Trabajo Fin de Grado

Grado en Ingeniería de las Tecnologías de Telecomunicación

Diseño y desarrollo de un sistema de información geográfica con interfaz web

Autor: Juan Antonio Villanueva Capel

Tutor: Isabel Román Martínez

Departamento de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2018

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Diseño y desarrollo de un sistema de información geográfica con interfaz web

Autor:
Juan Antonio Villanueva Capel

Tutor:
Isabel Román Martínez
Profesora colaboradora

Departamento de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2018

Trabajo Fin de Grado: Diseño y desarrollo de un sistema de información geográfica con interfaz web

Autor: Juan Antonio Villanueva Capel

Tutor: Isabel Román Martínez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2018

El Secretario del Tribunal

A mis padres

Agradecimientos

A mi padre, por haber estado en las buenas y las malas, haciendo de padre, madre, y de lo que hiciese falta.

A mi madre, por ser siempre una inspiración, motivación y ejemplo de lucha.

A Abel, por haber aguantado a un (futuro) ingeniero.

A los que empezaron el Grado siendo compañeros y lo acabaron siendo amigos, tanto los que aguantaron como los que se quedaron en el camino. A los que compartieron tantos días (e incluso noches) de estudio y trabajo, dentro y fuera de la Escuela.

A todos los profesores que han ido labrando lo que sé y lo que soy. En especial, a Isabel, por haber estado dispuesta a ayudarme con este proyecto, aunque no se lo haya puesto fácil.

Al equipo de Ayesa AT, por descubirme qué era un sistema de información geográfica.

A todos, gracias.

Juan Antonio Villanueva Capel

Sevilla-Madrid, 2018

Resumen

En este proyecto se ha desarrollado un sistema de información geográfica (SIG) con interfaz web. Con él, se pretende que el SIG pueda ser utilizado por multitud de operadores a la vez (al ser una aplicación web) y que tenga una curva de aprendizaje rápida, en comparación a la que presentan otras soluciones existentes en el mercado debido a su complejidad.

En esta memoria, siguiendo el esquema de un proyecto software, se especifican los requisitos, arquitectura, implementación, pruebas y despliegue del mismo.

La aplicación principal está desarrollada en el entorno .NET, utilizándose multitud de tecnologías de distintos ámbitos: AngularJS, Bootstrap, OpenLayers, GeoServer... que se irán contextualizando durante el trabajo.

El sistema de información geográfica implementado está centrado en la creación y el almacenado de información geográfica. Con ello, se pretende que el producto resultante sea un estándar adaptable al ámbito en que se quiera utilizar.

Abstract

In this project, a Geographic Information System (GIS) with a web interface has been developed. The system is expected to be used by a crowd of multiple operators at the same time (due to it being a web application) and it has a quick learning curve, in comparison with the one of other existing solutions in the market (due to its complexity).

In this report, following a software project scheme, the requirements, architecture, implementation, tests, and deployment of the project are specified.

The main application is developed in the .NET environment. A multitude of technologies of different fields are used: AngularJS, Bootstrap, OpenLayers, GeoServer, etc. which will be contextualized within the report.

The implemented geographic information system is focused on the creation and storage of geographical information. In this way, it is expected that the resulting product will become a standard adaptable to the sphere in which it can be required.

Índice

Agradecimientos i

Resumen iii

Abstract v

Índice vi

Índice de Tablas ix

Índice de Figuras xi

1 Introducción 1
1.1. Motivación 1

2 Objetivos del proyecto 3
2.1. Organización del trabajo 3

3 Herramientas a utilizar 5
3.1. Tomcat 5
3.1.1. GeoServer 6
3.2. PostgreSQL 6
3.2.1. PostGIS 6
3.3. Microsoft .NET 7
3.3.1. C# (controlador y modelo) 7
3.3.2. Bootstrap (vista) 7
3.3.3. AngularJS (vista) 7
3.3.4. OpenLayers (vista) 8

4 Especificación de Requisitos de Software 9
4.1. Información del dominio 9
4.1.1. Glosario 9
4.1.2. Mercado actual 10
4.2. Alcance 11
4.3. Relación con sistemas externos 11
4.4. Descripción de subsistemas a desarrollar 12
4.5. Actores del sistema 12
4.6. Requisitos generales 12
4.7. Requisitos funcionales 14
4.7.1. Requisitos de información 14
4.7.2. Casos de uso 19
4.7.3. Requisitos de reglas de negocio 33
4.7.4. Requisitos de conducta 34
4.8. Requisitos no funcionales 44
4.8.1. Requisitos de fiabilidad 44

