite eliminar los usuarios asociados a este grupo o asociar estos usuarios al grupo por defecto.

Ejemplo:

```
//Instantiate the API service - "GEOSERVICIOS.ums"
        GEOSERVICIOS.ums.dropGroup({
            method: "POST",
            username: "theUser",
            key: "theKey",
            groupname: "GroupName",
            cascade: "Cascade",
            callback: thecallback
        });

        function thecallback(response){
            //Do your work here
        }
```

3.5 *GETUSERS*

Devuelve la información de los usuarios asociados a una determinada cuenta de la Plataforma GeoServicios. Opcionalmente, se pueden indicar una serie de filtros sobre los usuarios a obtener.

Ejemplo:

```
//Instantiate the API service - "GEOSERVICIOS.ums"
         GEOSERVICIOS.ums.getUsers({
            method: "POST",
            username: "theUser",
            key: "theKey",
            filter: "",
            extendedInfo: false,
            callback: thecallback
         });

          function thecallback(response){
            //Do your work here
          }
```

3.6 *CREATEUSER*

Permite añadir un nuevo usuario asociado a una determinada cuenta de la Plataforma GeoServicios.

Ejemplo:

```
//Instantiate the API service - "GEOSERVICIOS.ums"
        GEOSERVICIOS.ums.createUser({
            method: "POST",
            username: "theUser",
            key: "theKey",
            filter: "",
            mail: "Mail",
            name: "Name",
            surname: "Surname",
            role: "Role",
            password: "Password",
            callback: thecallback
         });

         function thecallback(response){
            //Do your work here
         }
```

3.7 *MODIFYUSER*

Permite modificar un usuario ya existente asociado a una determinada cuenta de la Plataforma GeoServicios.

Ejemplo:

```
//Instantiate the API service - "GEOSERVICIOS.ums"
        GEOSERVICIOS.ums.modifyUser({
            method: "POST",
            username: "theUser",
            key: "theKey",
            mail: "Mail",
            name: "Name",
            surname: "Surname",
            role: "Role",
            password: "Password",
            callback: thecallback
        });

        function thecallback(response){
            //Do your work here
        }
```

3.8 DROPUSER

Permite eliminar un usuario existente asociado a una determinada cuenta de la Plataforma GeoServicios.

Ejemplo:

```
//Instantiate the API service - "GEOSERVICIOS.ums"
        GEOSERVICIOS.ums.dropUser({
            method: "POST",
            username: "theUser",
            key: "theKey",
            mail: "Mail",
            callback: thecallback
        });

        function thecallback(response){
            //Do your work here
        }
```

3.9 SEARCHUSERS

Permite buscar usuarios a partir de una o varias palabras de filtrado asociados a una determinada cuenta de la Plataforma GeoServicios.

Ejemplo:

```
//Instantiate the API service - "GEOSERVICIOS.ums"
        GEOSERVICIOS.ums.searchUsers({
            method: "POST",
            username: "theUser",
            key: "theKey",
            filter: "theFilter",
            callback: thecallback
        });

        function thecallback(response){
            //Do your work here
        }
```

3.10 SEARCHGROUPS

Permite buscar grupos a partir de una o varias palabras de filtrado asociados a una determinada cuenta de la Plataforma GeoServicios.

Ejemplo:

```
//Instantiate the API service - "GEOSERVICIOS.ums"

        GEOSERVICIOS.ums.searchUsers({
            method: "POST",
            username: "theUser",
            key: "theKey",
            filter: "theFilter",
            callback: thecallback
        });

        function thecallback(response){
            //Do your work here
```

ANEXO V: COMPONENTES MÓDULO DE DATOS

En el siguiente Anexo, se detalla el diseño y especificación de cada uno de los componentes del módulo de datos. En dicho anexo, se detalla tanto la interfaz gráfica, los casos de uso, como los diagramas de secuencia de cada uno de los componentes.

Los componentes del módulo datos que se especifican son los siguientes:

Componentes relativos a los almacenes de datos:

- WharehouseListDataPanel
- WharehouseEditDataPanel
- WharehouseConfigDataPanel

Componentes relativos a las plantillas de datos:

- TemplateListDataPanel
- TemplateEditDataPanel
- TemplateCreateDataPanel

Cada uno de los componentes estará asociado a ciertos servicios que se irán identificando al realizar los casos de uso pertinentes. Dichos servicios están especificados en el Anexo VI: Servicios módulo de datos.

1.COMPONENTE WAREHOUSELISTDATAPANEL

El siguiente componente, se encargará de mostrar el listado de almacenes de datos relacionados con una determinada cuenta de la Plataforma GeoServicios Online, y la funcionalidad de búsqueda en dicho listado. Por otro lado también se encarga del borrado, y edición de almacenes de datos.

1.1 DISEÑO INTERFAZ GRÁFICA

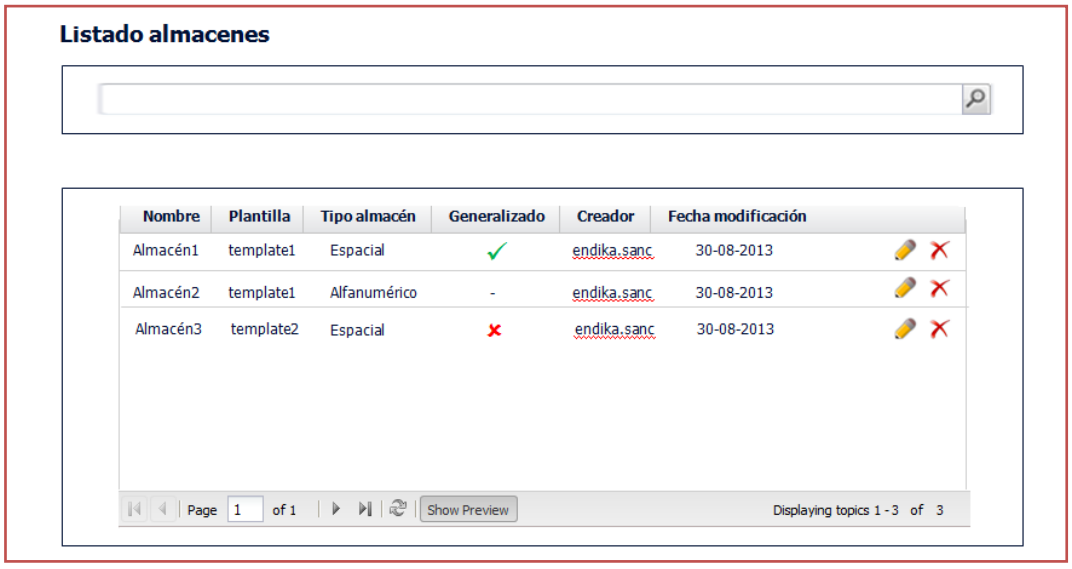


Ilustración 91 Diseño interfaz WarehouseListDataPanel

1.2 CASO DE USO

Las funcionalidades identificados en el siguiente componente son las que siguen:

- **Pedir lista de almacenes**
- **Borrar almacén**
- **Buscar almacén**
- **Editar almacén**

El diagrama de dicho caso de uso será el siguiente:

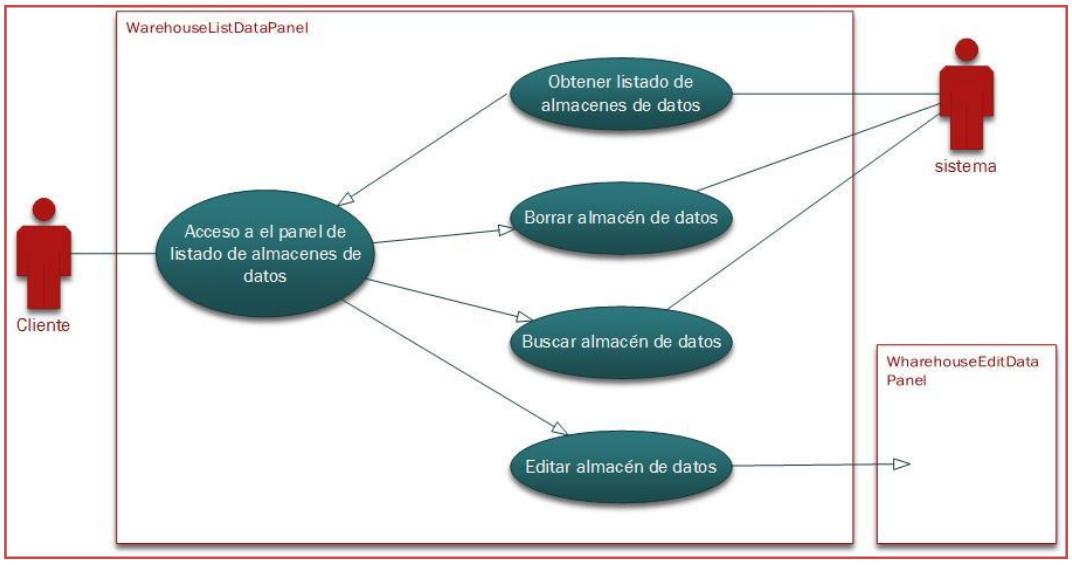


Ilustración 92 Caso de uso WarehouseListDataPanel

Pedir lista de almacenes: El sistema se encargará de devolver una lista de almacenes de datos y mostrarla en un listado. Mediante esta funcionalidad identificamos el siguiente servicio:

- getDataStores

Borrar almacén: Esta funcionalidad, se encargará del borrado de almacenes de datos. Mediante esta funcionalidad identificamos el siguiente servicio:

- dropDataStore

Buscar almacén: Esta funcionalidad, se encargará de la búsqueda de almacenes de datos dentro del listado devuelto por el sistema. Mediante esta funcionalidad identificamos el siguiente servicio:

- searchDataStores

Editar almacén: Esta funcionalidad , se encargará del acceso al componente WarehouseEditDataPanel, el cual se encargará de la edición de almacenes de datos.

1.3 DIAGRAMA DE SECUENCIA

Mediante el análisis realizado mediante el caso de uso e identificación de funcionalidades, procederemos a crear el diagrama de secuencia que será el siguiente.

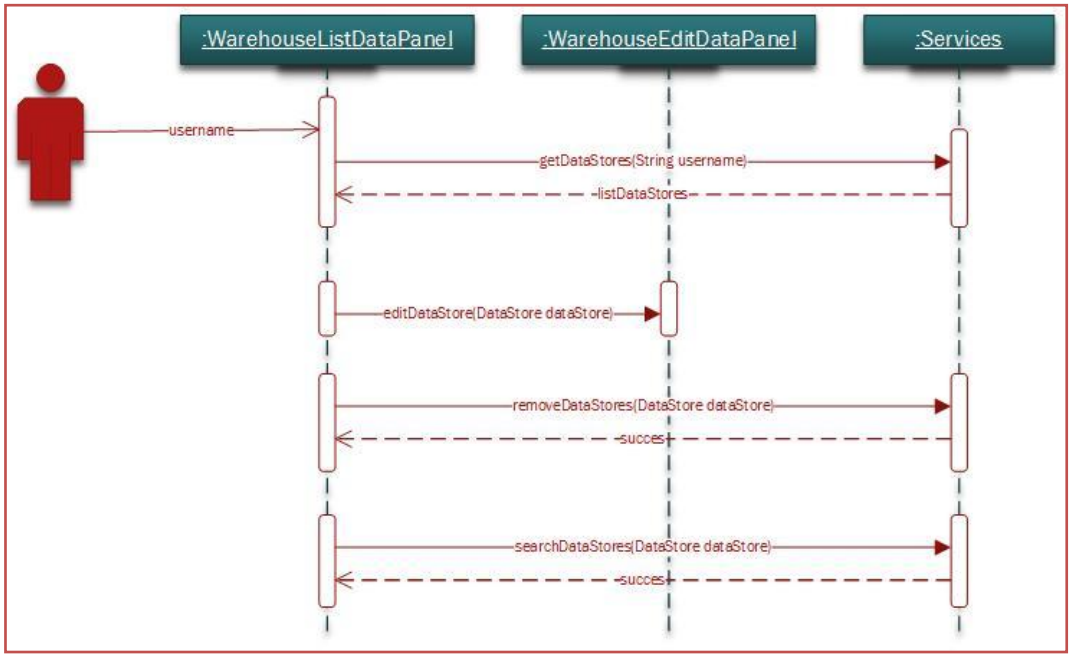


Ilustración 93 Diagrama de secuencia WarehouseListDataPanel

1.4 RESULTADO FINAL

El resultado final del componente una vez integrado, es el que se puede observar a continuación:

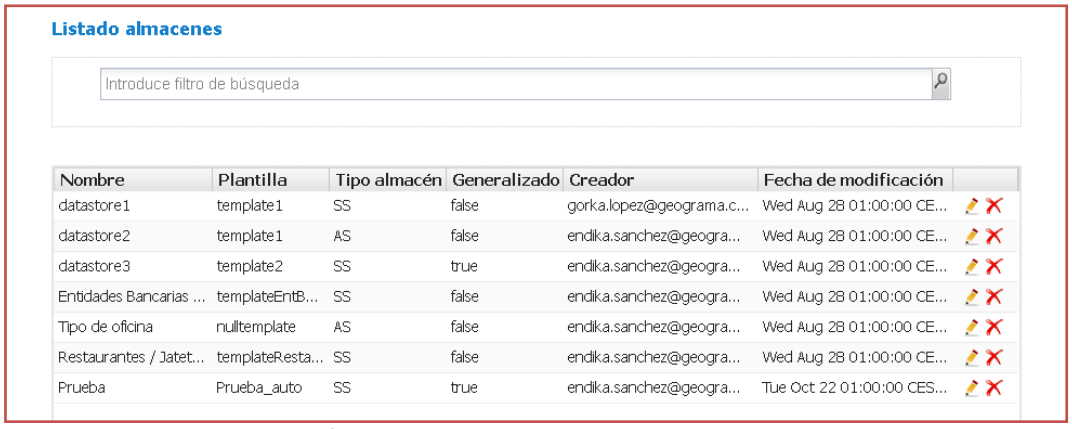


Ilustración 94 Resultado final WarehouseListDataPanel

2.COMPONENTE WAREHOUSEEDITDATAPANEL

El siguiente componente, se encargará de editar un determinado almacén de datos relacionados con una determinada cuenta de la Plataforma GeoServicios Online.

2.1 DISEÑO INTERFAZ GRÁFICA

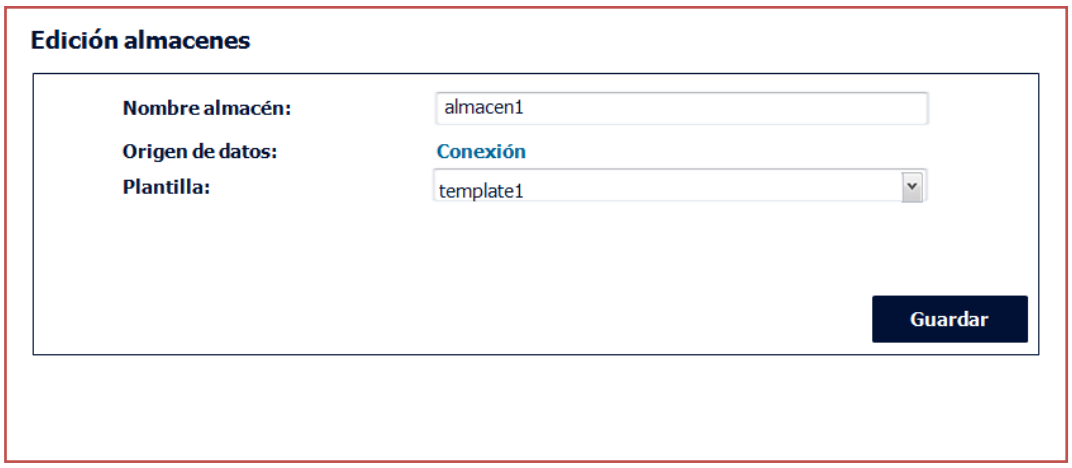


Ilustración 95 Diseño interfaz WarehouseEditDataPanel

2.2 CASO DE USO

Las funcionalidades identificados en el siguiente componente son las que siguen:

- **Recuperar datos de almacén de datos a editar**
- **Obtener plantillas disponibles**
- **Editar conexión de origen de datos**
- **Guardar almacén**

El diagrama de dicho caso de uso será el siguiente:

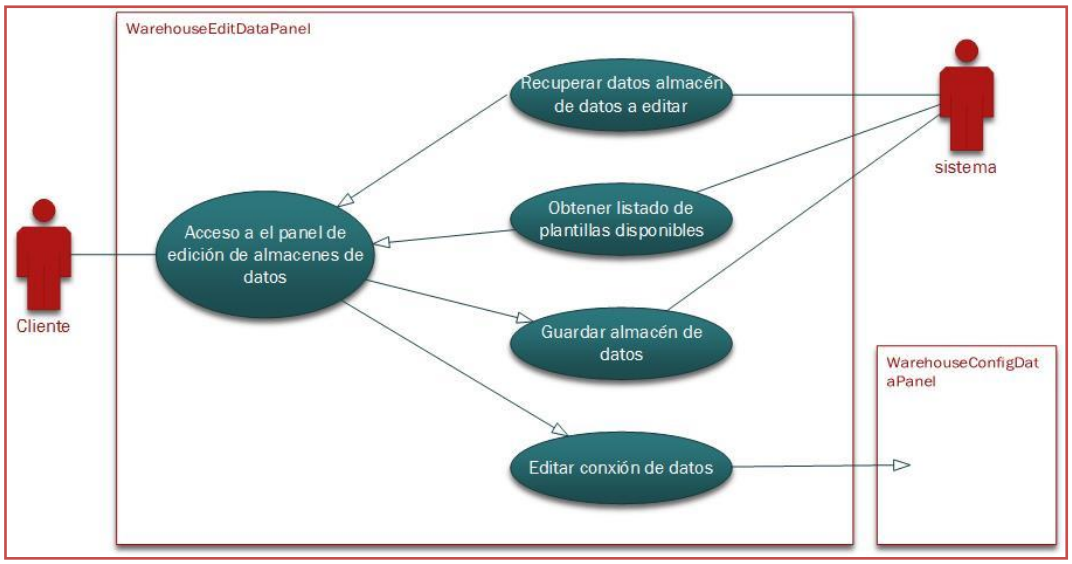


Ilustración 96 Caso de uso WarehouseEditDataPanel

Recuperar datos de almacén de datos a editar: El componente WarehouseListDataPanel se encargará de pasar la información relacionada el almacén de datos seleccionada al componente WarehouseEditDataPanel- Dicho componente se encargará de mostrar los datos obtenidos.

Obtener plantillas disponibles: Esta funcionalidad, se encargará de obtener un listado de las plantillas de datos disponibles. Mediante esta funcionalidad identificamos el siguiente servicio:

- getDataTemplate

Editar conexión origen de datos: Esta funcionalidad, se encargará de dar acceso al componente WarehouseConfigDataPanel, que se encargará de la edición de la conexión de origen de datos relacionada con el almacén de datos seleccionado.

Guardar almacén de datos: Esta funcionalidad , se encargará almacenar el almacén de datos seleccionado. Mediante esta funcionalidad, identificamos el siguiente servicio:

- modifyDataStore

2.3 DIAGRAMA DE SECUENCIA

Mediante el análisis realizado mediante el caso de uso e identificación de funcionalidades, procederemos a crear el diagrama de secuencia que será el siguiente.