4.8.2. Requisitos de usabilidad 44
4.8.3. Requisitos de eficiencia 45
4.8.4. Requisitos de mantenibilidad 45
4.8.5. Requisitos de portabilidad 45
4.8.6. Requisitos de seguridad 46

5 Arquitectura 47
5.1. Módulos principales del sistema 48
5.1.1. Autenticación 48
5.1.2. Administración de usuarios 48
5.1.3. Administración geográfica 49
5.1.4. Visor 50
5.2. Módulos auxiliares del sistema 50
5.2.1. Shared 50
5.2.2. Manager de seguridad 50
5.2.3. Database Context 51
5.3. Modelo de datos 51

6 Implementación 53
6.1. Autentificación 53
6.1.1. /Views/Home/Index.cshtml 54
6.1.2. /Controllers/HomeController.cs 54
6.1.3. /Controllers/HomeRestController.cs 54
6.1.4. /Components/home/login/login.js 55
6.1.5. /Components/home/login/templates/login.html 57
6.2. Administración de usuarios 58
6.2.1. /Views/AdminUsu/Index.cshtml 58
6.2.2. /Views/AdminUsu/NuevoUsuario.cshtml 58
6.2.3. /Controllers/AdminUsuController.cs 58
6.2.4. /Controllers/AdminUsuRestController.cs 58
6.2.5. /Components/adminUsu/consultaUsuarios/consultaUsuarios.js 60
6.2.6. /Components/adminUsu/consultaUsuarios/templates/consultaUsuarios.html 62
6.2.7. /Components/adminUsu/nuevoUsuario/nuevoUsuario.js 64
6.2.8. /Components/adminUsu/nuevoUsuario/templates/nuevoUsuario.html 65
6.3. Administración geográfica 65
6.3.1. /Views/AdminGeo/Index.cshtml 66
6.3.2. /Views/AdminGeo/NuevoMapa.cshtml y /Views/AdminGeo/EdicionMapa.cshtml 66
6.3.3. /Views/AdminGeo/NuevaCapa.cshtml y /Views/AdminGeo/EdicionCapa.cshtml 66
6.3.4. /Controllers/AdminGeoController.cs 66
6.3.5. /Controllers/AdminGeoRestController.cs 66
6.3.6. /Components/adminGeo/mapas 71
6.3.7. /Components/adminGeo/nuevoMapa 73
6.3.8. /Components/adminGeo/edicionMapa 75
6.3.9. /Components/adminGeo/capas 76
6.3.10. /Components/adminGeo/nuevaCapa 77
6.3.11. /Components/adminGeo/edicionCapa 80
6.3.12. /Components/adminGeo/usuariosMapas 82
6.4. Visor 83
6.4.1. /Views/Maps/Visor.cshtml 83
6.4.2. /Controllers/MapsController.cs 84
6.4.3. /Controllers/MapsRestController.cs 84
6.4.4. /Components/visor/app.js 85
6.4.5. /Components/visor/identify 87
6.4.6. /Components/visor/layers 88

6.4.7. /Components/visor/measure 88

7 Pruebas 91
7.1. Autentificación 91
7.2. Administración de usuarios 93
7.3. Administración geográfica 97
7.3.1. Mapas 97
7.3.2. Capas 102
7.3.3. Usuarios-mapas 108
7.4. Visor 110

8 Despliegue 113
8.1. Proyecto 113
8.1.1. Publicación 113
8.1.2. Despliegue 114
8.2. Tomcat y Geoserver 117
8.3. PostgreSQL y PostGIS 118

9 Conclusiones 121
9.1. Líneas de mejora 121

Referencias 123

ÍNDICE DE TABLAS

Tabla 1: Comparativa entre ArcGIS y QGIS 10
Tabla 2: RG-001 Administración de usuarios 12
Tabla 3: RG-002 Administración de mapas 13
Tabla 4: RG-003 Gestión de capas 13
Tabla 5: RG-004 Visualización de mapas 14
Tabla 6: RF-INF-001 Metadatos 14
Tabla 7: RF-INF-002 Tipo de polígono 15
Tabla 8: RF-INF-003 Entidades 15
Tabla 9: RF-INF-004 Información de un usuario 16
Tabla 10: RF-INF-005 Información de una capa 17
Tabla 11: RF-INF-006 Información de un mapa 18
Tabla 12: RF-INF-007 Listado de mapas base 18
Tabla 13: RF-INF-008 Información de la leyenda 19
Tabla 14: RF-INF-009 Recurso 19
Tabla 15: RF-CU-001 Autenticación 20
Tabla 16: RF-CU-002 Crear usuario 21
Tabla 17: RF-CU-003 Eliminar usuario 22
Tabla 18: RF-CU-004 Modificar rol de usuario 23
Tabla 19: RF-CU-005 Creación/Edición de mapa 24
Tabla 20: RF-CU-006 Creación/Edición de capa 25
Tabla 21: RF-CU-007 Creación/Edición de capa mediante shapefile 26
Tabla 22: RF-CU-008 Asignación (o desasignación) de mapas a usuarios 27
Tabla 23: RF-CU-009 Inicializar visor 28
Tabla 24: RF-CU-010 Visualizar capas 28
Tabla 25: RF-CU-011 Identificación de una entidad 29
Tabla 26: RF-CU-012 Localizar entidad por metadatos 29
Tabla 27: RF-CU-013 Localizar zona por coordenadas 30
Tabla 28: RF-CU-014 Dibujar y medir 30
Tabla 29: RF-CU-015 Descarga de recursos 31
Tabla 30: RF-CU-016 Añadir sitio predeterminado 31
Tabla 31: RF-CU-017 Ir a sitio predeterminado 32