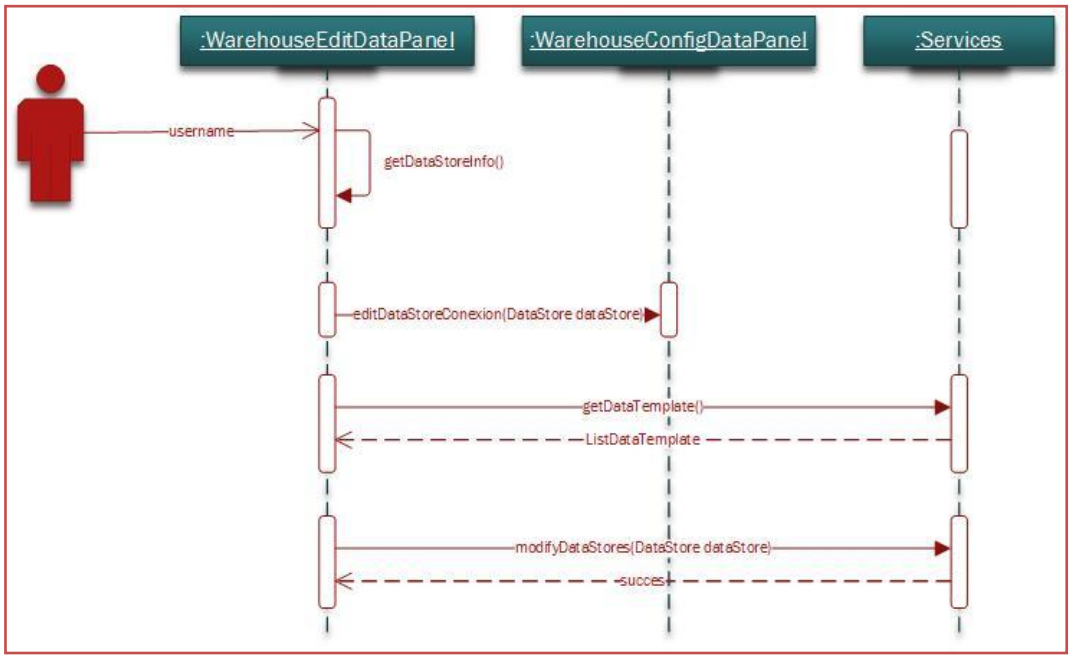


Ilustración 97 Diagrama de secuencia WarehouseEditDataPanel

2.4 RESULTADO FINAL

El resultado final del componente una vez integrado, es el que se puede observar a continuación:

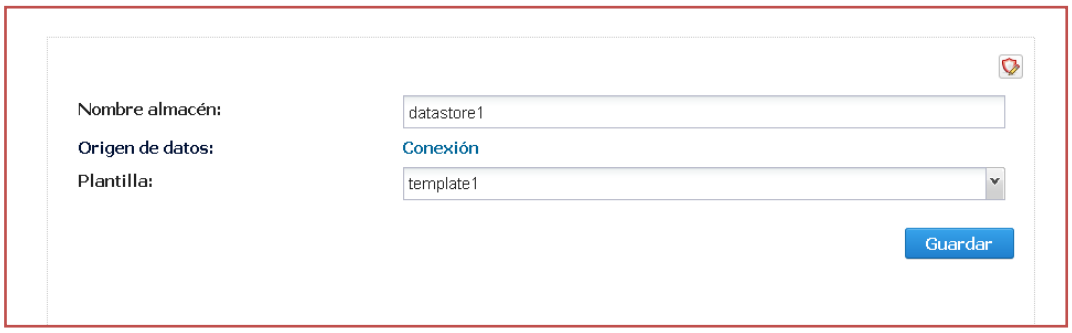


Ilustración 98 Resultado final WarehouseEditDataPanel

3.COMPONENTE WAREHOUSECONFIGDATAPANEL

El siguiente componente, se encargará de configurar una determinada conexión de datos asociada a un determinado almacén de datos. De esta manera crearemos los almacenes de datos. Dicho almacén de datos, estará relacionado con una determinada cuenta de la Plataforma GeoServicios Online.

3.1 DISEÑO INTERFAZ GRÁFICA

Ilustración 99 Diseño interfaz WarehouseConfigDataPanel

3.2 CASO DE USO

Las funcionalidades identificados en el siguiente componente son las que siguen:

- **Obtener tablas de una conexión de datos**
- **Obtener campos de una determinada tabla**
- **Crear almacén de datos**

El diagrama de dicho caso de uso será el siguiente:

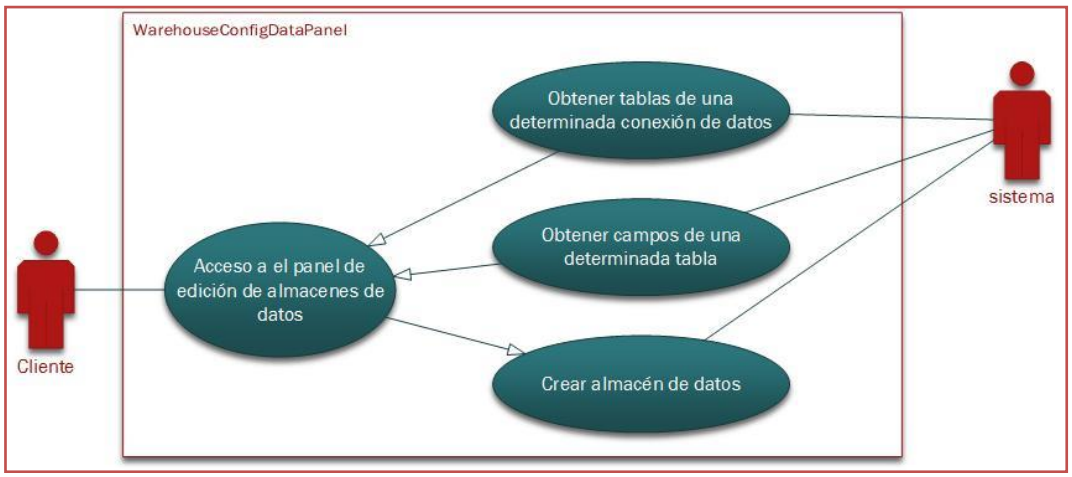


Ilustración 100 Caso de uso WarehouseConfigDataPanel

Obtener tablas de una determinada conexión de datos: Esta funcionalidad se encargará de obtener un listado de las tablas que contenga una determinada conexión de datos introducida por el usuario Mediante esta funcionalidad identificamos el siguiente servicio:

- getTablesFromConn

Obtener campos de una determinada tabla: Esta funcionalidad, se encargará de obtener un listado de los campos disponibles de una determinada tabla. Mediante esta funcionalidad identificamos el siguiente servicio:

- getFieldsFromConn

Crear almacén de datos: Esta funcionalidad , se encargará de almacenar el almacén de datos creado. Mediante esta funcionalidad, identificamos el siguiente servicio:

- createDataStore

3.3 DIAGRAMA DE SECUENCIA

Mediante el análisis realizado mediante el caso de uso e identificación de funcionalidades, procederemos a crear el diagrama de secuencia que será el siguiente.

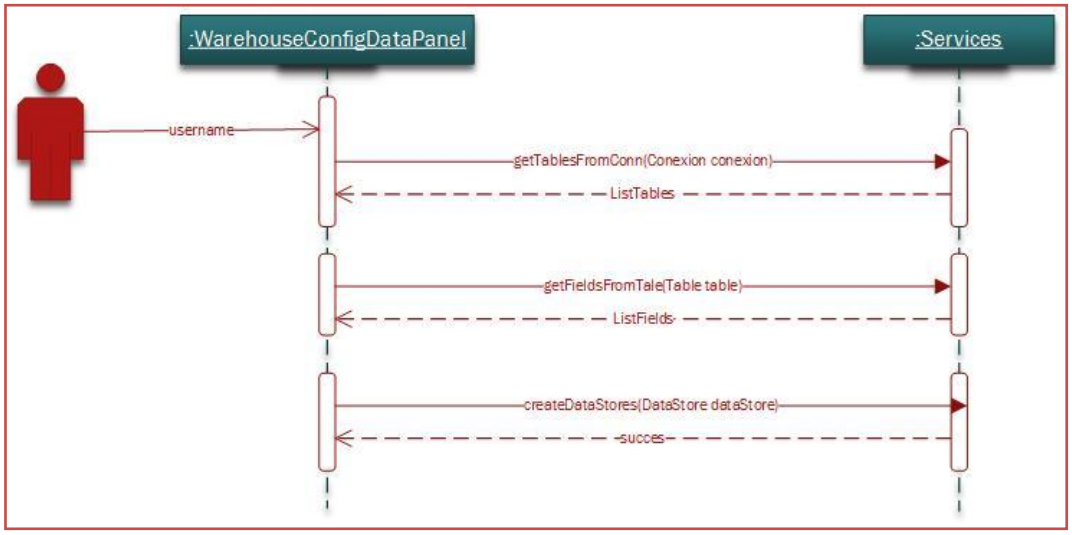


Ilustración 101 Diagrama de secuencia WarehouseConfigDataPanel

3.4 RESULTADO FINAL

El resultado final del componente una vez integrado, es el que se puede observar a continuación:

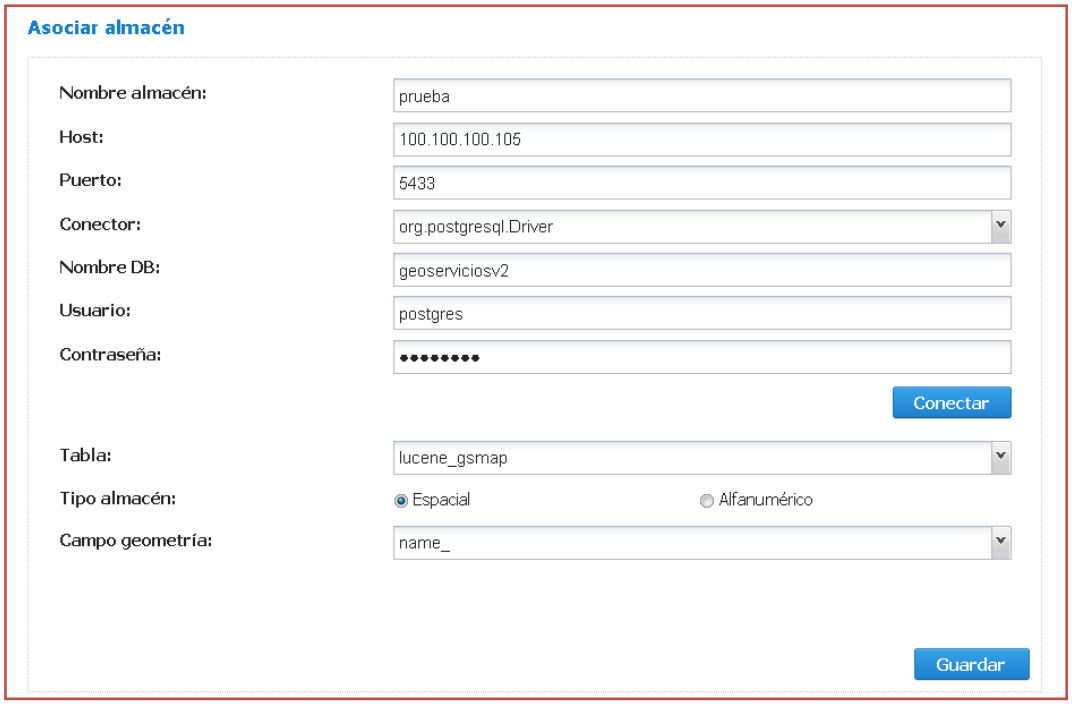


Ilustración 102 Resultado final WarehouseConfigDataPanel

4.COMPONENTE TEMPLATELISTDATAPANEL

El siguiente componente, se encargará de mostrar el listado de plantillas de datos relacionados con una determinada cuenta de la Plataforma GeoServicios Online, y la funcionalidad de búsqueda en dicho listado. Por otro lado también se encarga del borrado, y edición de plantillas de datos.

4.1 DISEÑO INTERFAZ GRÁFICA

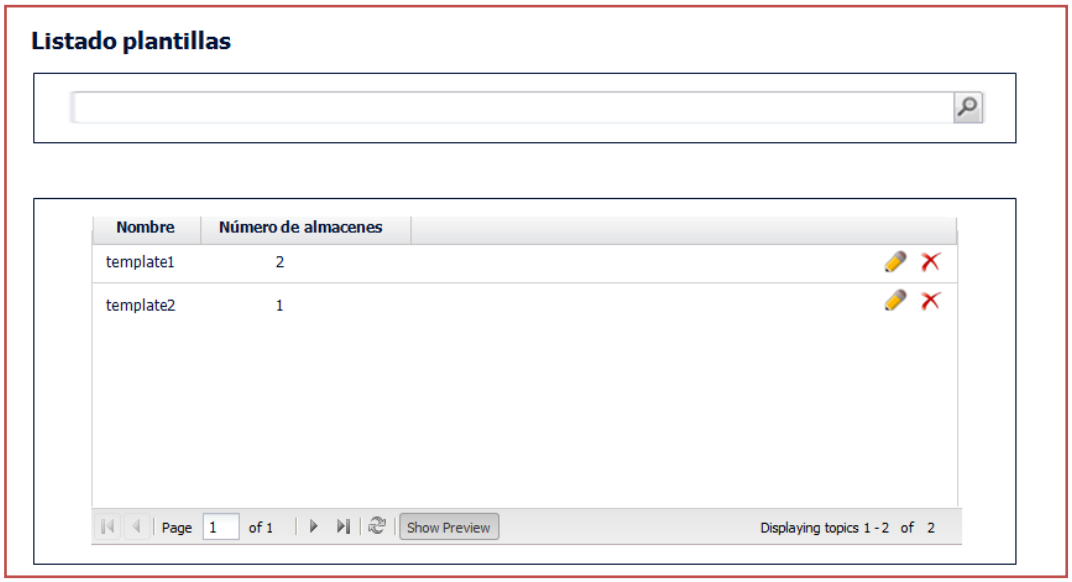


Ilustración 103 Diseño interfaz TemplateListDataPanel

4.2 CASO DE USO

Las funcionalidades identificados en el siguiente componente son las que siguen:

- **Pedir lista de plantillas de datos**
- **Borrar plantilla de datos**
- **Buscar plantilla de datos**
- **Editar plantilla de datos**

El diagrama de dicho caso de uso será el siguiente:

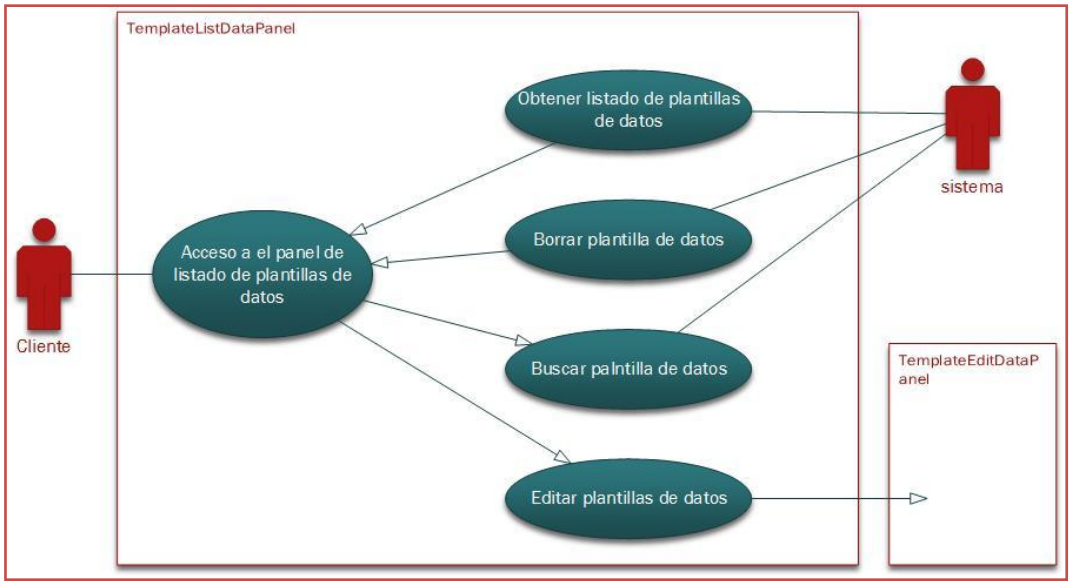


Ilustración 104 Caso de uso TemplateListDataPanel

Pedir lista de plantillas de datos: El sistema se encargará de devolver una lista de plantillas de datos y mostrarla en un listado. Mediante esta funcionalidad identificamos el siguiente servicio:

- getDataTemplate

Borrar plantilla de datos: Esta funcionalidad, se encargará del borrado de plantillas de datos. Mediante esta funcionalidad identificamos el siguiente servicio:

- dropDataTemplate

Buscar plantilla de datos: Esta funcionalidad, se encargará de la búsqueda de plantillas de datos dentro del listado devuelto por el sistema. Mediante esta funcionalidad identificamos el siguiente servicio:

- searchDataTemplate

Editar plantilla de datos: Esta funcionalidad , se encargará del acceso al componente TemplateEditDataPanel, el cual se encargará de la edición de plantillas de datos.

4.3 DIAGRAMA DE SECUENCIA

Mediante el análisis realizado mediante el caso de uso e identificación de funcionalidades, procederemos a crear el diagrama de secuencia que será el siguiente.

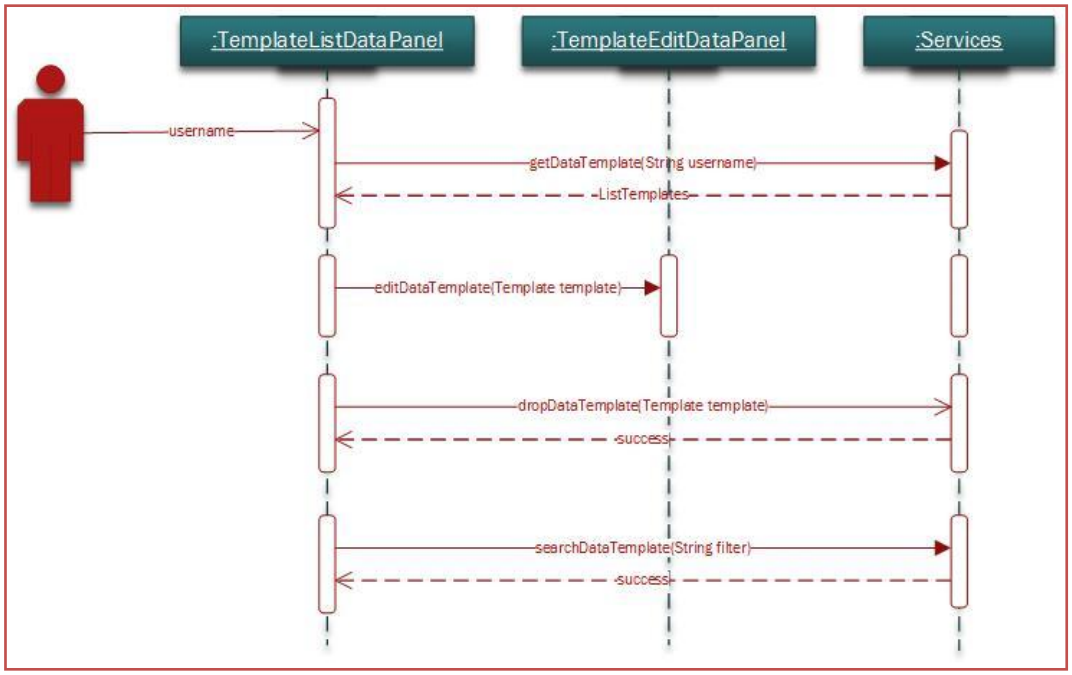


Ilustración 105 Diagrama de secuencia TemplateListDataPanel

4.4 RESULTADO FINAL

El resultado final del componente una vez integrado, es el que se puede observar a continuación:

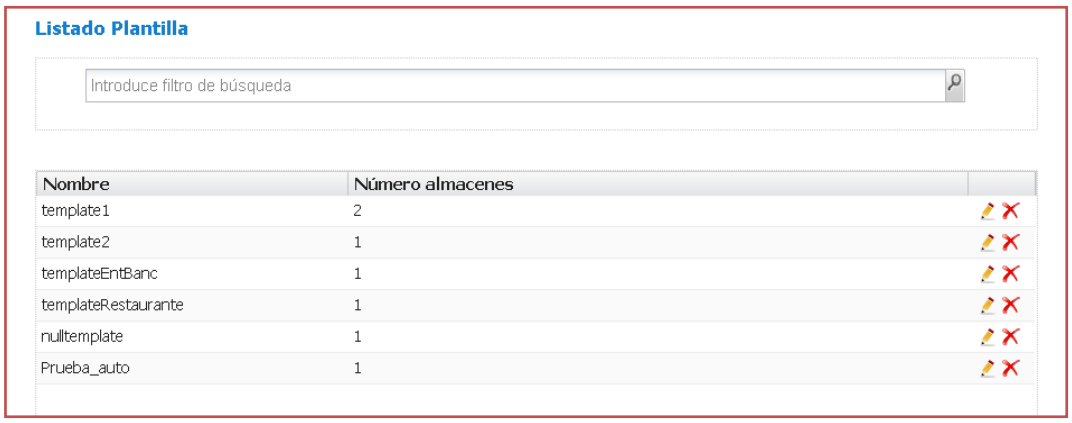


Ilustración 106 Resultado final TemplateListDataPanel

5.COMPONENTE TEMPLATEEDITDATAPANEL

El siguiente componente, se encargará de editar una determinada plantilla de datos relacionada con una determinada cuenta de la Plataforma GeoServicios Online.

5.1 DISEÑO INTERFAZ GRÁFICA

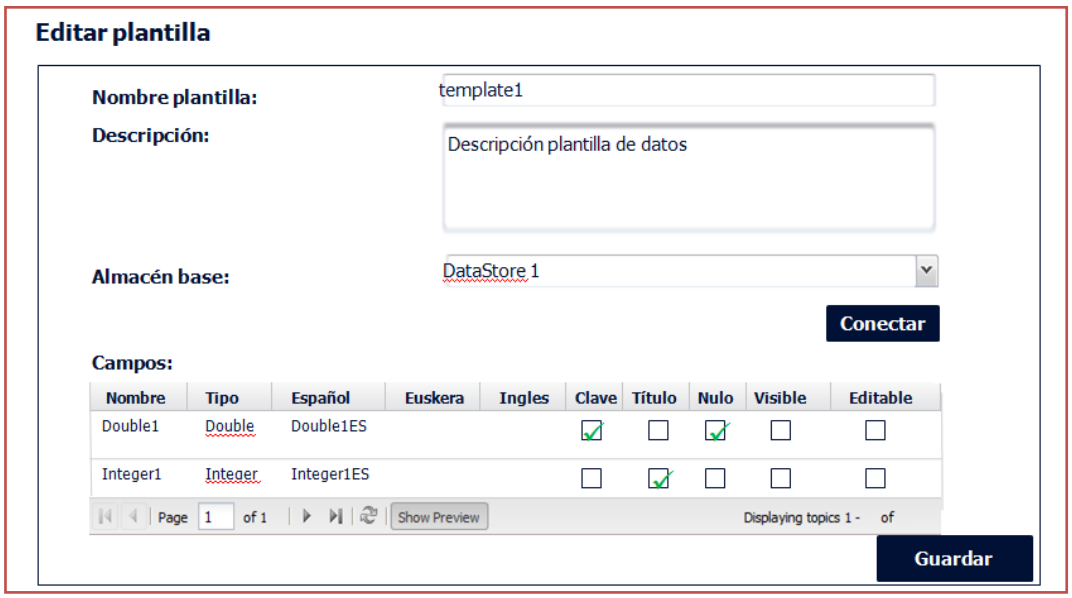


Ilustración 107 Diseño interfaz TemplateEditDataPanel

5.2 CASO DE USO

Las funcionalidades identificados en el siguiente componente son las que siguen:

- **Recuperar datos plantilla de datos a editar**
- **Obtener almacenes de datos disponibles**
- **Obtener campos almacén de datos seleccionado**
- **Guardar plantilla de datos**

El diagrama de dicho caso de uso será el siguiente:

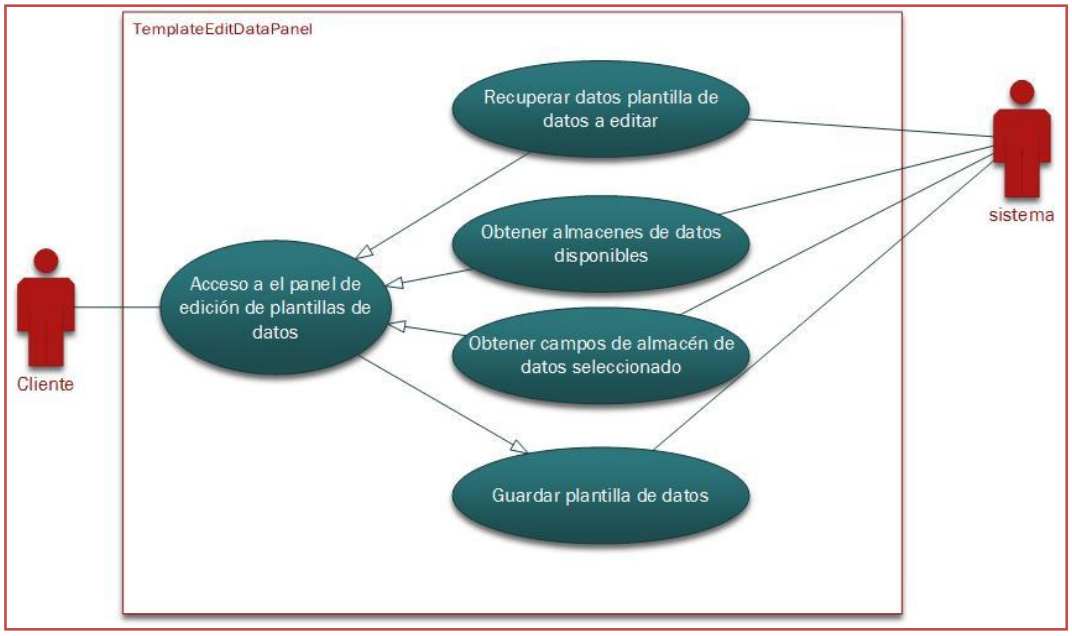


Ilustración 108 Caso de uso TemplateEditDataPanel

Recuperar datos plantilla de datos a editar: El componente TemplateListDataPanel se encargará de pasar la información relacionada la plantilla de datos seleccionada al componente TemplateEditDataPanel. Dicho componente se encargará de mostrar los datos obtenidos.

Obtener almacenes de datos disponibles: Esta funcionalidad, se encargará de obtener el listado de almacenes de datos disponibles. Mediante esta funcionalidad identificamos el siguiente servicio:

- getDataStore

Obtener campos de almacén de datos seleccionado: Esta funcionalidad, se encargará obtener los campos de la conexión de datos relacionada con el almacén de datos seleccionado. Mediante esta funcionalidad identificamos el siguiente servicio:

- getFieldsFromDataStoreConn

Guardar plantilla de datos: Esta funcionalidad , se encargará de guardar la plantilla de datos. Mediante esta funcionalidad identificamos el siguiente servicio:

- modifyDataTemplate

5.3 DIAGRAMA DE SECUENCIA

Mediante el análisis realizado mediante el caso de uso e identificación de funcionalidades, procederemos a crear el diagrama de secuencia que será el siguiente.

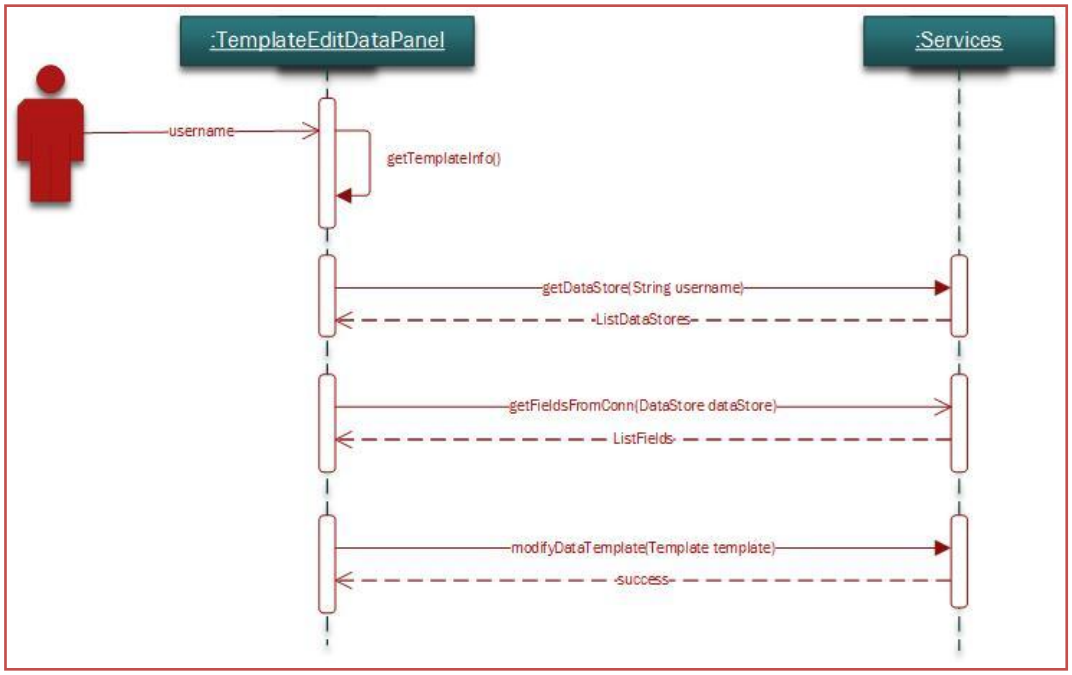


Ilustración 109 Diagrama de secuencia TemplateEditDataPanel

5.4 RESULTADO FINAL

El resultado final del componente una vez integrado, es el que se puede observar a continuación:

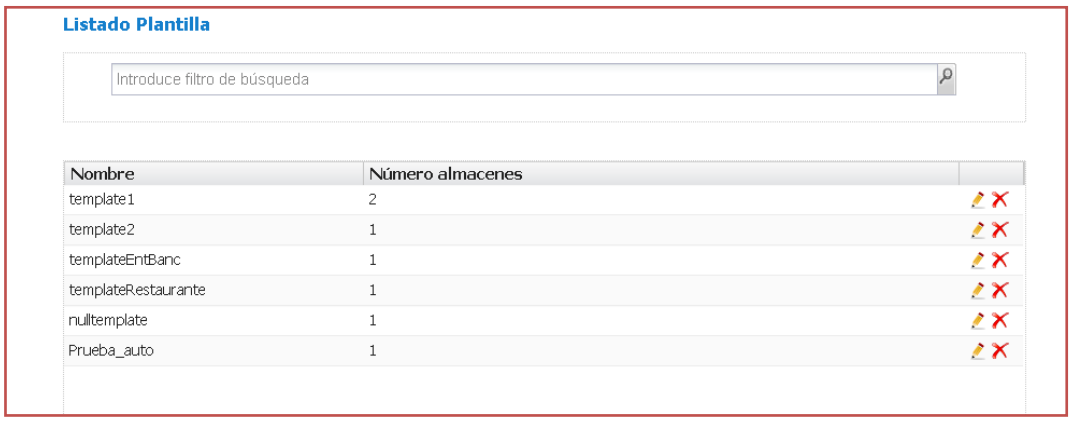


Ilustración 110 Resultado final TemplateEditDataPanel

6.COMPONENTE TEMPLATECREATEDATAPANEL

El siguiente componente, se encargará de crear una determinada plantilla de datos relacionada con una determinada cuenta de la Plataforma GeoServicios Online.

6.1 DISEÑO INTERFAZ GRÁFICA

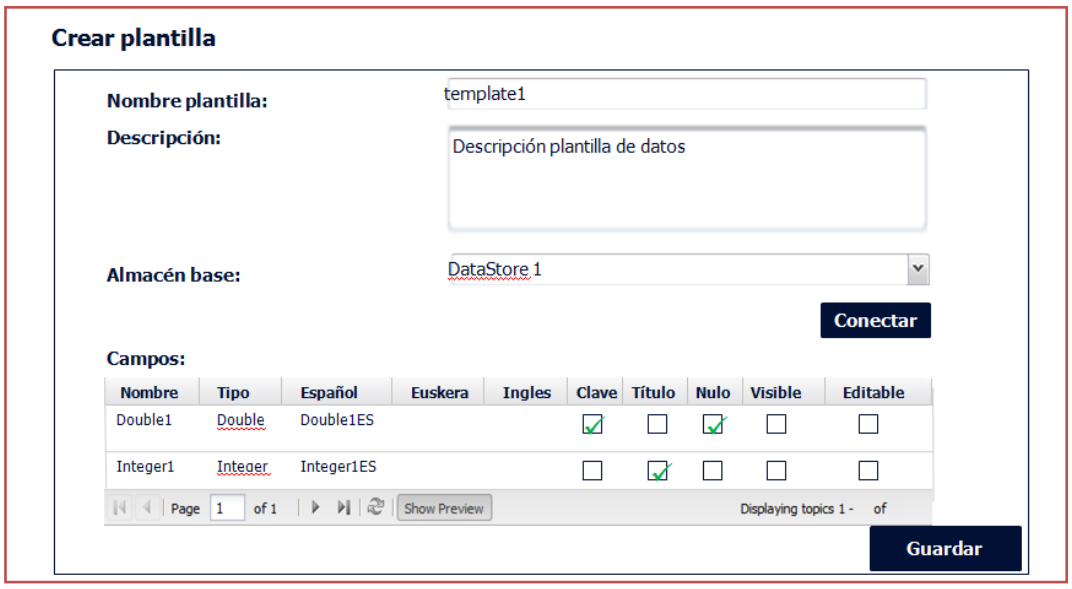


Ilustración 111 Diseño interfaz TemplateCreateDataPanel

6.2 CASO DE USO

Las funcionalidades identificados en el siguiente componente son las que siguen:

- **Obtener almacenes de datos disponibles**
- **Obtener campos almacén de datos seleccionado**
- **Crear plantilla de datos**

El diagrama de dicho caso de uso será el siguiente:

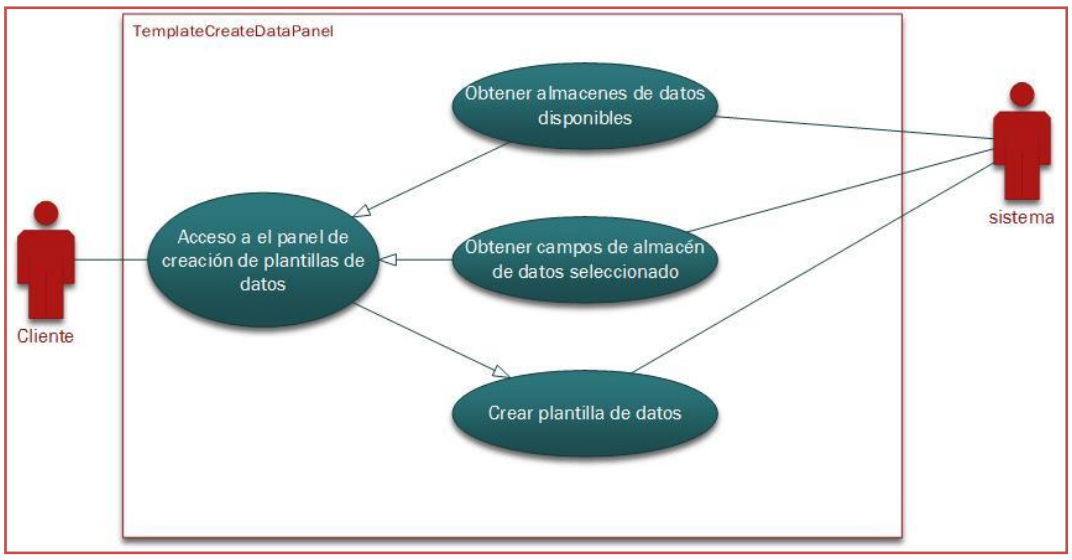


Ilustración 112 Caso de uso TemplateCreateDataPanel

Obtener almacenes de datos disponibles: Esta funcionalidad, se encargará de obtener el listado de almacenes de datos disponibles. Mediante esta funcionalidad identificamos el siguiente servicio:

- getDataStore

Obtener campos de almacén de datos seleccionado: Esta funcionalidad, se encargará obtener los campos relacionada con el almacén de datos seleccionado. Mediante esta funcionalidad identificamos el siguiente servicio:

- getFieldsFromConn

Crear plantilla de datos: Esta funcionalidad , se encargará de crear la plantilla de datos. Mediante esta funcionalidad identificamos el siguiente servicio:

- createDataTemplate

6.3 DIAGRAMA DE SECUENCIA

Mediante el análisis realizado mediante el caso de uso e identificación de funcionalidades, procederemos a crear el diagrama de secuencia que será el siguiente.

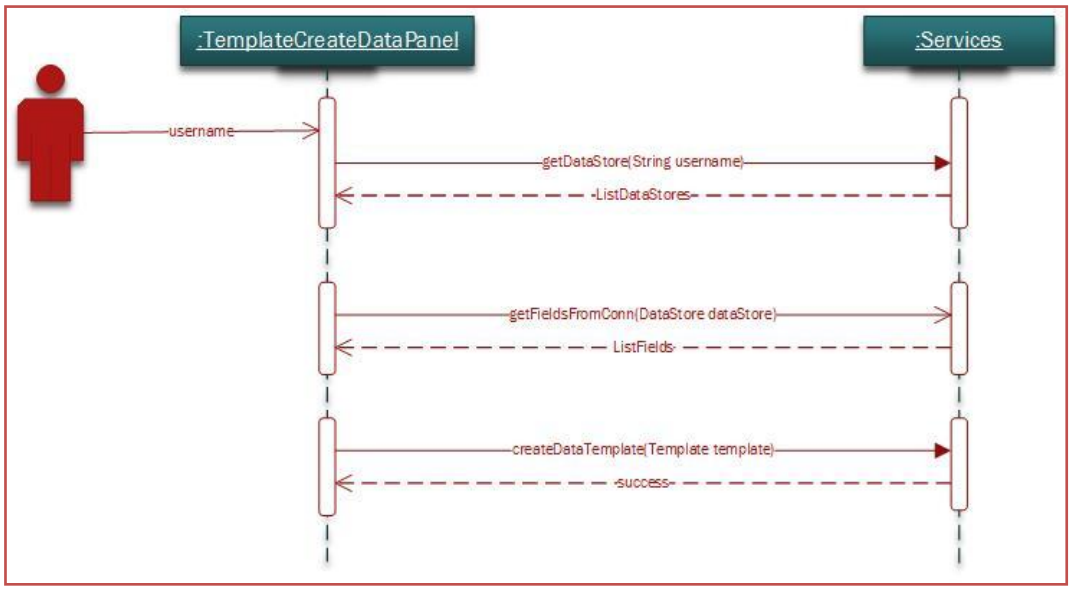


Ilustración 113 Diagrama de secuencia TemplateCreateDataPanel

6.4 RESULTADO FINAL

El resultado final del componente una vez integrado, es el que se puede observar a continuación:

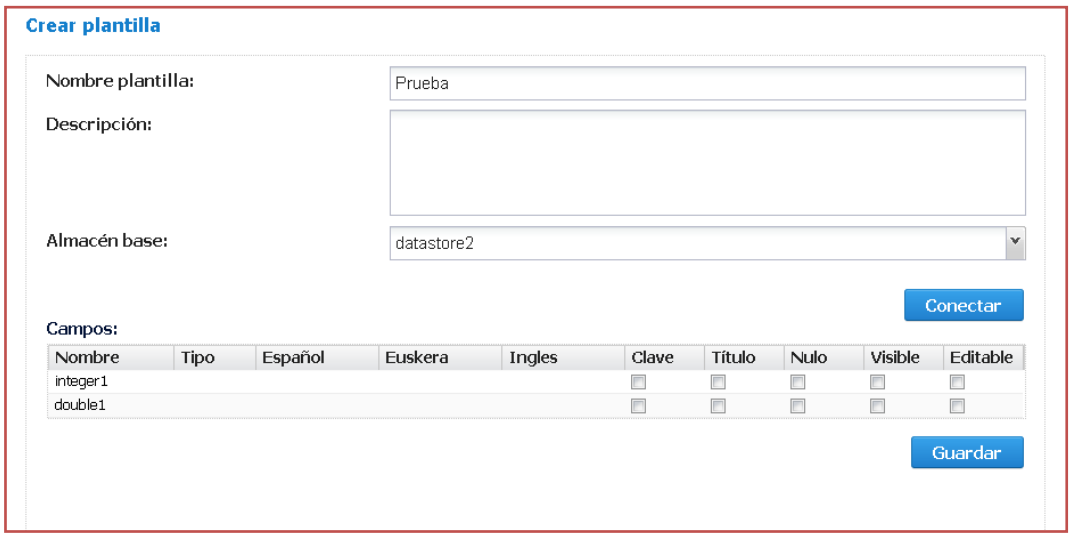


Ilustración 114 Resultado final TemplateCreateDataPanel

ANEXO VI: SERVICIOS MÓDULO DE DATOS

El objetivo del presente capítulo es definir los métodos u operaciones que conforman la lógica necesaria del módulo de gestión de datos (Data Management Service, de aquí en adelante, **DMS**) de la Plataforma GeoServicios Online.

La plataforma GeoServicios Online se compone de un conjunto de servicios web que encapsulan la lógica de la misma.