Tabla 32: RF-CU-018 Imprimir 32
Tabla 33: RF-CU-019 Clusterización de los elementos de una capa 33
Tabla 34: RF-NEG-001 Mapas base de cada mapa 33
Tabla 35: RF-NEG-002 Capas de un mapa 34
Tabla 36: RF-NEG-003 Clusterización de polígonos 34
Tabla 37: RF-CON-001 Autenticación 35
Tabla 38: RF-CON-002 Gestionar usuarios 35
Tabla 39: RF-CON-003 Gestionar mapas 36
Tabla 40: RF-CON-004 Creación y edición de capas 36
Tabla 41: RF-CON-005 Asociación de recursos a entidades 37
Tabla 42: RF-CON-006 Maximizar tamaño del mapa 37
Tabla 43: RF-CON-007 Navegación sobre el mapa 37
Tabla 44: RF-CON-008 Zoom sobre el mapa 38
Tabla 45: RF-CON-009 Barras de herramientas 38
Tabla 46: RF-CON-010 Tabla de contenidos 38
Tabla 47: RF-CON-011 Visualización de leyenda 39
Tabla 48: RF-CON-012 Visualización de escala 39
Tabla 49: RF-CON-013 Posición del cursor 39
Tabla 50: RF-CON-014 Mapa guía 40
Tabla 51: RF-CON-015 Mediciones 40
Tabla 52: RF-CON-016 Seleccionar ubicación según coordenadas 40
Tabla 53: RF-CON-017 Impresión de mapas 41
Tabla 54: RF-CON-018 Exportación a Excel 41
Tabla 55: RF-CON-019 Añadir y consultar favoritos 41
Tabla 56: RF-CON-020 Búsqueda alfanumérica de una entidad 42
Tabla 57: RF-CON-021 Identificación y consulta de entidades 42
Tabla 58: RF-CON-022 Descarga de recursos asociados a la entidad 42
Tabla 59: RF-CON-023 Consulta de elementos de una capa 43
Tabla 60: RF-CON-024 Filtrado de las entidades de una capa 43
Tabla 61: RF-CON-026 Clústers 44
Tabla 62: RNF-FIA-001 Tiempo de recuperación 44
Tabla 63: RNF-USA-001 Facilidad para encontrar la opción deseada 44
Tabla 64: RNF-EFI-001 Rapidez en la respuesta 45
Tabla 65: RNF-MAN-001 Cumplimiento de estándares 45
Tabla 66: RNF-POR-001 Multiplataforma en escritorio 45
Tabla 67: RNF-POR-002 Dispositivos móviles 46
Tabla 68: RNF-SEG-001 Protección ante ataques 46