Estos servicios, orientados a ser integrados en los diferentes productos desarrollados por Geograma, pero también abiertos a posibles integraciones de terceros, están preparados para ser consumidos mediante:

- **Servicios web SOAP**, orientados a ser consumidos por aplicaciones de escritorio mediante protocolo SOAP
- **Servlets**, orientados a ser consumidos en formato JSON por aplicaciones web
- **API Javascript**, orientado a ser consumidos por aplicaciones web o móviles con soporte Javascript

1. SERVICIOS WEB SOAP

1.1 GETDATASTORES

Devuelve la información de datastores asociados a una determinada cuenta de la Plataforma GeoServicios. Opcionalmente, se pueden indicar una serie de filtros sobre los almacenes a obtener, o si se desea obtener sólo un resumen de los almacenes o toda su información asociada.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
filter	String	(Opcional) Filtro sobre los almacenes a obtener (1)
extendedInfo	Boolean	Indica si se desea obtener también la información de la plantilla asociada al almacén

Tabla 37 Parámetros de entrada GetDataStores SOAP

(1) El formato del filtro asociado a la petición deberá ser de la siguiente forma (JSON)

```
{
	"filters": [{
		"filterType": "relation",
		"filterBody": {
			"type": "OR",
			"filters": [{
				"filterType": "simple",
				"filterBody": {
					"field": "groupname",
					"operator": "=",
					"type": "string",
					"value": "default"
				}
			},
			{
				"filterType": "relation",
				"filterBody": {
					"type": "AND",
					"filters": [{
						"filterType": "simple",
						"filterBody": {
							"field": "groupname",
							"operator": "=",
							"type": "string",
							"value": "Group1"
						}
					},
					{
						"filterType": "simple",
						"filterBody": {
							"field": "description",
							"operator": "=",
							"type": "string",
							"value": "Grupo nuevo creado para el dominio DEMO"
						}
					}]
				}
			}]
		}
	}]
}
```

Posibles valores del atributo 'type' del objeto 'relation': 'AND', 'OR'
Posibles valores del atributo 'field': "groupname", "description"
Posibles valores del atributo 'operator': "=", "!=", "IN", "NOT IN"

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -16: Error en el filtro enviado

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

- **datastores**: (array) Listado de almacenes (cero o varios) asociados al dominio y filtros pedidos.
  - id: Identificador del almacén
  - name: Nombre del almacén
  - sourcetype: Tipo de almacén de datos en función de su alojamiento
  - datatype: Tipo de almacén de datos en función de la información alojada
  - geometrygeneralized: Indica si las geometrías han sido previamente generalizadas o no
  - addeddate: Fecha de creación del almacén
  - lastmoddate: Fecha de última modificación del almacén
  - creator: Usuario creador del almacén
  - datatemplate: Plantilla asociada al almacén
  - name: Nombre de la plantilla asociada
  - description: Descripción de la plantilla asociada

1.2 CREATEDATASTORE

Permite añadir un nuevo almacén de datos asociado a una determinada cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
dsname	String	Nombre del almacén de datos a crear
dssourcetype	String	Indica si el almacén a crear es de la Plataforma (subido, 'P') o externo a ella (conectado, 'E')
dsdatatype	String	Indica si el almacén es de tipo alfanumérico (sin geometría asociada, 'AS') o espacial (con geometría asociada, 'SS')
geomgeneralized	String	Indica si la geometría debe ser almacenada generalizada o no (sólo si los datos del almacén son de tipo espacial, 'SS')
host	String	Host del servidor de base de datos del que obtener las tablas disponibles
port	String	Puerto de escucha del servidor de base de datos
dbconnector	String	Conector del servidor de base de datos
dbname	String	Nombre de la base de datos
user	String	Nombre del usuario con permisos sobre la base de datos
password	String	Contraseña del usuario con permisos sobre la base de datos
schema	String	Nombre del esquema de base de datos
tablename	String	Nombre de la tabla

Tabla 38 Parámetros de entrada CreateDataStore SOAP

El dominio se obtendrá a partir del usuario creador del almacén. La fecha de creación (y de última modificación) se informará con la fecha del sistema. Asimismo, de manera automática al crear el almacén, se creará una plantilla por defecto, en el idioma por defecto de la cuenta con los nombres de campos como alias.

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -18: Almacén ya existente

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

- **datastoreid**: Identificador univoco del almacén creado.

1.3 MODIFYDATASTORE

Permite modificar un almacén de datos existente asociado a una determinada cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
dsname	String	Nombre del almacén de datos a crear
dssourcetype	String	Indica si el almacén a crear es de la Plataforma (subido, 'P') o externo a ella (conectado, 'E')
dsdatatype	String	Indica si el almacén es de tipo alfanumérico (sin geometría asociada, 'AS') o espacial (con geometría asociada, 'SS')
geomgeneralized	String	Indica si la geometría debe ser almacenada generalizada o no (sólo si los datos del almacén son de tipo espacial, 'SS')
host	String	Host del servidor de base de datos del que obtener las tablas disponibles
port	String	Puerto de escucha del servidor de base de datos
dbconnector	String	Conector del servidor de base de datos

Parámetro	Tipo de Dato	Descripción
dbname	String	Nombre de la base de datos
user	String	Nombre del usuario con permisos sobre la base de datos
password	String	Contraseña del usuario con permisos sobre la base de datos
schema	String	Nombre del esquema de base de datos
tablename	String	Nombre de la tabla
dsid	String	Identificador univoco del almacén de datos
dstpname	String	Nombre de la plantilla asociada al almacén de datos a modificar

Tabla 39 Parámetros de entrada Parámetros de entrada ModifyDataStores SOAP

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -18: Almacén ya existente

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

- **datastoreid**: Identificador univoco del almacén modificado.

1.4 DROPDATASTORE

Permite borrar un almacén de datos existente asociado a una determinada cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
id	String	Identificador univoco del almacén de datos

Tabla 40 Parámetros de entrada DropDataStore SOAP

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -18: Almacén ya existente

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

1.5 SEARCHDATASTORE

Permite buscar almacenes de datos a partir de una o varias palabras de filtrado asociados a una determinada cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
filter	String	Filtro a buscar en la tabla de almacenes

Tabla 41 Parámetros de entrada SearchDataStore SOAP

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado

- **totalProperties**: Número de elementos (almacenes) devueltos por el servicio

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

- **datastores**: (array) Listado de almacenes (cero o varios) asociados al dominio y filtros pedidos.
  - id: Identificador del almacén
  - name: Nombre del almacén
  - sourcetype: Tipo de almacén de datos en función de su alojamiento
  - datatype: Tipo de almacén de datos en función de la información alojada
  - geometrygeneralized: Indica si las geometrías han sido previamente generalizadas o no
  - addeddate: Fecha de creación del almacén
  - lastmoddate: Fecha de última modificación del almacén
  - creator: Usuario creador del almacén
  - datatemplate: Plantilla asociada al almacén
  - name: Nombre de la plantilla asociada
  - description: Descripción de la plantilla asociada

1.6 GETDATATEMPLATE

Devuelve la información de la plantilla asociada a un determinado almacén de una cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
iddatastore	String	Identificador del almacén para el que obtener la plantilla

Tabla 42 Parámetros de entrada GetDataTemplate SOAP

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -16: Error en el filtro enviado

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

- **datatemplate**: Plantilla asociada al almacén pedido.
  - name: Nombre de la plantilla
  - description: Descripción de la plantilla
  - fields: (array) Listado de campos (uno o varios) que componen la plantilla
    - name: Nombre del campo
    - type: Tipo de campo
    - allownulls: Indica si el campo permite valores nulos o no
    - visible: Indica si el campo es visible o no
    - editable: Indica si el campo el editable o no
    - langs: (array) Listado de idiomas (uno o varios) que permite la plantilla
      - ❖ lang: Código ISO del idioma
      - ❖ alias: Alias del campo para el idioma indicado

1.7 CREATEDATATEMPLATE

Permite añadir una nueva plantilla de datos asociado a una determinada cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
tpname	String	Nombre de la plantilla de datos a crear
tpdescription	String	Descripción de la plantilla de datos a crear
fieldLang	String	Información sobre los campos y alias de la plantilla a crear (1)

Tabla 43 Parámetros de entrada CreateDataTemplate SOAP

El dominio se obtendrá a partir del usuario creador del almacén.
(1) El formato de la información de campos y alias deberá ser de la siguiente forma (JSON)

```
{
	"fields": [{
		"fieldName": "Field1",
		"fieldType": "integer",
		"allowNulls": "false",
		"visible": "true",
		"editable": "true",
		"langs": [{
				"lang": "es",
				"alias": "field1ES"
			}
		]
	},{
		"fieldName": "Field2",
		"fieldType": "varchar",
		"allowNulls": "false",
		"visible": "false",
		"editable": "true",
		"langs": [{
				"lang": "es",
				"alias": "field2ES"
			}
		]
	}]
}
```

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -18: Plantilla ya existente

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

1.8 MODIFYDATATEMPLATE

Permite modificar una plantilla de datos existente asociada a una determinada cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
tpname	String	Nombre de la plantilla de datos a crear
tpdescription	String	Descripción de la plantilla de datos a crear
fieldLang	String	Información sobre los campos y alias de la plantilla a crear (1)

Tabla 44 Parámetros de entrada ModifyDataTemplate SOAP

El dominio se obtendrá a partir del usuario creador del almacén.
(1) El formato de la información de campos y alias deberá ser de la siguiente forma (JSON)

```
{
	"fields": [{
		"fieldName": "Field1",
		"fieldType": "integer",
		"allowNulls": "false",
		"visible": "true",
		"editable": "true",
		"langs": [{
				"lang": "es",
				"alias": "field1ES"
			}
		]
	},{
		"fieldName": "Field2",
		"fieldType": "varchar",
		"allowNulls": "false",
		"visible": "false",
		"editable": "true",
		"langs": [{
				"lang": "es",
				"alias": "field2ES"
			}
		]
	}]
}
```


Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -18: Plantilla ya existente

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

1.9 DROPDATATEMPLATE

Permite añadir una nueva plantilla de datos asociado a una determinada cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
tpname	String	Nombre de la plantilla de datos a borrar

Tabla 45 Parámetros de entrada DropDataTemplate SOAP

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -18: Plantilla ya existente

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

1.10 SEARCHDATATEMPLATE

Permite buscar plantillas de datos a partir de una o varias palabras de filtrado asociados a una determinada cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
filter	String	Filtro a buscar en la tabla de plantillas

Tabla 46 Parámetros de entrada SearchDataTemplate SOAP

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -16: Error en el filtro enviado

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

- **datatemplate**: Plantilla asociada al almacén pedido.
  - name: Nombre de la plantilla
  - description: Descripción de la plantilla
  - fields: (array) Listado de campos (uno o varios) que componen la plantilla
    - name: Nombre del campo
    - type: Tipo de campo
    - allownulls: Indica si el campo permite valores nulos o no
    - visible: Indica si el campo es visible o no
    - editable: Indica si el campo el editable o no
    - langs: (array) Listado de idiomas (uno o varios) que permite la plantilla
      - lang: Código ISO del idioma
      - alias: Alias del campo para el idioma indicado

1.11 GETTABLESFROMCONN

Devuelve las tablas existentes en una determinada base de datos a partir de los datos de conexión de ésta.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
host	String	Host del servidor de base de datos del que obtener las tablas disponibles
port	String	Puerto de escucha del servidor de base de datos
dbconnector	String	Conector del servidor de base de datos
dbname	String	Nombre de la base de datos
user	String	Nombre del usuario con permisos sobre la base de datos
password	String	Contraseña del usuario con permisos sobre la base de datos

Tabla 47 Parámetros de entrada GetTablesFromConn SOAP

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -20: Error al conectarse a la base de datos indicada

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

- **tables**: (array) Listado de tablas (cero o varios) asociadas a la conexión indicada por el usuario.
  - name: Nombre de la tabla

1.12 GETFIELDSFROMCONN

Devuelve los campos existentes de una tabla en una determinada base de datos a partir de los datos de conexión de ésta.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
host	String	Host del servidor de base de datos del que obtener las tablas disponibles
port	String	Puerto de escucha del servidor de base de datos
dbconnector	String	Conector del servidor de base de datos
dbname	String	Nombre de la base de datos
user	String	Nombre del usuario con permisos sobre la base de datos
password	String	Contraseña del usuario con permisos sobre la base de datos
schema	String	Nombre del esquema de base de datos
table	String	Nombre de la tabla/vista sobre la que obtener sus campos

Tabla 48 Parámetros de entrada GetFieldsFromConn SOAP

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -20: Error al conectarse a la base de datos indicada

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

- **fields**: (array) Listado de campos asociadas a la tabla de la conexión indicada por el usuario.
  - name: Nombre del campo
  - type: Tipo del campo

1.13 GETFIELDSFROMDATASTORE

Devuelve los campos existentes de una tabla asociados a un determinado almacén de datos.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
dsid	String	Id del almacén de datos

Tabla 49 Parámetros de entrada GetFieldsFromDataStore SOAP

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -20: No existen campos

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

- **fields**: (array) Listado de campos asociados a el almacén de datos seleccionado.
  - name: Nombre del campo

2. SERVICIOS WEB SERVLET

2.1 GETDATASTORES

Devuelve la información de datastores asociados a una determinada cuenta de la Plataforma GeoServicios. Opcionalmente, se pueden indicar una serie de filtros sobre los almacenes a obtener, o si se desea obtener sólo un resumen de los almacenes o toda su información asociada.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
request	String	Tipo de petición. Para este método, el valor de este parámetro deberá ser 'getDataStores'
start	Int	(Opcional) Orden del grupo a partir del cual obtener el listado (para paginación)
limit	Int	Límite máximo de resultados a devolver por el servicio (para paginación)
callback	String	(Opcional) Función de callback que procesará la petición en el cliente
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
filter	String	(Opcional) Filtro sobre los almacenes a obtener (1)
extendedInfo	Boolean	Indica si se desea obtener también la información de la plantilla asociada al almacén

Tabla 50 Parámetros de entrada GetDataStores SERVLET

(1) El formato del filtro asociado a la petición deberá ser de la siguiente forma (JSON)

```
{
	"filters": [{
		"filterType": "relation",
		"filterBody": {
			"type": "OR",
			"filters": [{
				"filterType": "simple",
				"filterBody": {
					"field": "groupname",
					"operator": "=",
					"type": "string",
					"value": "default"
				}
			},
			{
				"filterType": "relation",
				"filterBody": {
					"type": "AND",
					"filters": [{
						"filterType": "simple",
						"filterBody": {
							"field": "groupname",
							"operator": "=",
							"type": "string",
							"value": "Group1"
						}
					},
					{
						"filterType": "simple",
						"filterBody": {
							"field": "description",
							"operator": "=",
							"type": "string",
							"value": "Grupo nuevo creado para el dominio DEMO"
						}
					}]
				}
			}]
		}
	}]
}
```

Posibles valores del atributo 'type' del objeto 'relation': 'AND', 'OR'
Posibles valores del atributo 'field': "groupname", "description"
Posibles valores del atributo 'operator': "=", "!=", "IN", "NOT IN"

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -16: Error en el filtro enviado

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

- **datastores**: (array) Listado de almacenes (cero o varios) asociados al dominio y filtros pedidos.
  - id: Identificador del almacén
  - name: Nombre del almacén
  - sourcetype: Tipo de almacén de datos en función de su alojamiento
  - datatype: Tipo de almacén de datos en función de la información alojada
  - geometrygeneralized: Indica si las geometrías han sido previamente generalizadas o no
  - addeddate: Fecha de creación del almacén
  - lastmoddate: Fecha de última modificación del almacén
  - creator: Usuario creador del almacén
  - datatemplate: Plantilla asociada al almacén
  - name: Nombre de la plantilla asociada
  - description: Descripción de la plantilla asociada

2.2 CREATEDATASTORE

Permite añadir un nuevo almacén de datos asociado a una determinada cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
request	String	Tipo de petición. Para este método, el valor de este parámetro deberá ser 'createDataStore'
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
dsname	String	Nombre del almacén de datos a crear
dssourcetype	String	Indica si el almacén a crear es de la Plataforma (subido, 'P') o externo a ella (conectado, 'E')

Parámetro	Tipo de Dato	Descripción
dsdatatype	String	Indica si el almacén es de tipo alfanumérico (sin geometría asociada, 'AS') o espacial (con geometría asociada, 'SS')
geomgeneralized	String	Indica si la geometría debe ser almacenada generalizada o no (sólo si los datos del almacén son de tipo espacial, 'SS')
host	String	Host del servidor de base de datos del que obtener las tablas disponibles
port	String	Puerto de escucha del servidor de base de datos
dbconnector	String	Conector del servidor de base de datos
dbname	String	Nombre de la base de datos
user	String	Nombre del usuario con permisos sobre la base de datos
password	String	Contraseña del usuario con permisos sobre la base de datos
schema	String	Nombre del esquema de base de datos
tablename	String	Nombre de la tabla
callback	String	(Opcional) Función de callback que procesará la petición en el cliente

Tabla 51 Parámetros de entrada CreateDataStores SERVLET

El dominio se obtendrá a partir del usuario creador del almacén. La fecha de creación (y de última modificación) se informará con la fecha del sistema. Asimismo, de manera automática al crear el almacén, se creará una plantilla por defecto, en el idioma por defecto de la cuenta con los nombres de campos como alias.

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -18: Almacén ya existente

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

- **datastoreid**: Identificador univoco del almacén creado.

2.3 MODIFYDATASTORE

Permite modificar un almacén de datos existente asociado a una determinada cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
request	String	Tipo de petición. Para este método, el valor de este parámetro deberá ser 'modifyDataStore'
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
dsname	String	Nombre del almacén de datos a crear
dssourcetype	String	Indica si el almacén a crear es de la Plataforma (subido, 'P') o externo a ella (conectado, 'E')
dsdatatype	String	Indica si el almacén es de tipo alfanumérico (sin geometría asociada, 'AS') o espacial (con geometría asociada, 'SS')
geomgeneralized	String	Indica si la geometría debe ser almacenada generalizada o no
host	String	Host del servidor de base de datos del que obtener las tablas disponibles
port	String	Puerto de escucha del servidor de base de datos
dbconnector	String	Conector del servidor de base de datos
dbname	String	Nombre de la base de datos
user	String	Nombre del usuario con permisos sobre la base de datos
password	String	Contraseña del usuario con permisos sobre la base de datos
schema	String	Nombre del esquema de base de datos
tablename	String	Nombre de la tabla
dsid	String	Identificador univoco del almacén de datos
dstpname	String	Nombre de la plantilla asociada al almacén de datos a modificar
callback	String	(Opcional) Función de callback que procesará la petición en el cliente

Tabla 52 Parámetros de entrada ModifyDataStores SERVLET

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -18: Almacén ya existente

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

- **datastoreid**: Identificador univoco del almacén modificado.

2.4 DROPDATASTORE

Permite borrar un almacén de datos existente asociado a una determinada cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
request	String	Tipo de petición. Para este método, el valor de este parámetro deberá ser 'dropDataStore'
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
id	String	Identificador univoco del almacén de datos
callback	String	(Opcional) Función de callback que procesará la petición en el cliente

Tabla 53 Parámetros de entrada DropDataStore SERVLET

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -18: Almacén ya existente

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

2.5 SEARCHDATASTORE

Permite buscar almacenes de datos a partir de una o varias palabras de filtrado asociados a una determinada cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
request	String	Tipo de petición. Para este método, el valor de este parámetro deberá ser 'searchDataStore'
username	String	Nombre de usuario a validar
start	Int	(Opcional) Orden del grupo a partir del cual obtener el listado (para paginación)
limit	Int	Límite máximo de resultados a devolver por el servicio (para paginación)
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
filter	String	Filtro a buscar en la tabla de almacenes
callback	String	(Opcional) Función de callback que procesará la petición en el cliente

Tabla 54 Parámetros de entrada SearchDataStore SERVLET

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado

- **totalProperties**: Número de elementos (almacenes) devueltos por el servicio

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

- **datastores**: (array) Listado de almacenes (cero o varios) asociados al dominio y filtros pedidos.
  - id: Identificador del almacén
  - name: Nombre del almacén
  - sourcetype: Tipo de almacén de datos en función de su alojamiento
  - datatype: Tipo de almacén de datos en función de la información alojada
  - geometrygeneralized: Indica si las geometrías han sido previamente generalizadas o no
  - addeddate: Fecha de creación del almacén
  - lastmoddate: Fecha de última modificación del almacén
  - creator: Usuario creador del almacén
  - datatemplate: Plantilla asociada al almacén
  - name: Nombre de la plantilla asociada
  - description: Descripción de la plantilla asociada

2.6 GETDATATEMPLATE

Devuelve la información de la plantilla asociada a un determinado almacén de una cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
request	String	Tipo de petición. Para este método, el valor de este parámetro deberá ser 'getDataTemplate'
username	String	Nombre de usuario a validar
start	Int	(Opcional) Orden del grupo a partir del cual obtener el listado (para paginación)
limit	Int	Límite máximo de resultados a devolver por el servicio (para paginación)
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
iddatastore	String	Identificador del almacén para el que obtener la plantilla
callback	String	(Opcional) Función de callback que procesará la petición en el cliente

Tabla 55 Parámetros de entrada GetDataTemplate SERVLET

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado

- -16: Error en el filtro enviado

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

- **datatemplate**: Plantilla asociada al almacén pedido.
  - name: Nombre de la plantilla
  - description: Descripción de la plantilla
  - fields: (array) Listado de campos (uno o varios) que componen la plantilla
    - name: Nombre del campo
    - type: Tipo de campo
    - allownulls: Indica si el campo permite valores nulos o no
    - visible: Indica si el campo es visible o no
    - editable: Indica si el campo el editable o no
    - langs: (array) Listado de idiomas (uno o varios) que permite la plantilla
      - lang: Código ISO del idioma
      - alias: Alias del campo para el idioma indicado

2.7 CREATEDATATEMPLATE

Permite añadir una nueva plantilla de datos asociado a una determinada cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
request	String	Tipo de petición. El valor de este parámetro deberá ser 'createDataTemplate'
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
tpname	String	Nombre de la plantilla de datos a crear
tpdescription	String	Descripción de la plantilla de datos a crear
fieldLang	String	Información sobre los campos y alias de la plantilla a crear (1)
callback	String	(Opcional) Función de callback que procesará la petición en el cliente

Tabla 56 Parámetros de entrada CreateDataTemplate SERVLET

El dominio se obtendrá a partir del usuario creador del almacén.
(1) El formato de la información de campos y alias deberá ser de la siguiente forma (JSON)

```
{
	"fields": [{
			"fieldName": "Field1",
			"fieldType": "integer",
			"allowNulls": "false",
			"visible": "true",
			"editable": "true",
			"langs": [{
					"lang": "es",
					"alias": "field1ES"
				}
			]
		},{
			"fieldName": "Field2",
			"fieldType": "varchar",
			"allowNulls": "false",
			"visible": "false",
			"editable": "true",
			"langs": [{
					"lang": "es",
					"alias": "field2ES"
				}
			]
		}]
}
```

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -18: Plantilla ya existente
- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

2.8 MODIFYDATATEMPLATE

Permite modificar una plantilla de datos existente asociada a una determinada cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
request	String	Tipo de petición. El valor de este parámetro deberá ser 'modifyDataTemplate'
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
tpname	String	Nombre de la plantilla de datos a crear
tpdescription	String	Descripción de la plantilla de datos a crear

Parámetro	Tipo de Dato	Descripción
fieldLang	String	Información sobre los campos y alias de la plantilla a crear (1)
callback	String	(Opcional) Función de callback que procesará la petición en el cliente

Tabla 57 Parámetros de entrada ModifyDataTemplate SERVLET

El dominio se obtendrá a partir del usuario creador del almacén.
(1) El formato de la información de campos y alias deberá ser de la siguiente forma (JSON)

```
{
	"fields": [{
		"fieldName": "Field1",
		"fieldType": "integer",
		"allowNulls": "false",
		"visible": "true",
		"editable": "true",
		"langs": [{
				"lang": "es",
				"alias": "field1ES"
			}
		]
	},{
		"fieldName": "Field2",
		"fieldType": "varchar",
		"allowNulls": "false",
		"visible": "false",
		"editable": "true",
		"langs": [{
				"lang": "es",
				"alias": "field2ES"
			}
		]
	}]
}
```

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -18: Plantilla ya existente

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

2.9 DROPDATATEMPLATE

Permite añadir una nueva plantilla de datos asociado a una determinada cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
request	String	Tipo de petición. El valor de este parámetro deberá ser 'dropDataTemplate'
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
tpname	String	Nombre de la plantilla de datos a borrar
callback	String	(Opcional) Función de callback que procesará la petición en el cliente

Tabla 58 Parámetros de entrada DropDataTemplate SERVLET

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -18: Plantilla ya existente
- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)32.10 searchDataTemplate

2.10 SEARCHDATATEMPLATE

Permite buscar plantillas de datos a partir de una o varias palabras de filtrado asociados a una determinada cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
request	String	Tipo de petición. El valor de este parámetro deberá ser 'searchDataTemplate'
username	String	Nombre de usuario a validar
start	Int	(Opcional) Orden del grupo a partir del cual obtener el listado (para paginación)
limit	Int	Límite máximo de resultados a devolver por el servicio (para paginación)

Parámetro	Tipo de Dato	Descripción
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
filter	String	Filtro a buscar en la tabla de plantillas
callback	String	(Opcional) Función de callback que procesará la petición en el cliente

Tabla 59 Parámetros de entrada SearchDataTemplate SERVLET

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -16: Error en el filtro enviado

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

- **datatemplate**: Plantilla asociada al almacén pedido.
  - name: Nombre de la plantilla
  - description: Descripción de la plantilla
  - fields: (array) Listado de campos (uno o varios) que componen la plantilla
    - name: Nombre del campo
    - type: Tipo de campo
    - allownulls: Indica si el campo permite valores nulos o no
    - visible: Indica si el campo es visible o no
    - editable: Indica si el campo el editable o no
    - langs: (array) Listado de idiomas (uno o varios) que permite la plantilla
      - lang: Código ISO del idioma
      - alias: Alias del campo para el idioma indicado

2.11 GETTABLESFROMCONN

Devuelve las tablas existentes en una determinada base de datos a partir de los datos de conexión de ésta.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
request	String	Tipo de petición. El valor de este parámetro deberá ser 'getTablesFromConn'
username	String	Nombre de usuario a validar

Parámetro	Tipo de Dato	Descripción
start	Int	(Opcional) Orden del grupo a partir del cual obtener el listado (para paginación)
limit	Int	Límite máximo de resultados a devolver por el servicio (para paginación)
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
host	String	Host del servidor de base de datos del que obtener las tablas disponibles
port	String	Puerto de escucha del servidor de base de datos
dbconnector	String	Conector del servidor de base de datos
dbname	String	Nombre de la base de datos
user	String	Nombre del usuario con permisos sobre la base de datos
password	String	Contraseña del usuario con permisos sobre la base de datos
callback	String	(Opcional) Función de callback que procesará la petición en el cliente

Tabla 60 Parámetros de entrada GetTablesFromConn SERVLET

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -20: Error al conectarse a la base de datos indicada

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

- **tables**: (array) Listado de tablas (cero o varios) asociadas a la conexión indicada por el usuario.
  - name: Nombre de la tabla

2.12 GETFIELDSFROMCONN

Devuelve los campos existentes de una tabla en una determinada base de datos a partir de los datos de conexión de ésta.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
request	String	Tipo de petición. El valor de este parámetro deberá ser 'getFieldsFromConn'
username	String	Nombre de usuario a validar
start	Int	(Opcional) Orden del grupo a partir del cual obtener el listado (para paginación)
limit	Int	Límite máximo de resultados a devolver por el servicio (para paginación)
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
host	String	Host del servidor de base de datos del que obtener las tablas disponibles
port	String	Puerto de escucha del servidor de base de datos
dbconnector	String	Conector del servidor de base de datos
dbname	String	Nombre de la base de datos
user	String	Nombre del usuario con permisos sobre la base de datos
password	String	Contraseña del usuario con permisos sobre la base de datos
schema	String	Nombre del esquema de base de datos
table	String	Nombre de la tabla/vista sobre la que obtener sus campos
callback	String	(Opcional) Función de callback que procesará la petición en el cliente

Tabla 61 Parámetros de entrada GetFieldsFromConn SERVLET

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -20: Error al conectarse a la base de datos indicada

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

- **fields**: (array) Listado de campos asociadas a la tabla de la conexión indicada por el usuario.
  - name: Nombre del campo
  - type: Tipo del campo

2.13 GETFIELDSFROMDATASTORE

Devuelve los campos existentes de una tabla asociados a un determinado almacén de datos.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
request	String	Tipo de petición. El valor de este parámetro deberá ser 'getFieldsFromDataStore'
username	String	Nombre de usuario a validar
start	Int	(Opcional) Orden del grupo a partir del cual obtener el listado (para paginación)
limit	Int	Límite máximo de resultados a devolver por el servicio (para paginación)
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
dsid	String	Id del almacén de datos
callback	String	(Opcional) Función de callback que procesará la petición en el cliente

Tabla 62 Parámetros de entrada GetFieldsFromDataStore SERVLET

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -20: No existen campos

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

- **fields**: (array) Listado de campos asociados a el almacén de datos seleccionado.
  - name: Nombre del campo

3. SERVICIOS WEB JAVASCRIPT

La especificación de los métodos de la capa de servicios Javascript, es análoga a la capa Servlet, con la diferencia de que todos los métodos añadirán el siguiente parámetro:

- method: "GET" o "POST"
- No se introducen los parámetros **start** y **limit** en los métodos **getDataTemplate, getDataStore, getTablesFromConn** y **getFieldsFromConn**.

Por lo tanto, en este apartado, se expondrá un ejemplo de utilización de cada uno de los métodos para observar cual debe de ser su comportamiento.

3.1 GETDATASTORE

Devuelve la información de datastores asociados a una determinada cuenta de la Plataforma GeoServicios. Opcionalmente, se pueden indicar una serie de filtros sobre los almacenes a obtener, o si se desea obtener sólo un resumen de los almacenes o toda su información asociada.

Ejemplo:

```
GEOSERVICIOS.dms.getDataStores({
        method: "POST",
        username: "theUser",
        key: "theKey",
        filter: "",
        extendedInfo: isExtendedInfo,
        callback: thecallback
    });

    function thecallback(response){
        //Do your work here
    }
```

3.2 CREATEDATASTORE

Permite añadir un nuevo almacén de datos asociado a una determinada cuenta de la Plataforma GeoServicios.

Ejemplo:

```
//Instantiate the API service - "GEOSERVICIOS.dms"
        GEOSERVICIOS.dms.createDataStore({
            method: "POST",
            username: "theUser",
            key: "theKey",
            dsname : "theDsname",
            dssourcetype: "theDssourcetype",
            dsdatatype: "theDsdatatype",
            geomgeneralized: isGeomgeneralized,
            host: "theHost",
            port: thePort,
            dbconnector: "theDbconnecto",
            dbname: "theDbname",
            user: "theUser",
            password: "thePassword",
            schema: "theSchema",
            tablename:  "theTablename",
            callback: thecallback
        });
        function thecallback(response){
            //Do your work here
        }
```

3.3 MODIFYDATASTORE

Permite modificar un almacén de datos existente asociado a una determinada cuenta de la Plataforma GeoServicios.

Ejemplo:

```
//Instantiate the API service - "GEOSERVICIOS.dms"
        GEOSERVICIOS.dms.modifyDataStore({
            method: "POST",
            username: "theUser",
            key: "theKey",
            dsname : "theDsname",
            dstpname: "dstpname",
            dsid:"theDsid",
            dssourcetype: "theDssourcetype",
            dsdatatype: "theDsdatatype",
            geomgeneralized: isGeomgeneralized,
            host: "theHost",
            port: thePort,
            dbconnector: "theDbconnecto",
            dbname: "theDbname",
            user: "theUser",
            password: "thePassword",
            schema: "theSchema",
            tablename:  "theTablename",
            callback: thecallback
        });
        function thecallback(response){
            //Do your work here
        }
```

3.4 DROPDATASTORE

Permite borrar un almacén de datos existente asociado a una determinada cuenta de la Plataforma GeoServicios.

Ejemplo:

```
//Instantiate the API service - "GEOSERVICIOS.dms"
        GEOSERVICIOS.dms.dropDataStore({
            method: "POST",
            username: "theUser",
            key: "theKey",
            id: "theId",
            callback: thecallback
        });

        function thecallback(response){
            //Do your work here
        }
```

3.5 SEARCHDATASTORES

Permite buscar almacenes de datos a partir de una o varias palabras de filtrado asociados a una determinada cuenta de la Plataforma GeoServicios.

Ejemplo:

```
//Instantiate the API service - "GEOSERVICIOS.dms"
        GEOSERVICIOS.dms.searchDataStores({
            method: "POST",
            username: "theUser",
            key: "theKey",
            filter: "theFilter",
            callback: thecallback
        });

        function thecallback(response){
            //Do your work here
        }
```

3.6 GETDATATEMPLATE

Devuelve la información de la plantilla asociada a un determinado almacén de una cuenta de la Plataforma GeoServicios.

Ejemplo:

```
//Instantiate the API service - "GEOSERVICIOS.dms"
        GEOSERVICIOS.dms.getDataTemplate({
            method: "POST",
            username: "theUsername",
            key: "theKey",
            iddatastore: "theIddatastore",
            callback: thecallback
        });

        function thecallback(response){
            //Do your work here
        }
```

3.7 CREATEDATATEMPLATE

Permite añadir una nueva plantilla de datos asociado a una determinada cuenta de la Plataforma GeoServicios.

Ejemplo:

```
//Instantiate the API service - "GEOSERVICIOS.dms"
        GEOSERVICIOS.dms.createDataTemplate({
            method: "POST",
            username: "theUser",
            key: "theKey",
            tpname: "theTpname",
            tpdescription: "theTpdescription",
            fieldLang: "theFieldLang",
            callback: thecallback
        });

        function thecallback(response){
            //Do your work here
        }
```

3.8 MODIFYDATATEMPLATE

Permite modificar una plantilla de datos existente asociada a una determinada cuenta de la Plataforma GeoServicios.

Ejemplo:

```
//Instantiate the API service - "GEOSERVICIOS.dms"
        GEOSERVICIOS.dms.modifyDataTemplate({
            method: "POST",
            username: "theUser",
            key: "theKey",
            tpname: "theTpname",
            tpdescription: "theTpdescription",
            fieldLang: "theFieldLang",
            callback: thecallback
        });

        function thecallback(response){
            //Do your work here
        }
```

3.9 DROPDATATEMPLATE

Permite añadir una nueva plantilla de datos asociado a una determinada cuenta de la Plataforma GeoServicios.

Ejemplo:

```
//Instantiate the API service - "GEOSERVICIOS.dms"
        GEOSERVICIOS.dms.dropDataTemplate({
            method: "POST",
            username: "theUser",
            key: "theKey",
            name: "theName",
            callback: thecallback
        });

        function thecallback(response){
            //Do your work here
        }
```

3.10 SEARCHDATATEMPLATE

Permite buscar plantillas de datos a partir de una o varias palabras de filtrado asociados a una determinada cuenta de la Plataforma GeoServicios.

Ejemplo:

```
//Instantiate the API service - "GEOSERVICIOS.dms"
        GEOSERVICIOS.dms.searchDatatemplates({
            method: "POST",
            username: "theUser",
            key: "theKey",
            filter: "theFilter",
            callback: thecallback
        });

        function thecallback(response){
            //Do your work here
        }
```

3.11 GETTABLESFROMCONN

Devuelve las tablas existentes en una determinada base de datos a partir de los datos de conexión de ésta.

Ejemplo:

```
//Instantiate the API service - "GEOSERVICIOS.dms"
        GEOSERVICIOS.dms.getTablesFromConn({
            method: "POST",
            username: "theUser",
            key: "theKey",
            host: "theHost",
            port: "thePort",
            dbconnector: "theDbconnector",
            dbname: "theDbname",
            user: "theUser"
            password: "thePassword"
            callback: thecallback
        });

        function thecallback(response){
            //Do your work here
        }
```

3.12 GETFIELDSFROMCONN

Devuelve los campos existentes de una tabla en una determinada base de datos a partir de los datos de conexión de ésta.

Ejemplo:

```
//Instantiate the API service - "GEOSERVICIOS.dms"
        GEOSERVICIOS.dms.getFieldsFromConn({
            method: "POST",
            username: "theUsername",
            key: "theKey",
            host: "theHost",
            port: "thePort",
            dbconnector: "theDbconnector",
            dbname: "theDbname",
            user: "theUser",
            password: "thePass",
            schema: "theSchema",
            table:"theTable",
            callback: thecallback
        });

        function thecallback(response){
            //Do your work here
        }
```

3.13 GETFIELDSFROMDATASTORE

Devuelve los campos existentes de una tabla asociados a un determinado almacén de datos.

Ejemplo:

```
//Instantiate the API service - "GEOSERVICIOS.dms"
        GEOSERVICIOS.dms.getFieldsFromDataStore({
            method: "POST",
            username: "theUsername",
            key: "theKey",
            dsid: "theDsid",
            callback: thecallback
        });

        function thecallback(response){
            //Do your work here
        }
```

ANEXO VII: COMPONENTES MÓDULO GEOASSET

En el siguiente Anexo, se detalla el diseño y especificación de cada uno de los componentes del módulo geoAsset. En dicho anexo, se detalla tanto la interfaz gráfica, los casos de uso, como los diagramas de secuencia de cada uno de los componentes.

Los componentes del módulo geoAsset que se especifican, son los siguientes:

Componentes relativos a permisos a nivel de aplicación:

- AssociateUserGroupPanel
- AssociateDataPanel
- AssociateMapPanel

Cada uno de los componentes esta asociado a ciertos servicios que se irán identificando al realizar los casos de uso pertinentes. Dichos servicios están especificados en el AnexoVIII: Servicios móduo geoAsset.

1.COMPONENTE ASSOCIATEUSERGROUPPANEL

El siguiente componente, se encargará de mostrar tanto el listado de usuarios como el listado de grupos relacionados con una determinada cuenta de la Plataforma GeoServicios Online. Dicho componente, se encargará de establecer permisos a nivel de aplicación tanto de usuarios como de grupos.

1.1 DISEÑO INTERFAZ GRÁFICA

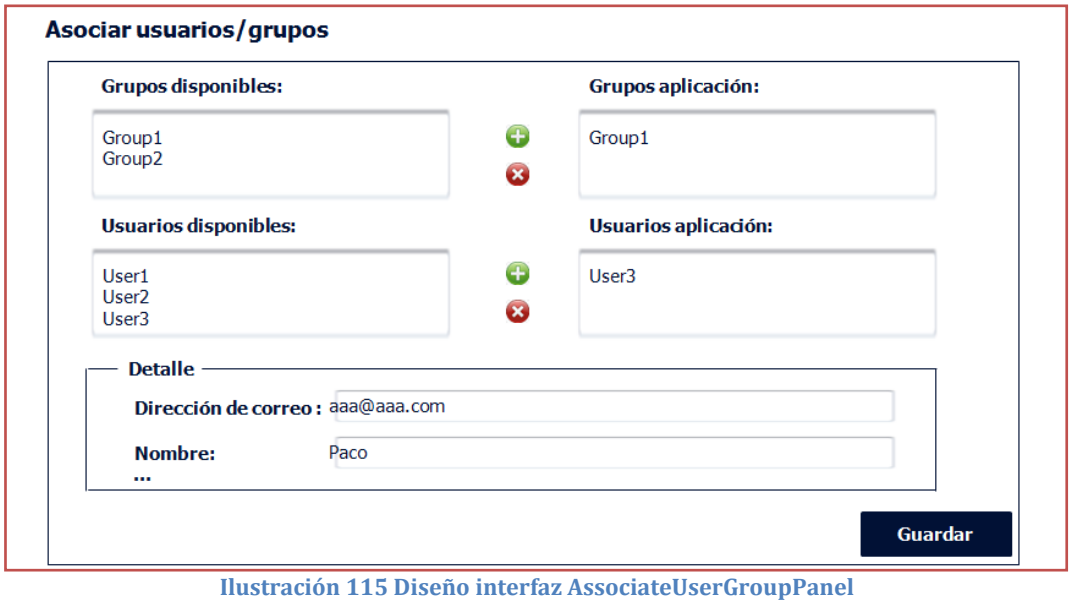


Ilustración 115 Diseño interfaz AssociateUserGroupPanel

1.2 CASO DE USO

Las funcionalidades identificados en el siguiente componente son las que siguen:

- **Obtener listado de usuarios**
- **Obtener listado de grupos**
- **Obtener permisos relativos a usuarios y grupos**
- **Guardar permisos usuarios y grupos**

El diagrama de dicho caso de uso será el siguiente:

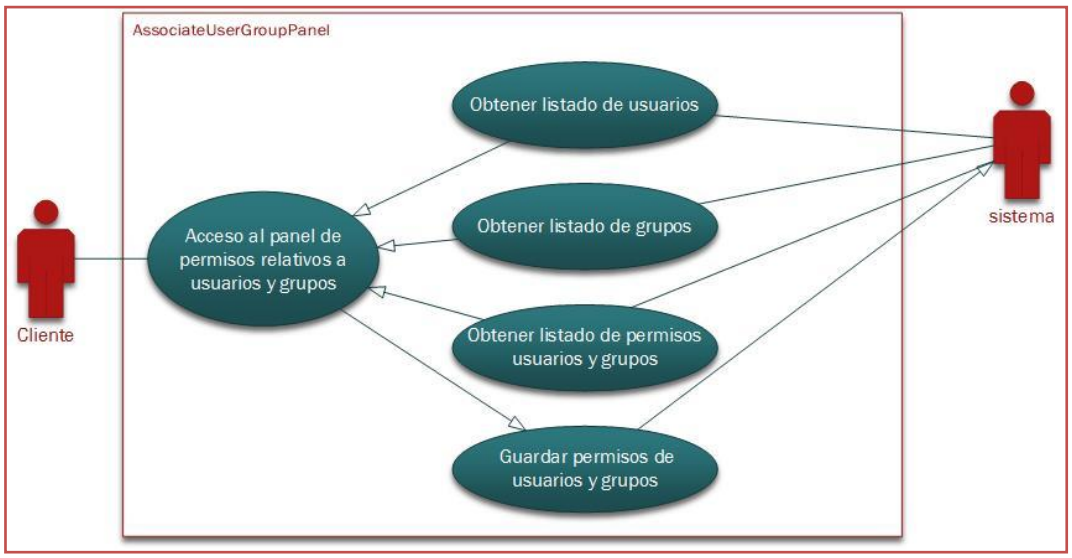


Ilustración 116 Caso de uso AssociateUserGroupPanel

Obtener listado de usuarios: El sistema se encargará de devolver una lista de usuarios y mostrarla en un listado. Mediante esta funcionalidad identificamos el siguiente servicio:

- getUsers

Obtener listado de grupos: El sistema se encargará de devolver una lista de grupos y mostrarla en un listado. Mediante esta funcionalidad identificamos el siguiente servicio:

- getGroups

Obtener listado de permisos usuarios y grupos: El sistema se encargará de devolver una lista de permisos de usuarios y grupos y mostrarla en un listado. Mediante esta funcionalidad identificamos el siguiente servicio:

- getAcl

Guardar permisos de usuarios y grupos: El sistema se encargará de almacenar los permisos establecidos a nivel de aplicación relativos a usuario y grupos. Mediante esta funcionalidad identificamos el siguiente servicio:

- setAcl

1.3 DIAGRAMA DE SECUENCIA

Mediante el análisis realizado mediante el caso de uso e identificación de funcionalidades, procederemos a crear el diagrama de secuencia que será el siguiente.