ÍNDICE DE FIGURAS

Ilustración 1: Tecnologías utilizadas 5
Ilustración 2: Relación con sistemas externos 11
Ilustración 3: Arquitectura MVC 47
Ilustración 4: Modelo de datos 51
Ilustración 5: Módulo de Autentificación 53
Ilustración 6: Clase 'jumbotron' de Bootstrap 54
Ilustración 7: Decodificación de usuario y contraseña 54
Ilustración 8: Alerta de usuario no existente en el sistema 57
Ilustración 9: Módulo de administración de usuarios 58
Ilustración 10: Alerta al tratar de hacer administrador a un usuario con mapas 61
Ilustración 11: Modificación de rol de usuario exitosa 61
Ilustración 12: Alerta para la confirmación del eliminado de un usuario 62
Ilustración 13: Mensaje informativo del eliminado de un usuario 62
Ilustración 14: Eliminado de un usuario exitoso 62
Ilustración 15: Relación entre la vista y los métodos Javascript en el panel de administración de usuarios 63
Ilustración 16: Panel con encabezado, cuerpo y pie utilizando Bootstrap 65
Ilustración 17: Módulo de administración geográfica 65
Ilustración 18: Relación entre la vista y los métodos Javascript en el panel de configuración de mapas 72
Ilustración 19: Alerta para la confirmación del eliminado de un mapa 73
Ilustración 20: Ventana informativa del eliminado de un mapa 73
Ilustración 21: Alerta al tratar de eliminar un mapa relacionado con algún usuario 73
Ilustración 22: Relación entre la vista y los métodos Javascript en el panel de creación de un mapa 74
Ilustración 23: Relación entre la vista y los métodos Javascript en el panel de configuración de capas 76
Ilustración 24: Relación entre la vista y los métodos Javascript en el panel de creación de capas 77
Ilustración 25: Relación entre la vista y los métodos Javascript en el panel de edición de capas 80
Ilustración 26:Relación entre la vista y los métodos Javascript en el panel de configuración de usuarios 82
Ilustración 27: Módulo del visor 83
Ilustración 28: Módulo visor antes de seleccionar ningún mapa 84
Ilustración 29: Herramienta para identificar elementos del visor 87
Ilustración 30: Herramienta para gestionar las capas visibles en el visor 88
Ilustración 31: Herramienta para realizar medidas en el visor 88

Ilustración 32: Prueba 1 de Autenticación correcta 91
Ilustración 33: Prueba 2 de Autenticación correcta 92
Ilustración 34: Prueba de Autenticación errónea 93
Ilustración 35: Prueba de Cambiar el rol de un usuario 94
Ilustración 36: Prueba de Cambiar el rol de un usuario con mapas asignados 95
Ilustración 37: Prueba de Eliminar un usuario 95
Ilustración 38: Prueba de Crear un usuario exitosamente 96
Ilustración 39: Prueba de Crear un usuario con campos sin rellenar 96
Ilustración 40: Prueba de Filtrado según metadatos del mapa 97
Ilustración 41: Prueba de Nombres únicos de mapas 97
Ilustración 42: Prueba de creación de mapa exitosa 98
Ilustración 43: Prueba de Creación de mapa con campos sin rellenar 98
Ilustración 44: Prueba de Edición de mapa con campos sin rellenar 99
Ilustración 45: Prueba de Edición de mapa sin capas seleccionadas 99
Ilustración 46: Prueba de Edición de mapa exitosa 100
Ilustración 47: Prueba de Eliminar mapa con usuario vinculado 101
Ilustración 48: Prueba de Eliminar mapa con capa vinculada 102
Ilustración 49: Prueba de Nombres únicos de capas 102
Ilustración 50: Prueba de Creación de capa exitosa 103
Ilustración 51: Prueba de Creación de capa con campos sin rellenar 103
Ilustración 52: Prueba de Comprobación de la creación de los elementos de la nueva capa 104
Ilustración 53: Prueba de Edición de capa con campos sin rellenar 105
Ilustración 54: Prueba de Edición de capa con nombre repetido 105
Ilustración 55: Prueba de Edición de capa exitosa 106
Ilustración 56: Prueba de Edición de datos de un elemento 106
Ilustración 57: Prueba de Eliminar una capa ligada a un mapa 107
Ilustración 58: Prueba de Eliminar una capa sin relaciones 107
Ilustración 59: Prueba de Añadir un mapa a un usuario 108
Ilustración 60: Prueba de Añadir un mapa a un usuario que ya lo tiene asignado 108
Ilustración 61: Prueba de Eliminar un mapa de un usuario 109
Ilustración 62: Prueba de Listado de usuarios 109
Ilustración 63: Prueba de Listado de mapas 110
Ilustración 64: Prueba de Apertura del mapa correspondiente 110
Ilustración 65: Prueba de Identificación del mapa 111
Ilustración 66: Prueba de visibilidad de capas 112
Ilustración 67: Prueba de Medición de distancias 112
Ilustración 68: Despliegue 113
Ilustración 69: Publicar proyecto en Visual Studio 114
Ilustración 70: Perfil de carpeta para publicar proyecto 114

Ilustración 71: Instalación de servidor ISS 115
Ilustración 72: Grupos de aplicaciones IIS 115
Ilustración 73: Directorio para desplegar el proyecto en el servidor 116
Ilustración 74: Proyecto desplegado 116
Ilustración 75: Variables de entorno Java para utilizar Tomcar 117
Ilustración 76: GeoServer desplegado 118
Ilustración 77: Base de datos 'TFG' en PGAdmin 119

1 INTRODUCCIÓN

"Ni el huevo ni la gallina, lo primero es una buena introducción"

- Adaptación de Cafés Delikia -

Para entender qué es un sistema de información geográfica (SIG), resulta útil desglosar el término SIG, y así definir los conceptos de *sistema de información* e *información geográfica*.