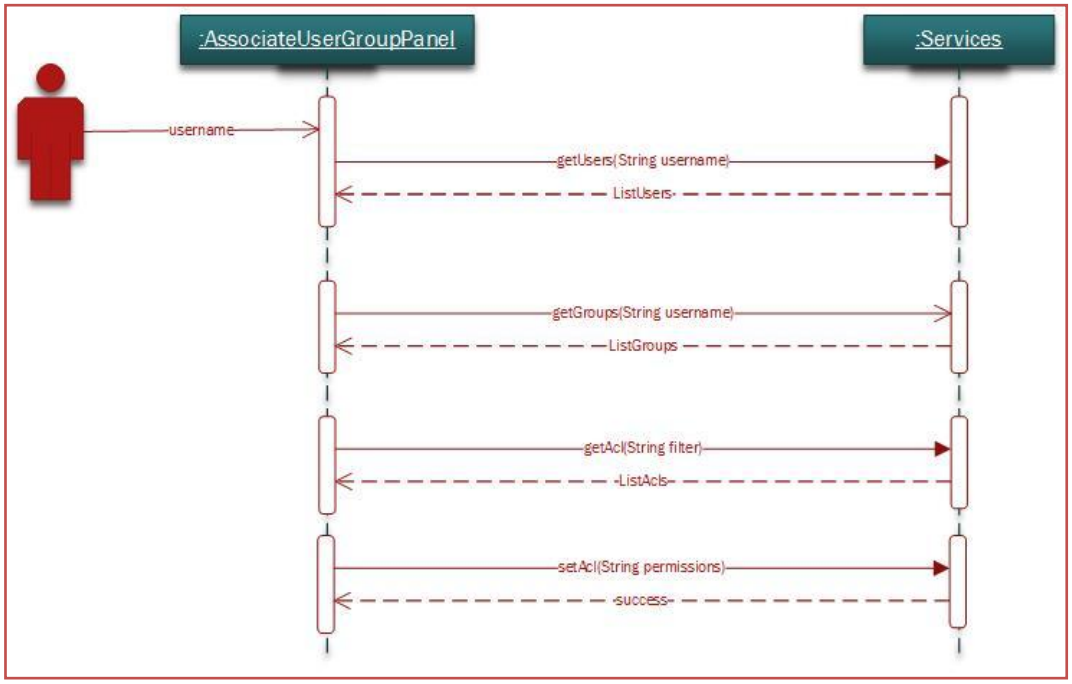


Ilustración 117 Diagrama de secuencia AssociateUserGroupPanel

1.4 RESULTADO FINAL

El resultado final del componente una vez integrado, es el que se puede observar a continuación:

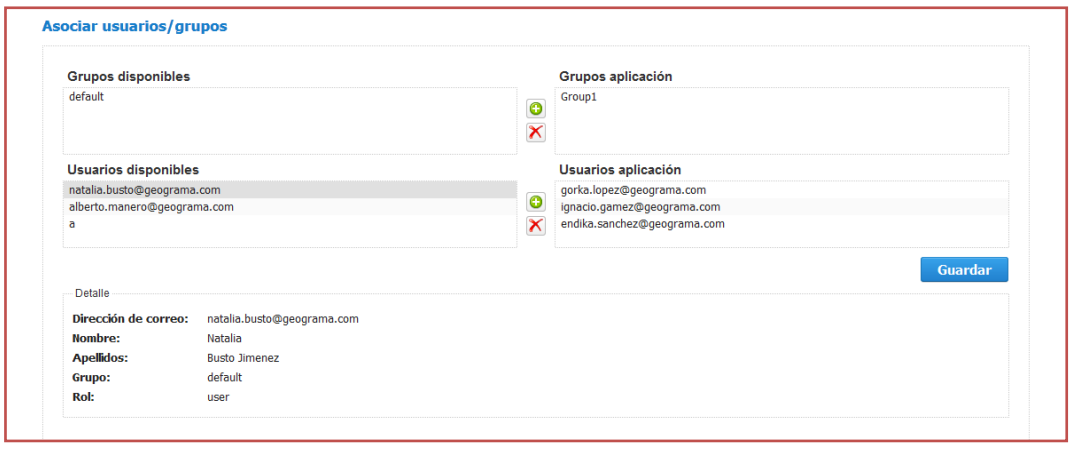


Ilustración 118 Resultado final AssociateUserGroupPanel

2.COMPONENTE ASSOCIATEDATAPANEL

El siguiente componente, se encargará de mostrar el listado de almacenes de datos relacionados con una determinada cuenta de la Plataforma GeoServicios Online. Dicho componente, se encargará de establecer permisos a nivel de aplicación relativos a los almacenes de datos.

2.1 DISEÑO INTERFAZ GRÁFICA

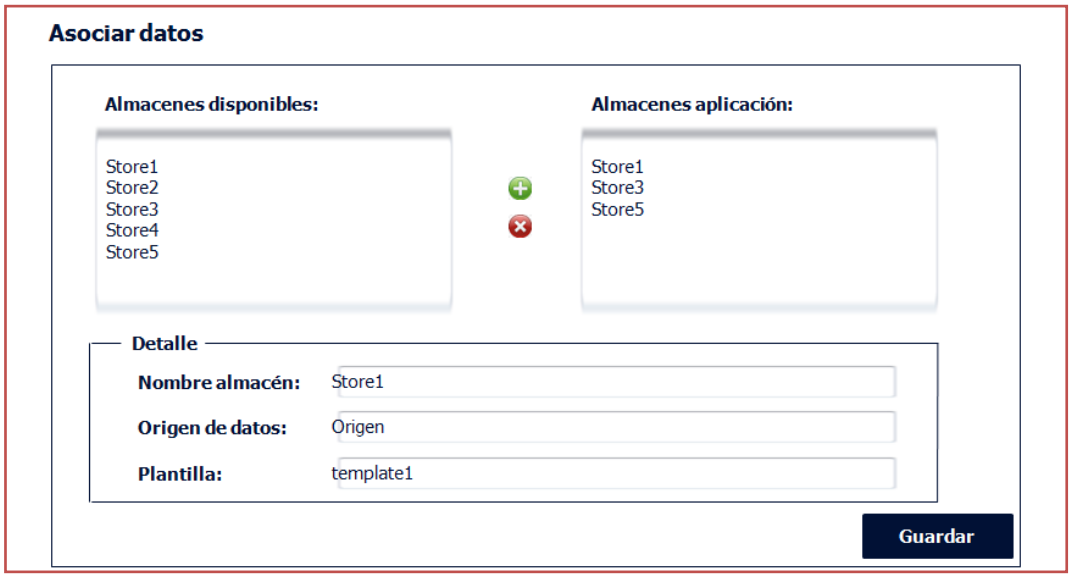


Ilustración 119 Diseño interfaz AssociateDataPanel

2.2 CASO DE USO

Las funcionalidades identificados en el siguiente componente son las que siguen:

- **Obtener listado de almacenes de datos**
- **Obtener permisos relativos a almacenes de datos**
- **Guardar permisos relativos a almacenes de datos**

El diagrama de dicho caso de uso será el siguiente:

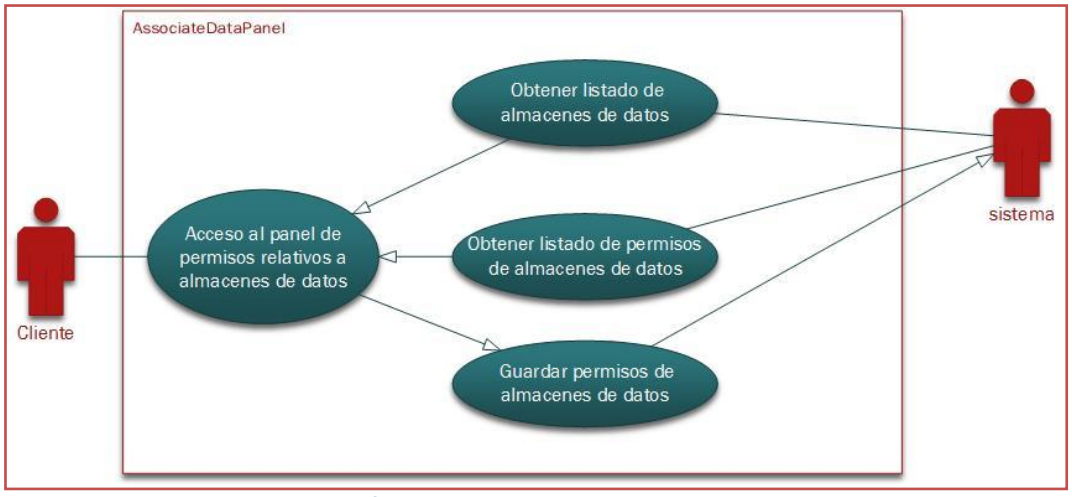


Ilustración 120 Caso de uso AssociateDataPanel

Obtener listado de almacenes de datos: El sistema se encargará de devolver una lista de almacenes de datos y mostrarla en un listado. Mediante esta funcionalidad identificamos el siguiente servicio:

- getDataStores

Obtener listado de permisos relativos a almacenes de datos: El sistema se encargará de devolver una lista de permisos de almacenes de datos y mostrarla en un listado. Mediante esta funcionalidad identificamos el siguiente servicio:

- getAcl

Guardar permisos de relativos a almacenes de datos: El sistema se encargará de almacenar los permisos establecidos a nivel de aplicación relativos a almacenes de datos. Mediante esta funcionalidad identificamos el siguiente servicio:

- setAcl

2.3 DIAGRAMA DE SECUENCIA

Mediante el análisis realizado mediante el caso de uso e identificación de funcionalidades, procederemos a crear el diagrama de secuencia que será el siguiente.

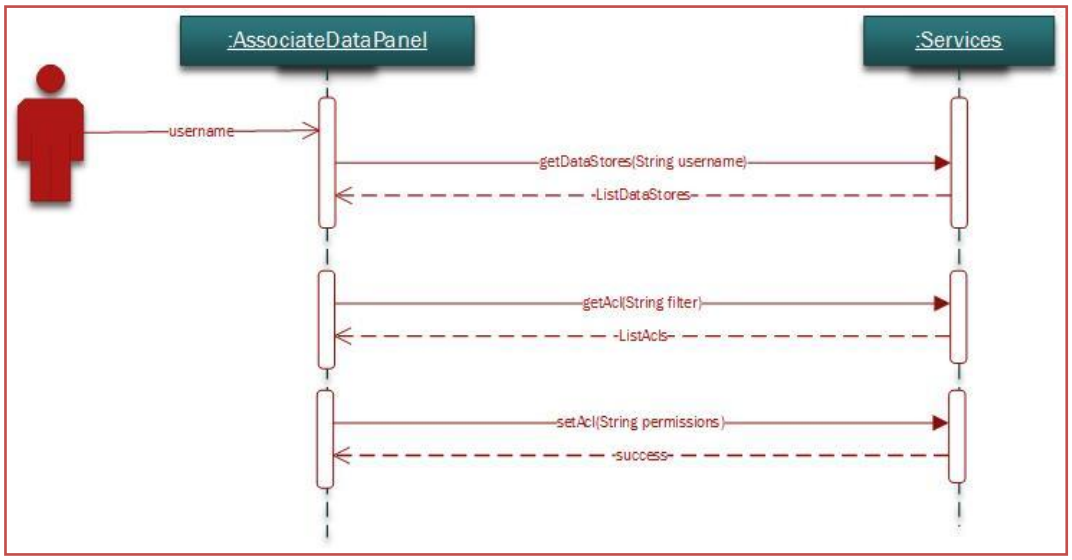


Ilustración 121 Diagrama de secuencia AssociateDataPanel

2.4 RESULTADO FINAL

El resultado final del componente una vez integrado, es el que se puede observar a continuación:

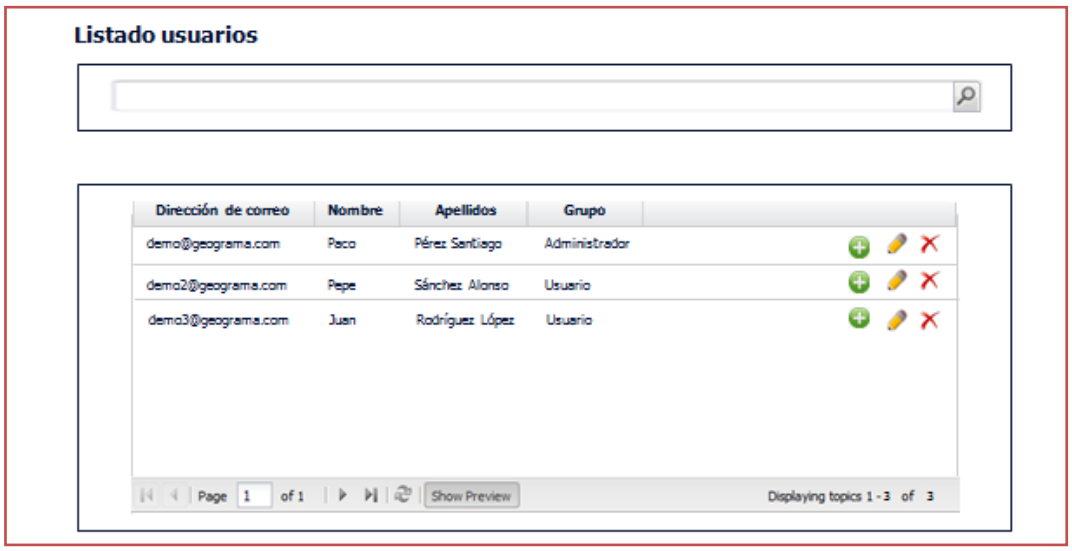


Ilustración 122 Resultado final AssociateDataPanel

3.COMPONENTE ASSOCIATEMAPPANEL

El siguiente componente, se encargará de mostrar el listado de mapas relacionados con una determinada cuenta de la Plataforma GeoServicios Online. Dicho componente, se encargará de establecer permisos a nivel de aplicación relativos a los mapas.

3.1 DISEÑO INTERFAZ GRÁFICA

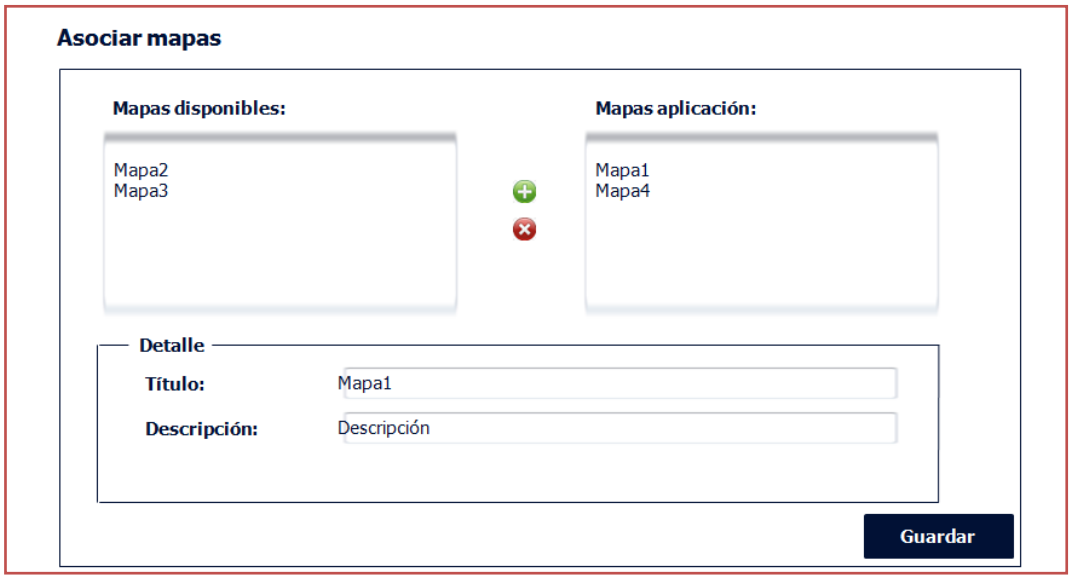


Ilustración 123 Diseño interfaz AssociateMapPanel

3.2 CASO DE USO

Las funcionalidades identificados en el siguiente componente son las que siguen:

- **Obtener listado de mapas**
- **Obtener permisos relativos a mapas**
- **Guardar permisos relativos a mapas**

El diagrama de dicho caso de uso será el siguiente:

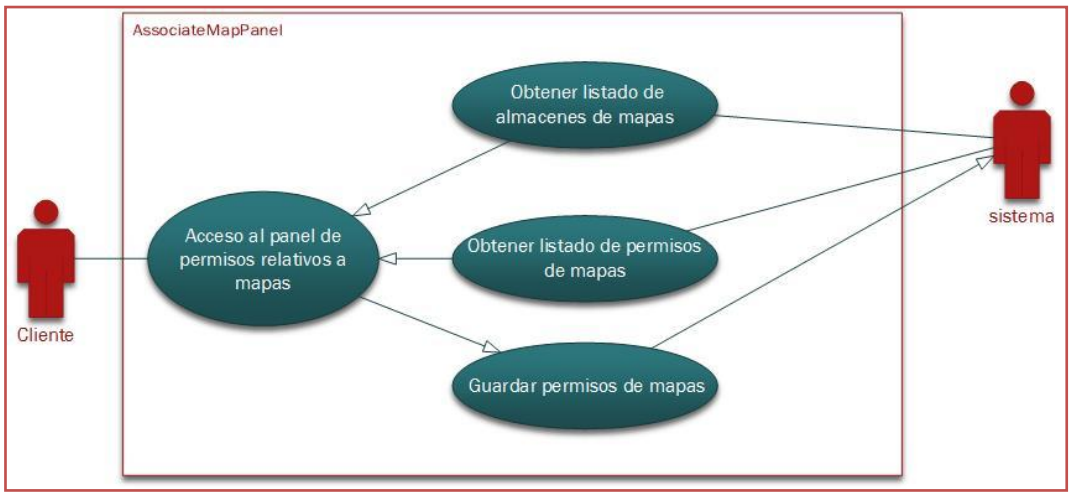


Ilustración 124 Caso de uso AssociateMapPanel

Obtener listado de mapas: El sistema se encargará de devolver una lista de mapas y mostrarla en un listado.

Obtener listado de permisos relativos a mapas: El sistema se encargará de devolver una lista de permisos relativos a mapas y mostrarla en un listado. Mediante esta funcionalidad identificamos el siguiente servicio:

- getAcl

Guardar permisos relativos de mapas: El sistema se encargará de almacenar los permisos establecidos a nivel de aplicación relativos a mapas. Mediante esta funcionalidad identificamos el siguiente servicio:

- setAcl

3.3 DIAGRAMA DE SECUENCIA

Mediante el análisis realizado mediante el caso de uso e identificación de funcionalidades, procederemos a crear el diagrama de secuencia que será el siguiente.

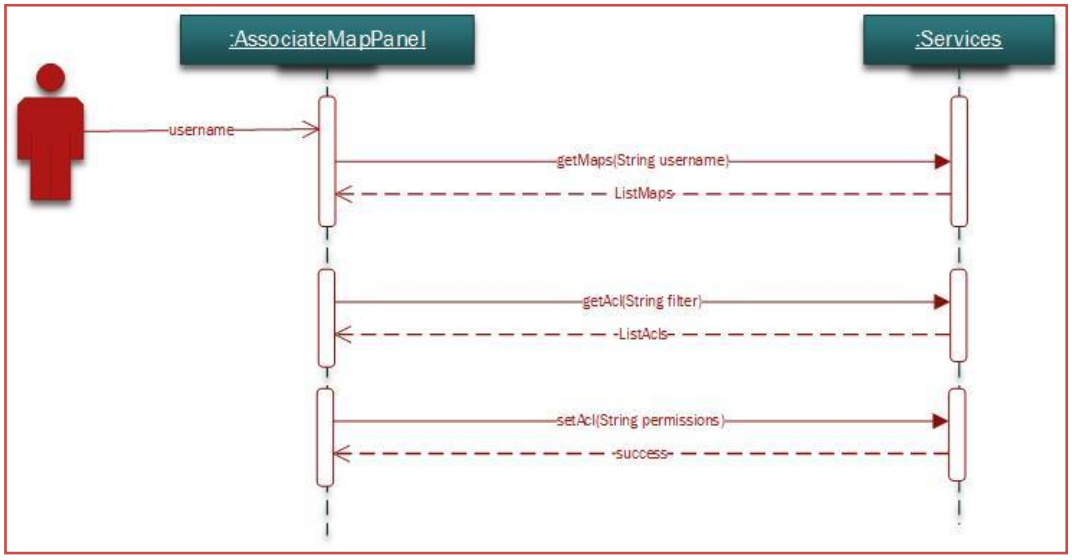


Ilustración 125 Diagrama de secuencia AssociateMapPanel

3.4 RESULTADO FINAL

El resultado final del componente una vez integrado, es el que se puede observar a continuación:

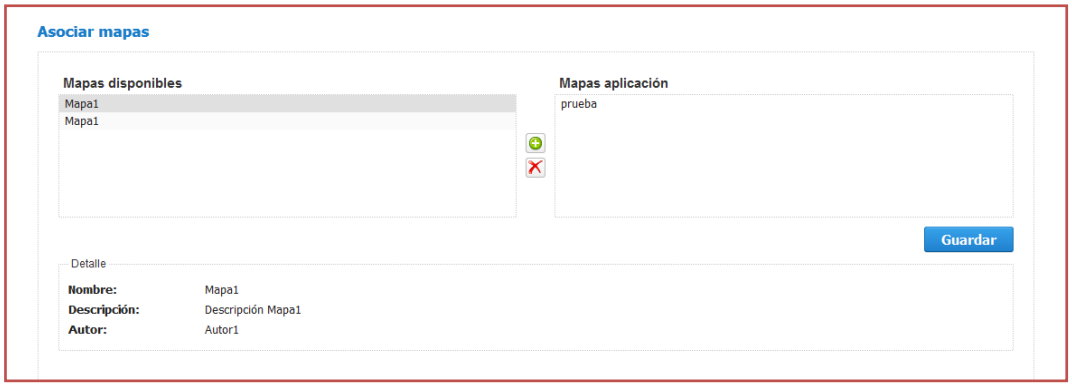


Ilustración 126 Resultado final AssociateMapPanel

ANEXO VIII: SERVICIOS MÓDULO GEOASSET

El objetivo del presente capítulo es definir los métodos u operaciones que conforman la lógica necesaria del módulo geoAsset (Access Control List Management Service, de aquí en adelante, **ACLMS**) de la Plataforma GeoServicios Online.

La plataforma GeoServicios Online se compone de un conjunto de servicios web que encapsulan la lógica de la misma.

Estos servicios, orientados a ser integrados en los diferentes productos desarrollados por Geograma, pero también abiertos a posibles integraciones de terceros, están preparados para ser consumidos mediante:

- **Servicios web SOAP**, orientados a ser consumidos por aplicaciones de escritorio mediante protocolo SOAP
- **Servlets**, orientados a ser consumidos en formato JSON por aplicaciones web
- **API Javascript**, orientado a ser consumidos por aplicaciones web o móviles con soporte Javascript

1. SERVICIOS WEB SOAP

1.1 GETACL

Devuelve la información de permisos asociados a una determinada cuenta de la Plataforma GeoServicios. Opcionalmente, se pueden indicar una serie de filtros sobre los tipos de permisos a obtener.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
filter	String	Filtro asociado al tipo de permisos a obtener. El formato será: (type1|type2|...)

Tabla 63 Parámetros de entrada GetAcl SOAP

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -16: Filtro enviado no válido

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

- **acls**: (array) Listado de permisos (cero o varios) asociados al dominio y filtros pedidos.
  - name: Nombre del objeto asociado a el permiso
  - type: Tipo de objeto
  - permission: Tipo de permiso para el objeto

1.2 SETACL

Permite añadir una serie de permisos de aplicación asociados a una determinada cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
permissions	String	Contendrá el listado de permisos a establecer (1)

Tabla 64 Parámetros de entrada SetAcl SOAP

(1) El formato asociado a la petición deberá ser de la siguiente forma (JSON)

```
{
        "permissions": [{
                "acl": [{
                        "name": "user1",
                        "type": "user",
                        "permission": "R"
                },
                {
                        "name": "datastore1",
                        "type": "datastore",
                        "permission": "R"
                },
                {
                        "name": "map1",
                        "type": "map",
                        "permission": "R"
                }]
        }]
}
```

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:

  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -18: Permisos no validos

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

2. SERVICIOS WEB SERVLET

2.1 GETACL

Devuelve la información de permisos asociados a una determinada cuenta de la Plataforma GeoServicios. Opcionalmente, se pueden indicar una serie de filtros sobre los tipos de permisos a obtener.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
request	String	Tipo de petición. Para este método, el valor de este parámetro deberá ser 'getAcl'
start	Int	(Opcional) Orden del grupo a partir del cual obtener el listado (para paginación)
limit	Int	Límite máximo de resultados a devolver por el servicio (para paginación)
callback	String	(Opcional) Función de callback que procesará la petición en el cliente
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
filter	String	Filtro asociado al tipo de permisos a obtener. El formato será: (type1|type2|...)

Tabla 65 Parámetros de entrada GetAcl SERVLET

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:

- -1: Excepción genérica del método
- -2: Parámetros no válidos
- -3: Testigo no válido para el usuario y/o producto indicado
- -16: Filtro enviado no válido

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

- **acls**: (array) Listado de permisos (cero o varios) asociados al dominio y filtros pedidos.
  - name: Nombre del objeto asociado a el permiso
  - type: Tipo de objeto
  - permission: Tipo de permiso para el objeto

2.2 SETACL

Permite añadir una serie de permisos de aplicación asociados a una determinada cuenta de la Plataforma GeoServicios.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
request	String	Tipo de petición. Para este método, el valor de este parámetro deberá ser 'setAcl'
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
permissions	String	Contendrá el listado de permisos a establecer (1)

Tabla 66 Parámetros de entrada SetAcl SERVLET

(1) El formato asociado a la petición deberá ser de la siguiente forma (JSON)

```
{
        "permissions": [{
                "acl": [{
                        "name": "user1",
                        "type": "user",
                        "permission": "R"
                },
                {
                        "name": "datastore1",
                        "type": "datastore",
                        "permission": "R"
                },
                {
                        "name": "map1",
                        "type": "map",
                        "permission": "R"
                }]
        }]
}
```

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:

  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien
- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado
  - -18: Permisos no validos

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

3. SERVICIOS WEB JAVASCRIPT

La especificación de los métodos de la capa de servicios Javascript, es análoga a la capa Servlet, con la diferencia de que todos los métodos añadirán el siguiente parámetro:

- method: "GET" o "POST"

- No se introducen los parámetros **start** y **limit** en el método **getAcl**.

Por lo tanto, en este apartado, se expondrá un ejemplo de utilización de cada uno de los métodos para observar cual debe de ser su comportamiento.

3.1 GETACL

Devuelve la información de permisos asociados a una determinada cuenta de la Plataforma GeoServicios. Opcionalmente, se pueden indicar una serie de filtros sobre los tipos de permisos a obtener.

Ejemplo:

```
GEOSERVICIOS.aclms.getAcl({
     method: "POST",
     username: "theUser",
     key: "theKey",
     filter: "user|group",
     callback: thecallback
 });

 function thecallback(response){
     //Do your work here
 }
```

3.2 SETACL

Permite añadir una serie de permisos de aplicación asociados a una determinada cuenta de la Plataforma GeoServicios.

Ejemplo:

```
GEOSERVICIOS.aclms.setAcl({
    method: "POST",
    username: "theUser",
    key: "theKey",
    permissions : "JSON permission",
    callback: thecallback
});
function thecallback(response){
    //Do your work here
}
```

ANEXO IX: COMPONENTES MEJORA MÓDULO DE DATOS

En el siguiente Anexo, se detalla el diseño y especificación de cada uno de los componentes del módulo mejora de datos. En dicho anexo, se detalla tanto la interfaz gráfica, los casos de uso, como los diagramas de secuencia de cada uno de los componentes.

Los componentes del módulo mejora de datos que se especifican, son los siguientes:

- UploadDataPanel

Cada uno de los componentes esta asociado a ciertos servicios que se irán identificando al realizar los casos de uso pertinentes. Dichos servicios están especificados en el Anexo X: Servicios módulo mejora de datos.

1.COMPONENTE UPLOADDATAPANEL

El siguiente componente, se encargará de gestionar la subida de datos relacionados con una determinada cuenta de la Plataforma GeoServicios Online. La funcionalidad de dicho componente, se dividirá en dos pasos principales:

- Subida de datos
- Carga de datos:
  - Datos de tipo CSV
  - Datos de tipo SHP

1.1 DISEÑO INTERFAZ GRÁFICA

Ilustración 127 Diseño interfaz UploadDataPanel

Una vez cargado los datos, tendremos dos situaciones posibles:

- Archivo de tipo SHP

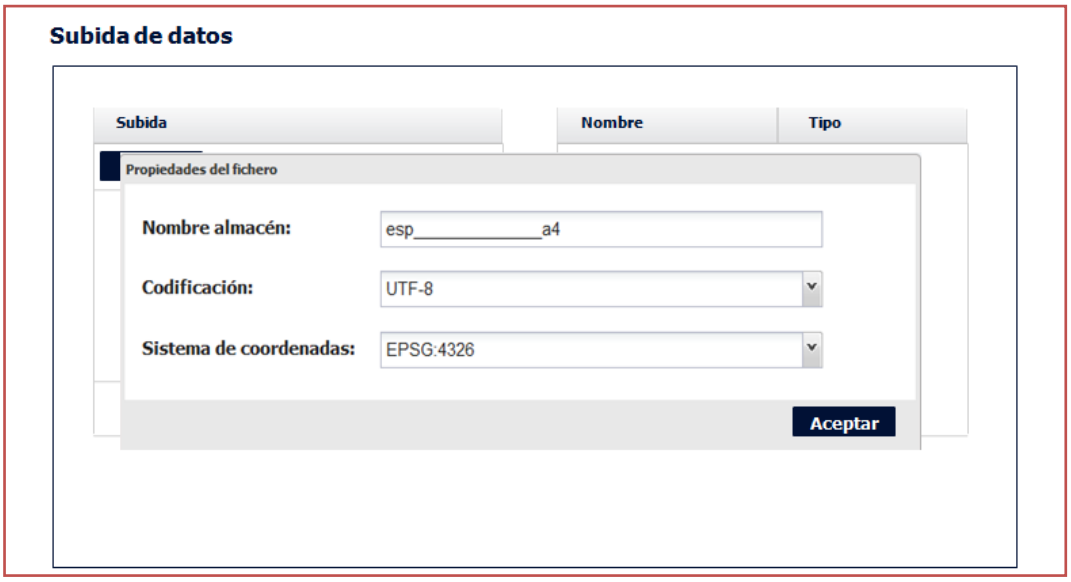


Ilustración 128 Diseño interfaz UploadDataPanel SHP

- Archivo de tipo CSV

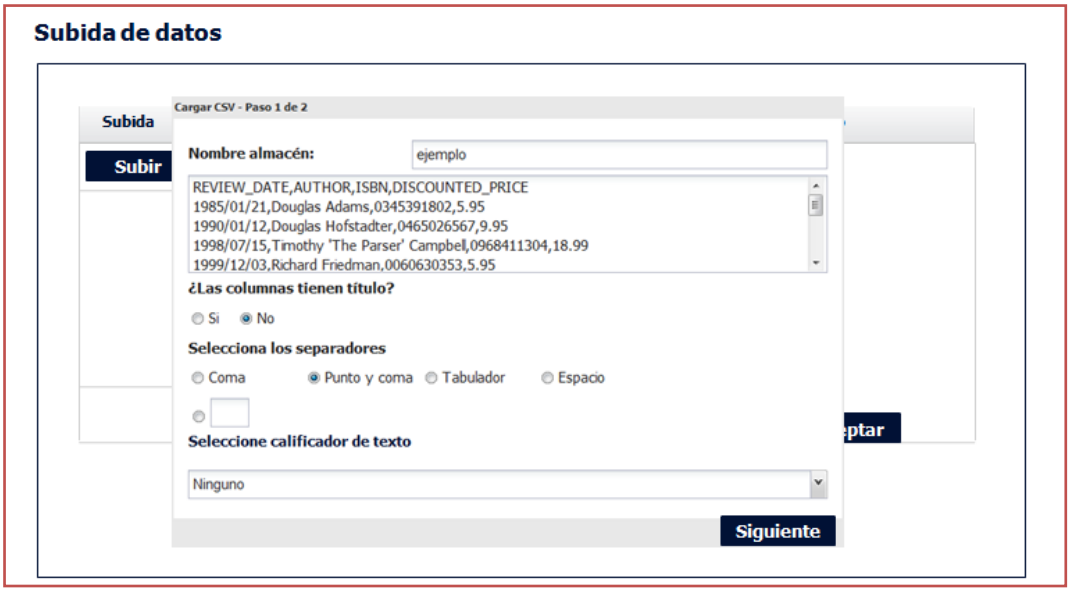


Ilustración 129 Diseño interfaz UploadDataPanel CSV

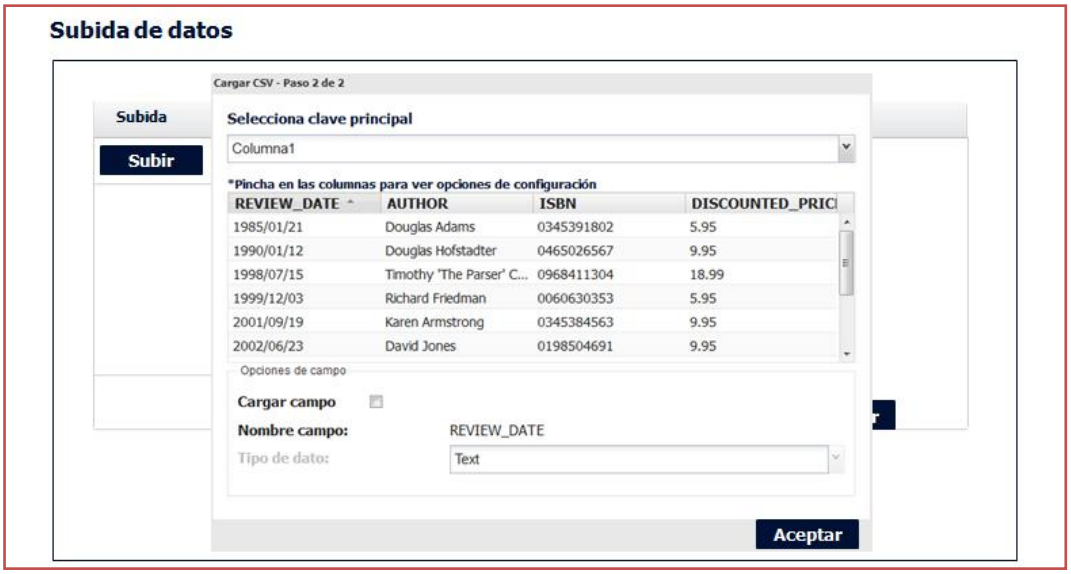


Ilustración 130 Diseño interfaz UploadDataPanel CSV 2

1.2 CASO DE USO

Las funcionalidades identificados en el siguiente componente son las que siguen:

- **Obtener listado de almacenes de datos**
- **Seleccionar archivos a subir**
- **Subir archivos**
- **Comprobar tipo de archivo**

En caso de que el archivo sea de tipo SHP

- **Obtener sistema de coordenadas**
- **Cargar datos**

En caso de que el archivo sea de tipo CSV

- **Obtener contenido archivo CSV**
- **Cargar datos**

El diagrama de dicho caso de uso será el siguiente:

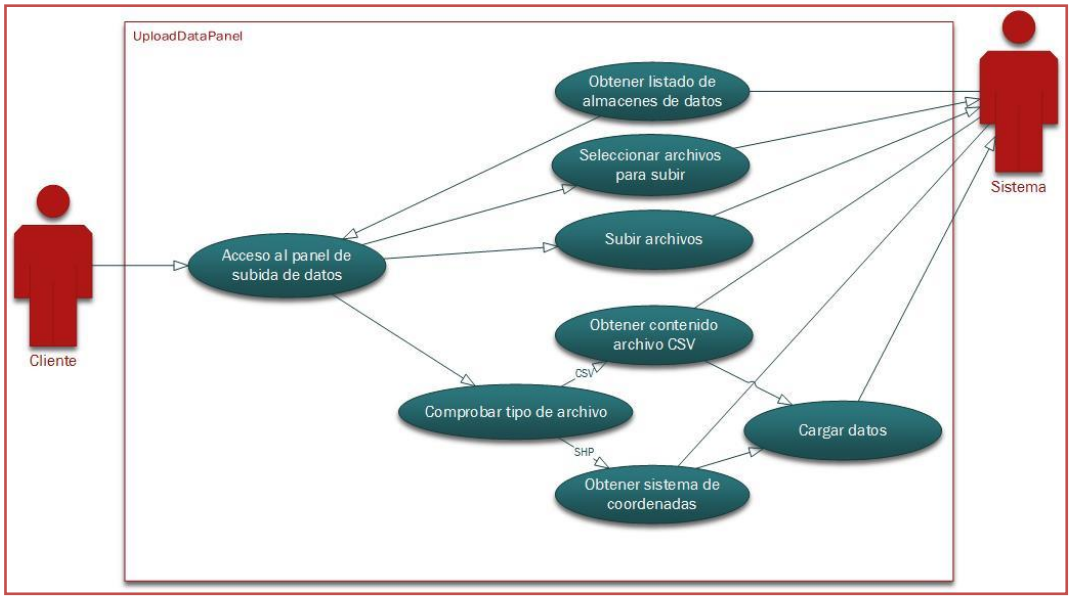


Ilustración 131 Diagrama de secuencia UploadDataPanel

Obtener listado de almacenes de datos: El sistema se encargará de devolver un listado de almacenes de datos y mostrarlo en un listado. Mediante esta funcionalidad identificamos el siguiente servicio:

- getDataStores

Seleccionar archivos para subir: El sistema se encargará integrar la funcionalidad para poder seleccionar los archivos a subir. Esto se podrá realizar tanto arrastrando como mediante el botón seleccionar archivos

Subir archivos: El sistema se encargará de realizar la subida de los archivos seleccionados. Mediante esta funcionalidad identificamos el siguiente servicio:

- uploadData

Comprobar tipo de archivo: El sistema se encargará de comprobar si el archivo es de tipo CSV o SHP.

Obtener contenido archivo CSV: El sistema se encargará de obtener el contenido del archivo CSV. Esta funcionalidad está relacionada con la respuesta del siguiente servicio:

- uploadData

Obtener sistema de coordenadas: El sistema se encargará de obtener un listado de los sistemas de coordenadas disponibles.

Cargar datos: El sistema se encargará de realizar la carga de datos. Esta constará en la creación de un almacén de datos y una tabla con dicha información en una determinada base de datos. Mediante esta funcionalidad, identificamos el siguiente servicio:

- loadData

1.3 DIAGRAMA DE SECUENCIA

Mediante el análisis realizado mediante el caso de uso e identificación de funcionalidades, procederemos a crear el diagrama de secuencia que será el siguiente.