La información geográfica (IG) es información sobre un elemento en la superficie de la Tierra, es el conocimiento sobre "dónde" hay algo o "qué hay" en un determinado lugar.

Por sistema de información (SI) se entiende el uso de herramientas informáticas (programas o softwares) para la gestión y el análisis de información con unos objetivos concretos [Introducción_1]. Un sistema de información se usa para manipular, consultar, editar, visualizar… información almacenada, generalmente, en una base de datos.

Con la suma de estos términos, ya que SI+IG=SIG, podemos definir un SIG como un sistema de hardware, software, datos, organizaciones y convenios institucionales para la recopilación, almacenamiento, análisis y distribución de información de territorios de la Tierra [Introducción_2]. Estos sistemas pueden facilitar la incorporación de aspectos socio-culturales, económicos y ambientales que conducen a la toma de decisiones de una manera más eficaz.

1.1. Motivación

Existe un gran número de softwares SIG en el mercado, pero todos cuentan con una curva de aprendizaje elevada o están formados por una multitud de herramientas complejas.

Por ello, la finalidad de este proyecto será desarrollar un sistema de información geográfica con interfaz web y de fácil utilización, el cual permita:

- Crear, editar y asignar mapas a los administradores del sistema.
- Observar mapas, junto a un catálogo de herramientas (coordenadas, escalas…) para dar mayor utilidad a su visualización, a los usuarios de éste.

Se tratará de usar una pila tecnológica de código libre para facilitar la implementación del sistema por parte de cualquier empresa y, a su vez, para facilitar su desarrollo, mantenimiento y extensión.

Desde una perspectiva académica, se tratará de plasmar en un proyecto propio algunos de los conocimientos adquiridos durante mis prácticas de empresa, con lenguajes y plataformas que no he utilizado durante el Grado.

2 OBJETIVOS DEL PROYECTO

"Si no sabes dónde vas, acabarás en otra parte"

- Laurence J. Peter-

Como ya se ha introducido, el objetivo de este proyecto puede sintetizarse en que se quiere desarrollar un sistema de información geográfica con la capacidad de crear, editar, asignar y observar mapas.

El producto que se obtendrá de este proyecto será una aplicación web. Esta aplicación podrá ser usada simultáneamente por distintos usuarios, y el administrador podrá cambiar dinámicamente el contenido que verán los mismos. El diseño web que se utilizará será, en un principio, adaptable, pero no se tomará como una prioridad que sea usable en un dispositivo móvil. Se buscará una interfaz lo más clara y liviana posible, y en la que sea necesario el menor número de clicks posibles.

En el lado de un administrador del sistema, la configuración de mapas y usuarios se realizará mediante distintos tipos de menús, según el tipo de información que se maneje. Por ejemplo, para elegir el rol de un usuario (dos opciones), resulta práctico un desplegable junto a cada uno de ellos en la misma ventana que muestre el listado de usuarios. En cambio, para elegir las capas de un mapa (serán muy numerosas), se presentará una nueva ventana con el listado de todas las posibles.

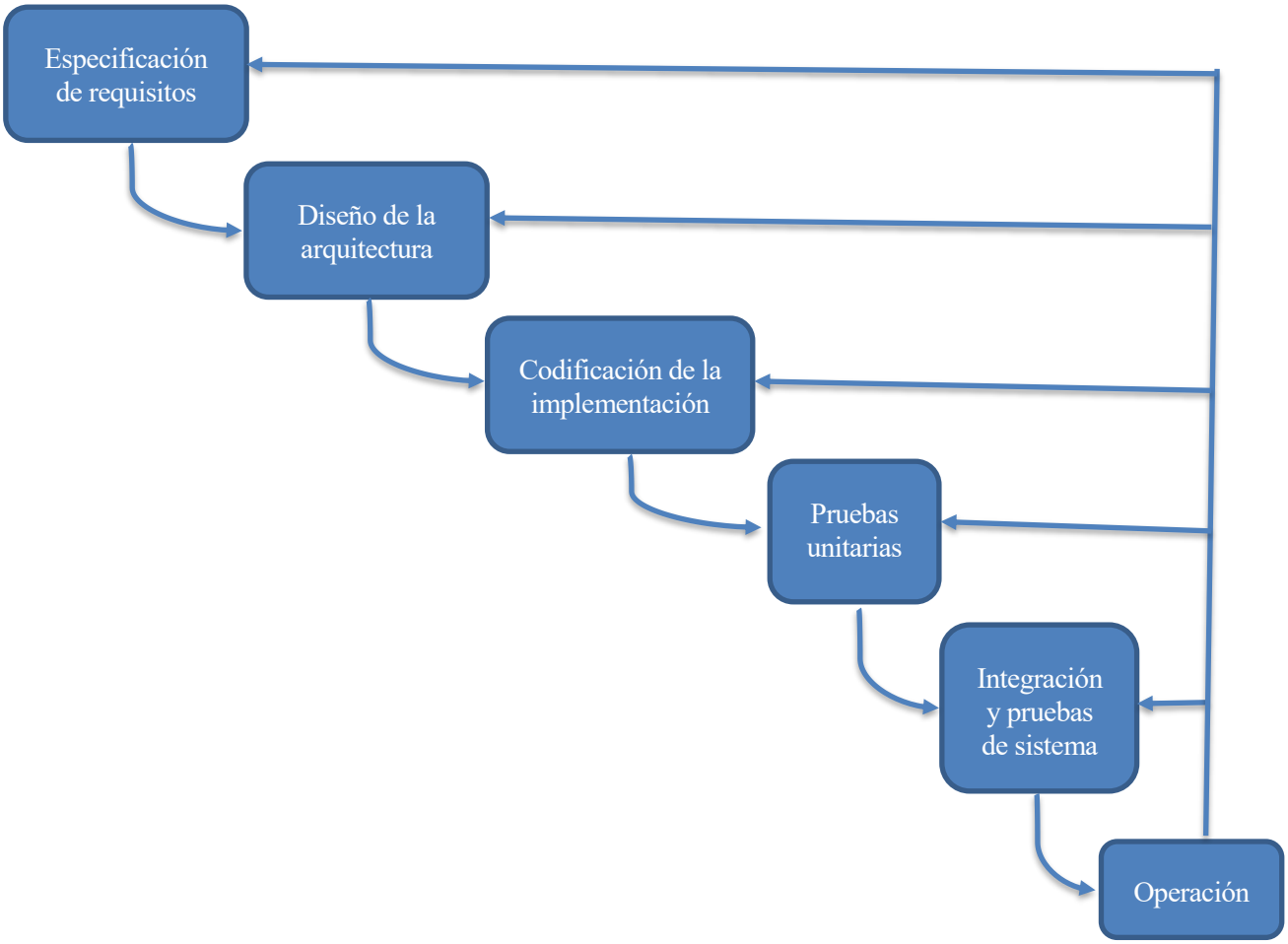
En cuanto al puesto de un usuario, el catálogo de herramientas disponibles en la visualización del mapa, aportarán facilidades para trabajar con él. Por ejemplo, será posible medir distancias, encontrar algún elemento según sus metadatos... También contará con funciones básicas y típicas de un mapa, como hacer zoom.

El alcance del proyecto será más amplio en el campo de la administración y el almacenamiento de la información geográfica que en su análisis, con el objetivo de que el sistema desarrollado pueda ser utilizado como estándar, ya que las herramientas necesarias para analizar un conjunto de información geográfica dependen sobremanera del ámbito en el que se quiere utilizar el SIG.

2.1. Organización del trabajo

Este trabajo está planteado como un proyecto software. De entre los posibles enfoques, se ha elegido un modelo de proceso clásico o en cascada.

En éste modelo, una fase debe completarse antes de pasar a la siguiente, pero se necesitan retornos para incorporar correcciones. Cada una de estas fases es, aproximadamente, un apartado de esta memoria [Objetivos_1].



Así, una especificación de requisitos formal será la que determine el alcance y diseño del sistema, implementándose la arquitectura diseñada y, finalmente, comprobando mediante las pruebas unitarias y del sistema que los requisitos especificados efectivamente se cumplen.

3 HERRAMIENTAS A UTILIZAR

"No tengo miedo a los ordenadores. A lo que tengo miedo es a la falta de ellos"

- Isaac Asimov -

Para la elaboración de este proyecto se utilizarán varias tecnologías que estarán relacionadas. Aunque será en el apartado de implementación en el que se explique con mayor detalle cómo se utiliza cada una de ellas, se explicarán ahora sus funcionalidades y cómo se relacionan entre ellas.