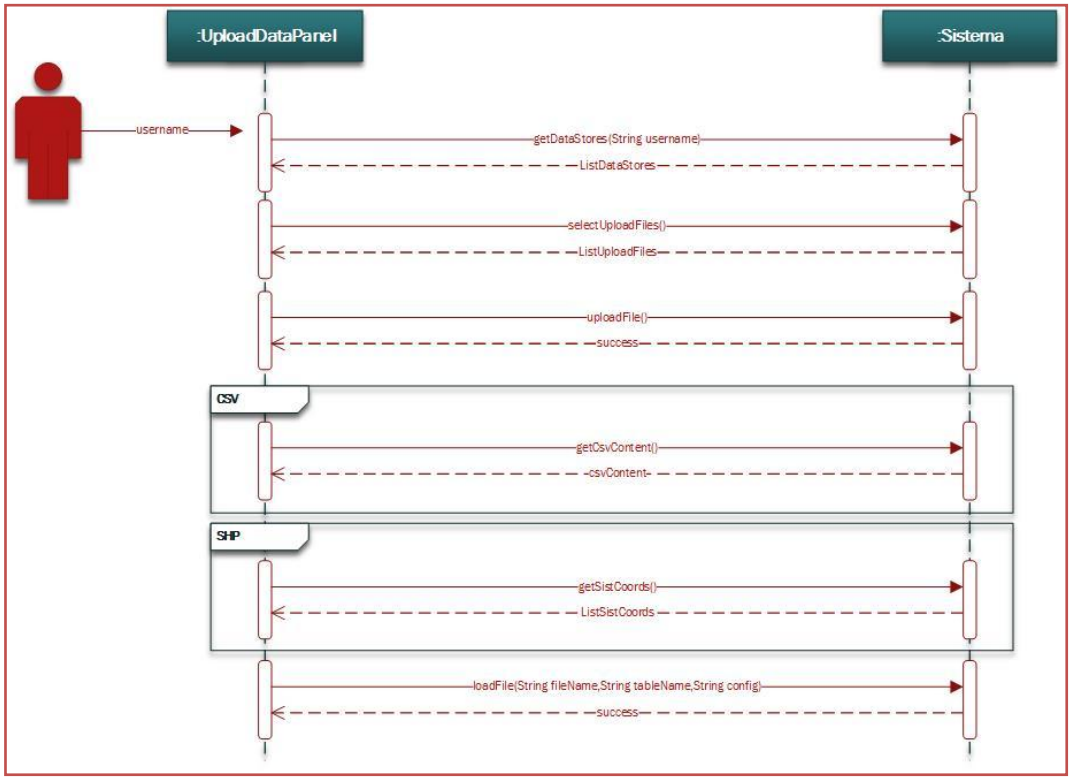


Ilustración 132 Diagrama de secuencia UploadDataPanel

1.4 RESULTADO FINAL

El resultado final del componente una vez integrado, es el que se puede observar a continuación:

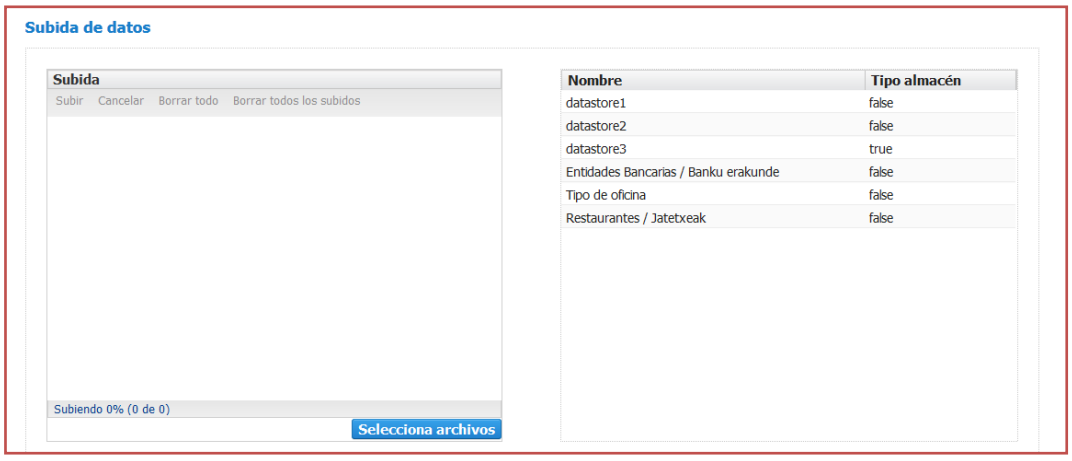


Ilustración 133Resultado final UploadDataPanel

En caso de que el archivo sea de tipo SHP:

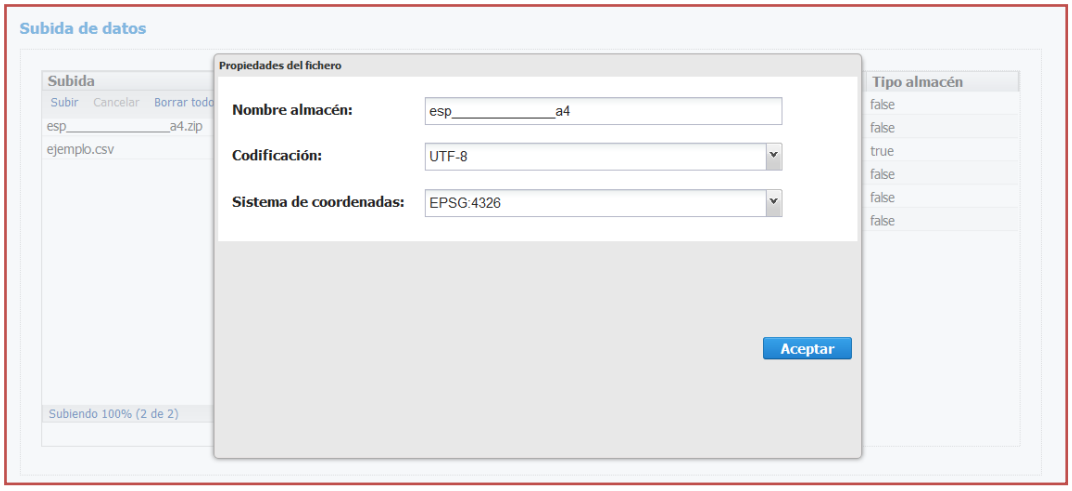


Ilustración 134 Resultado final UploadDataPanel SHP

En caso de que el archivo sea de tipo CSV:

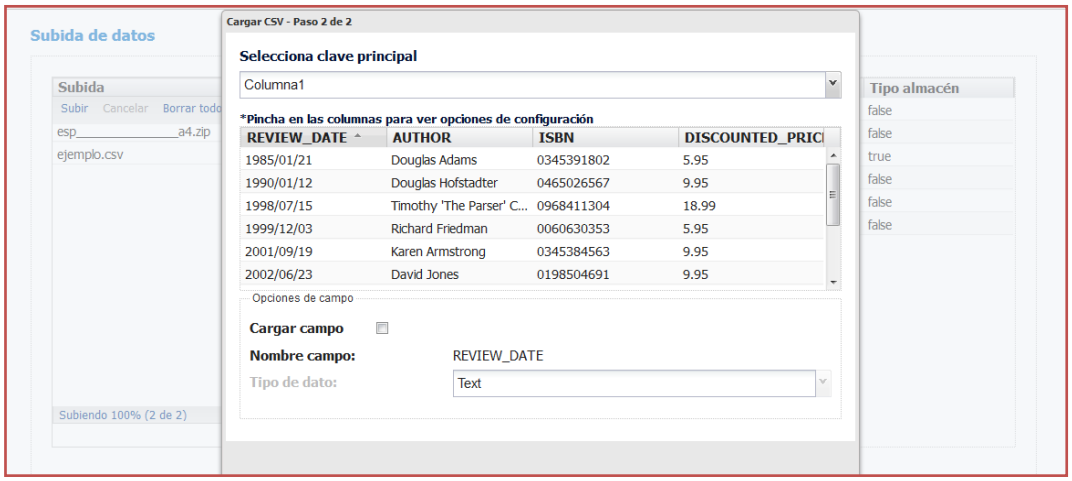


Ilustración 135 Resultado final UploadDataPanel CSV 1

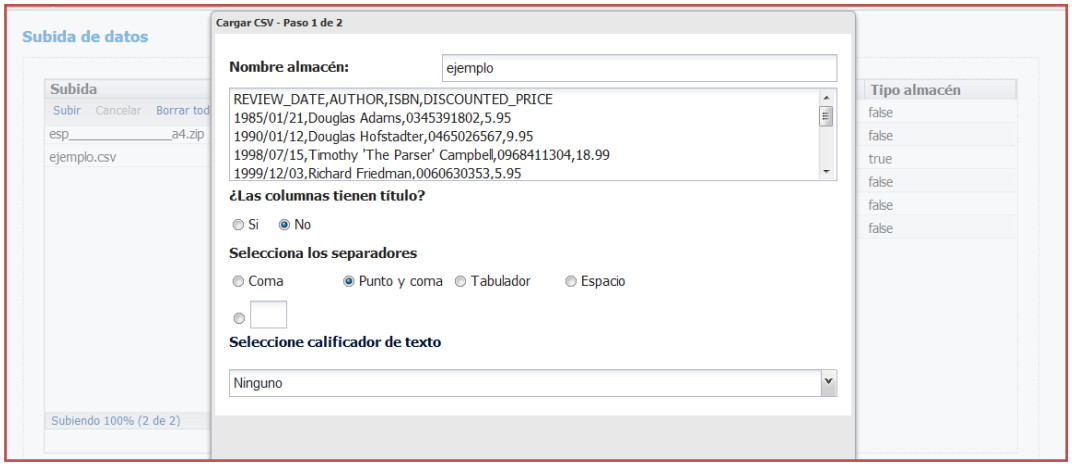


Ilustración 136 Resultado final UploadDataPanel CSV 2

ANEXO X: SERVICIOS MEJORA MÓDULO DE DATOS

El objetivo del presente capítulo es definir los métodos u operaciones que conforman la lógica necesaria del módulo mejora módulo de datos (Se incluirán dentro del servicio ya existente, Data Management Service, **DMS**) de la Plataforma GeoServicios Online.

La plataforma GeoServicios Online se compone de un conjunto de servicios web que encapsulan la lógica de la misma.

Estos servicios, orientados a ser integrados en los diferentes productos desarrollados por Geograma, pero también abiertos a posibles integraciones de terceros, están preparados para ser consumidos mediante:

- **Servicios web SOAP**, orientados a ser consumidos por aplicaciones de escritorio mediante protocolo SOAP
- **Servlets**, orientados a ser consumidos en formato JSON por aplicaciones web
- **API Javascript**, orientado a ser consumidos por aplicaciones web o móviles con soporte Javascript.

En este caso, los métodos creados para la mejora del módulo de datos, solo harán uso de las capas Servlet y javascript.

1. SERVICIOS WEB SERVLET

1.1 UPLOADDATA

Se encarga de realizar la subida de datos a una ubicación determinada establecida a nivel de configuración de aplicación.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
request	String	Tipo de petición. Para este método, el valor de este parámetro deberá ser 'uploadData'
callback	String	(Opcional) Función de callback que procesará la petición en el cliente
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken

Tabla 67 Parámetros de entrada UploadData SOAP

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado

- **fileName**:(String) Nombre del fichero que se ha subido.

- **csvFile**:(String) (Solo en caso de que el fichero a subir sea de tipo CSV) Contenido del fichero CSV.

1.2 LOADDATA

Permite realizar la carga de archivos de tipo CSV y SHP. Mediante este método, se generará el almacén de datos y las tablas correspondientes al contenido de dichos archivos.

Parámetros de entrada:

Parámetro	Tipo de Dato	Descripción
request	String	Tipo de petición. Para este método, el valor de este parámetro deberá ser 'loadData'
username	String	Nombre de usuario a validar
token	String	Testigo asociado al usuario a validar obtenido mediante el método getToken
tableName	String	Nombre de la tabla y almacén de datos a crear.
fileName	String	Nombre del fichero a cargar.
config	String	Contendrá un listado de parámetros para la creación de el almacén de datos. (1)
callback	String	(Opcional) Función de callback que procesará la petición en el cliente

Tabla 68 Parámetros de entrada LoadData SOAP

(1) El formato asociado a la petición deberá ser de la siguiente forma (JSON)

```
{
	"config": [{
		"parameters": [{
			"coord": "EPG4526",
			"encoding": "utf-8",
		},
		{
			"load": "true",
			"type": "Text",
			"extraParams":[{
				"param1": "paam1"
			},{
				"param2": "paam2"
			}]
		}]
	}]
}
```

Parámetros de salida:

- **success**: (boolean) Indica si se ha ejecutado correctamente el método. Posibles valores:
  - false: En caso de que al método no se le haya pasado los parámetros indicados, haya terminado con excepciones o el token indicado no sea válido
  - true: En caso de que todo haya ido bien

- **error**: (int) Indica, en caso de que no se haya ejecutado el método de forma correcta, el error asociado. Éste puede ser:
  - -1: Excepción genérica del método
  - -2: Parámetros no válidos
  - -3: Testigo no válido para el usuario y/o producto indicado

- **metadata**: Informa de los metadatos asociados a la petición (tiempo de respuesta del servicio, tamaño de la respuesta, etc.)

2.SERVICIOS WEB JAVASCRIPT

La especificación de los métodos de la capa de servicios Javascript, es análoga a la capa Servlet, con la diferencia de que todos los métodos añadirán el siguiente parámetro:

- method: "GET" o "POST"

Por lo tanto, en este apartado, se expondrá un ejemplo de utilización de cada uno de los métodos para observar cual debe de ser su comportamiento.

2.1 UPLOADDATA

Se encarga de realizar la subida de datos a una ubicación determinada establecida a nivel de configuración de aplicación.

Ejemplo:

```
GEOSERVICIOS.dms.uploadData({
    method: "POST",
    username: "theUser",
    key: "theKey",
    callback: thecallback
});

function thecallback(response){
    //Do your work here
}
```

2.2 LOADDATA

Permite realizar la carga de archivos de tipo CSV y SHP. Mediante este método, se generará el almacén de datos y las tablas correspondientes al contenido de dichos archivos.

Ejemplo:

```
GEOSERVICIOS.dms.loadData({
    method: "POST",
    username: "theUser",
    key: "theKey",
    tableName: "theTableName",
    fileName: "theFileName",
    config : "JSON permission",
    callback: thecallback
});
function thecallback(response){
    //Do your work here
```

ANEXO XI: SEGUIMIENTO DIARIO

JULIO2013

	LUNES	MARTES	MIÉRCOLES	JUEVES	VIERNES	SÁB/DOM
SEMANA 1	1	2 Aprendizaje Open Layer y extJs	3 Aprendizaje Open Layer y extJs	4 Aprendizaje Open Layer y extJs	5 Análisis OpenTripPlanner	6/7
notas		8:30 horas	8:30 horas	8:30 horas	6 horas	
SEMANA 2	8 Análisis OpenTripPlanner	9 Análisis OpenTripPlanner	10 TENERIFE EXAMENES	11 TENERIFE EXAMENES	12 TENERIFE EXAMENES	13/14
notas	8:30 horas	8:30 horas				
SEMANA 3	15 TENERIFE EXAMENES	16 Aprendizaje Open Layer y extJs	17 (1) Aprendizaje Open Layer y extJs (2) Instalación y configuración Hibernate (3) Document. crear servicio soap (RECURSO ESCRITO)	18 (1)Documentación crear servicio soap. (2)Actualización componentes core	19 COMENZAR PANEL ADMINISTRACIÓN Login_c	20/21
notas		8:30 horas	(1) 4 hora (2) 2 hora (3) 2:30 horas	(1) 4:30 horas (2) 4 horas	6 horas	
SEMANA 4	22 Desarrollo pantalla Login_c	23 Desarrollo panel ListUserPanel **(MODULO USUARIOS)**	24 Finalización panel LoginPanel	25 Día Festivo	26 Desarrollo panel ListUserPanel **(MODULO USUARIOS)**	27/28
notas	8:30 horas	8:30 horas	8:30 horas		8:30 horas	
SEMANA 5	29 Desarrollo panel List User panel Desarollo panel CreateUserPanel	30 Finalización panel ListUserPanel Finalización panel CreateUserPanel	31 Desarrollo servicios getUser Desarrollo servicio createUser **(SERVICIOS UMS)**			
notas	8:30 horas	8:30 horas	8:30horas			

AGOSTO2013

	LUNES	MARTES	MIÉRCOLES	JUEVES	VIERNES	SÁB/DOM
SEMANA 1				1 Desarrollo servicios getUser Desarrollo servicio createUser (**SERVICIOS UMS**)	2 Desarrollo servicios getUser Desarrollo servicio createUser (**SERVICIOS UMS**)	3/4
notas				8:30 horas	6 horas	
SEMANA 2	5 VACACIONES	6 VACACIONES	7 VACACIONES	8 VACACIONES	9 VACACIONES	10/11
notas						
SEMANA 3	12 VACACIONES	13 VACACIONES	14 VACACIONES	15 VACACIONES	16 VACACIONES	17/18
notas						
SEMANA 4	19 VACACIONES	20 VACACIONES	21 VACACIONES	22 VACACIONES	23 VACACIONES	24/25
notas						
SEMANA 5	26 (1)Finalización servicios getUser, createUser (2)Solucionar errores LoginPanel, LisUserPanel (3)Desarrollo de servicios modifyUsrer y Modify Group (**SERVICIOS UMS**)	27 (1) Finalización servicios getUser, CreateUser. (2) Desarrollo servicios dropUser y dropGroup (**SERVICIOS UMS**)	28 (1) Finalización servicios dropUser y dropGroup. (**SERVICIOS UMS**) (2) Crear paneles AddUserPanel y AddGroupPanel. Integrar Botonera. **(MODULO USUARIOS)**	29 (1) Desarrollo panel createGroupPanel . (2) Desarrollo e integración pantallas EditGroupPanel, ProfileGroupPanel y ProfileUserPanel	30 Cambios diseño paneles y mejora del código de dichos paneles	31
notas	(1)2 horas (2) 1 hora (3) 5:30 horas	(1)2 horas (2)6:30 horas	(1)3 horas (2) 5:30 horas	(1) 5 horas (2) 3:30 horas	6 horas	

SEPTIEMBRE2013

	LUNES	MARTES	MIÉRCOLES	JUEVES	VIERNES	SÁB/DOM
SEMANA 2	2 Desarrollo paneles almacen de datos, creación de datos, listado de plantillas edicion de almacen de datos. **(MODULO DE DATOS)**	3 (1)Desarrollo servicio getTablesFromConn **(SERVICIOS DMS)** (2) Puesta apunto del portatil nuevo	4 (1)Finalización servicio getTablesFromConn. (2) Desarrollo y finalización servicio createDataTemplate **(SERVICIOS DMS)**	5 EXAMENES TENERIFE	6 EXAMENES TENERIFE	7/8
notas	8:30 horas	(1) 4:horas (2) 4:30 horas	(1) 1:30 horas (2) 7 horas			
SEMANA 3	9 EXAMENES TENERIFE	10 (1)Finalización servicio createDataTemplate **(SERVICIOS DMS)** (2) Cerrar módulo de usuarios, integración de servicios y cambio de pantallas.	11 (1) Cerrar módulo de usuarios, integración de servicios y cambio de pantallas. (2) Modificación servicios createUser,modifyUser (password opcional)	12 (1) Cerrar módulo de usuarios, integración de servicios y cambio de pantallas. (2) Solucionar errores servicios. createUser no cambia el grupo	13 Modificación de servicios para integrar el servicio de búsqueda del módulo de administración.	14/15
notas		(1) 2:30 horas (2) 6 horas	(1)7 horas (1)1:30 horas	(1) 7 horas (2) 1:30 horas	(1) 6 horas	
SEMANA 4	16 Realizar servicios de búsqueda searchUser y searchGroup	17 Realizar servicios de búsqueda searchDataStores searchTemplates	18 REUNION TUTOR PFC	19 Crear servicios DropDataStore y create Data template	20 Creación paneles modulo de datos plantillas.	21/22
notas	8:30 horas	8:30 horas	8:30 horas	8:30 horas	8:30 horas	
SEMANA 5	23 (1) Paneles plantillas de datos (2) Crear servicio getFieldsFromCon	24 (1)Creación paneles modulo geoAsset.	25 (1)Desarrollo paneles módulo de geoAsset. (2) Creación servicios getAcl y setAcl	26 Desarrollo y finalización servicios getAcl y setAcl.	27 Puesta apunto módulo de usuarios	28 29
notas	(1) 4:00 (2) 4:30	8:30 horas	(1)6 horas (2) 2:30	8:30 horas	6 horas	
SEMANA 6	30 **(HITO) Entregar Prototipo Reducido panel de administración**					
notas	8:30 horas					

OCTUBRE2013

	LUNES	MARTES	MIÉRCOLES	JUEVES	VIERNES	SÁB/DOM
SEMANA 1		1 Desarrollo paneles módulo geoAsset	2 Finalización paneles módulo geoAsset	3 Desarrollo panel subida de datos(mejora módulo de datos)	4 Desarrollo servicio uploadData.	5/6
notas		8:30 horas	8:30 horas	8:30 horas	6 horas	
SEMANA 2	7 Desarrollo servicio upload data	8 Desarrollo servicio upload data	9 Desarrollo servicio uploadData.	10 Desarrollo panel subida de datos(mejora módulo de datos)	11 Desarrollo panel subida de datos(mejora módulo de datos)	12/13
notas	8:30 horas	8:30 horas	8:30 horas	8:30 horas	6:00 horas	
SEMANA 3	14 Baja por enfermedad	15 Baja por enfermedad	16 Desarrollo panel subida de datos(mejora módulo de datos)	17 Desarrollo panel subida de datos(mejora módulo de datos)	18 Desarrollo servicio uploadData.	19/20
notas			8:30 horas	8:30 horas	6:00 horas	
SEMANA 4	21 Desarrollo del servicio upload Data.	22 Desarrollo del servicio upload Data	23 **HITO. FInalización del proyecto**	24	25	26/27
notas	8:30 horas	8:30 horas				
SEMANA 5	28	29	30	31		
notas						
SEMANA 6						
notas						