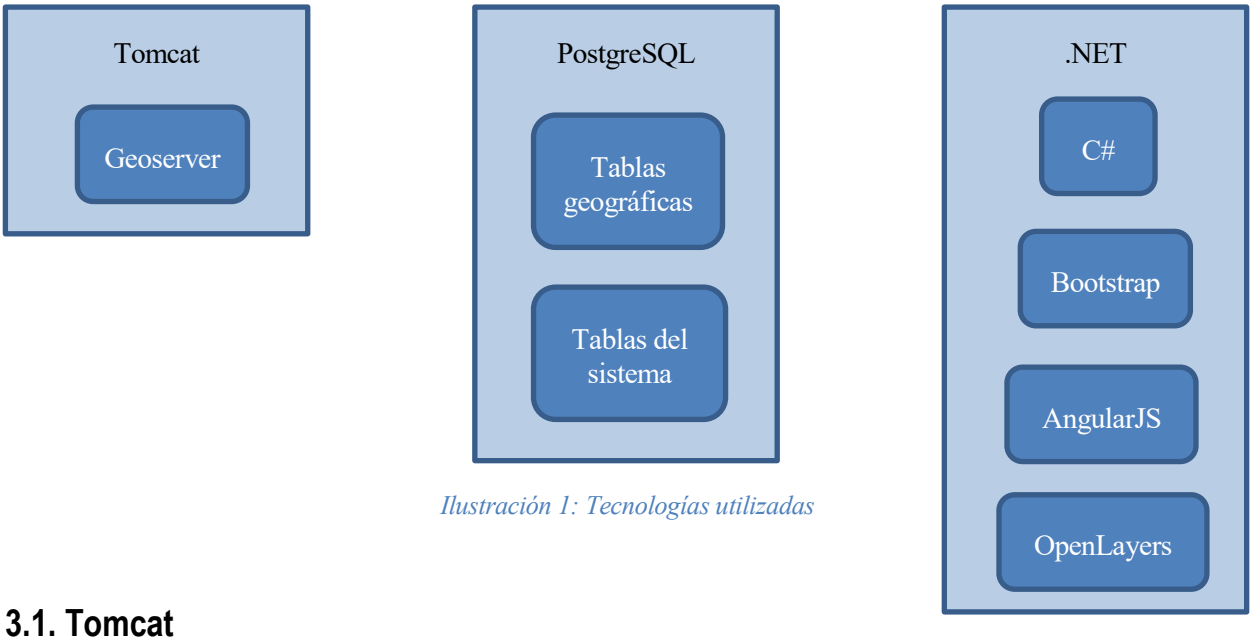


Ilustración 1: Tecnologías utilizadas

3.1. Tomcat

El software Apache Tomcat es una implementación de código abierto de las tecnologías Java Servlet, JavaServer Pages, Java Expression Language y Java WebSocket [Herramientas_1].

Funciona como un contenedor de servlets. Un servlet es una clase Java que se ejecuta dentro de un servidor Web. Los servlets reciben y responden a peticiones realizadas desde un cliente Web, habitualmente mediante HTTP, el Protocolo de Transferencia de HiperTexto [Herramientas_2].

Una aplicación web puede empaquetarse en un archivo WAR (de Web Application Archive - Archivo de aplicación web). Este archivo es un JAR (Java ARchive) utilizado para distribuir una colección de JavaServer Pages, servlets, clases Java, archivos XML, bibliotecas de tags y páginas web estáticas (HTML y archivos relacionados) que conforman la aplicación [Herramientas_3].

Este archivo se incluye en el directorio de Tomcat que contiene las aplicaciones web del contenedor.

3.1.1. GeoServer

El fichero WAR en que se empaqueta GeoServer será desplegada en el servidor Tomcat.

GeoServer es un servidor basado en Java que permite a los usuarios ver y editar datos geoespaciales. Mediante el uso de estándares abiertos establecidos por el Open Geospatial Consortium (OGC), GeoServer permite una gran flexibilidad en la creación de mapas y el intercambio de datos [Herramientas_4].

Al implementar el estándar del Web Map Service (WMS), GeoServer puede crear mapas en una variedad de formatos de salida. OpenLayers, una biblioteca de mapas gratuita, está integrada en GeoServer, lo que hace que la generación de mapas sea rápida y fácil. GeoServer está basado en Geotools, un juego de herramientas Open Source Git de Java [Herramientas_4].

GeoServer también se ajusta al estándar del Web Feature Service (WFS), que permite compartir y editar los datos que se utilizan para generar los mapas. Otros pueden incorporar sus datos en sus sitios web y aplicaciones, liberando sus datos y permitiendo una mayor transparencia.

GeoServer es software libre, al contrario que los productos SIG tradicionales, y es también código abierto. Las correcciones de errores y las mejoras de características en el software de código abierto se aceleran mucho en comparación con las soluciones de software tradicionales.

GeoServer puede mostrar datos en cualquiera de las populares aplicaciones de mapas como Google Maps, Google Earth, Yahoo Maps y Microsoft Virtual Earth. Además, GeoServer puede conectarse con arquitecturas GIS tradicionales como ESRI ArcGIS.

3.2. PostgreSQL

PostgreSQL es un sistema de base de datos relacional de código abierto con más de 15 años de desarrollo activo [Herramientas_5].

Se ejecuta en todos los principales sistemas operativos, incluidos Linux, UNIX (AIX, BSD, HP-UX, macOS, Solaris) y Windows. Es totalmente compatible con ACID, tiene soporte completo para claves externas, combinaciones, vistas, disparadores y procedimientos almacenados (en varios idiomas). Incluye la mayoría de los tipos de datos SQL: 2008, incluidos INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL y TIMESTAMP. También es compatible con el almacenamiento de objetos grandes binarios, incluyendo imágenes, sonidos o video. Tiene interfaces de programación nativas para C / C ++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, entre otros.

Cuenta con funciones de ámbito empresarial como Multi-Version Concurrency Control (MVCC), recuperación puntual, tablespaces, replicación asíncrona, transacciones anidadas (puntos de rescate), copias de seguridad en línea/calientes y planificador/optimizador de consultas. Admite conjuntos de caracteres internacionales, codificaciones de caracteres multibyte, Unicode, y es compatible con la configuración regional para la clasificación, la distinción entre mayúsculas y minúsculas y el formateo.

Es altamente escalable tanto en la gran cantidad de datos que puede administrar como en la cantidad de usuarios simultáneos que puede acomodar, existiendo entornos que gestionan terabytes y petabytes de datos.

3.2.1. PostGIS

En cuanto a las tablas geográficas del sistema, son compatibles con el sistema mediante PostGIS.

PostGIS es un expansor de base de datos espacial para la base de datos relacional de objetos PostgreSQL. Agrega soporte para objetos geográficos que permiten que las consultas de ubicación se ejecuten en SQL [Herramientas_6].

3.3. Microsoft .NET

Microsoft .NET se trata de una plataforma para el desarrollo de software lanzada por Microsoft con la finalidad de fusionar su catálogo de productos, que va desde sus múltiples sistemas operativos hasta herramientas de desarrollo [Herramientas_7].

Desde un punto de vista tecnológico, con la creación de .NET se pretende desarrollar aplicaciones y sistemas independientes de la arquitectura física y del sistema operativo sobre el que se ejecutarán.

Desde el punto de vista comercial, .NET podría interpretarse como la alternativa por parte de Microsoft en el sector de los desarrollos web para competir con la plataforma Java de Oracle Corporation, así como con los diversos entornos de trabajo basados en PHP.

Al tratarse de una plataforma de propósito general, se pueden realizar tanto desarrollos web, como programas de escritorio o aplicaciones para dispositivos móviles. Además, favorece el desarrollo en multiplataforma, como por ejemplo el que una misma aplicación pueda correr indistintamente en los diferentes sistemas operativos móviles como iOS, Android o Windows Phone, garantizando la comunicación entre los diferentes dispositivos.

La construcción de aplicaciones usando .NET permite disminuir el tiempo de desarrollo de los proyectos, utilizar las funcionalidades ya diseñadas para realizar un proyecto de importación y adaptación a las necesidades en lugar de un desarrollo desde cero o simplificar el mantenimiento de las aplicaciones desarrolladas en esta plataforma.

Es muy común decir que una aplicación está desarrollada en .NET, pero lo correcto sería decir que está construida sobre .NET usando alguno de los lenguajes de programación contenidos en la propia plataforma como son C#, C++, Visual Basic .NET o F#.

Dentro de las posibilidades de la plataforma .NET, se utiliza el framework ASP.NET MVC, que implementa el clásico patrón arquitectónico Modelo-Vista-Controlador.

3.3.1. C# (controlador y modelo)

C# es un lenguaje de programación con seguridad de tipos y orientado a objetos, que permite a los desarrolladores crear una gran variedad de aplicaciones seguras y sólidas que se ejecutan en .NET Framework. Puede usarse para crear aplicaciones cliente de Windows, servicios web XML, componentes distribuidos, aplicaciones cliente-servidor o aplicaciones de base de datos, entre otros [Herramientas_8].

En este proyecto, los controladores y la definición de los modelos estarán en este lenguaje.

3.3.2. Bootstrap (vista)

Bootstrap es una biblioteca de componentes front-end para diseñar proyectos adaptables, facilitando la tarea respecto de una configuración únicamente con CSS [Herramientas_9].

Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS.

3.3.3. AngularJS (vista)

AngularJS es un framework de JavaScript de código abierto client-side (lado del cliente) que promueve una experiencia de desarrollo web de alta productividad.

Fue construido con la creencia de que la programación declarativa (desarrollo de programas "declarando" un conjunto de condiciones, proposiciones, afirmaciones... que describen el problema y detallan su solución, sin especificar cómo encontrarla) es la mejor opción para construir la interfaz de usuario, mientras que la programación imperativa (conjunto de instrucciones que le indican al computador cómo realizar una tarea) es preferible para implementar la lógica de negocio de una aplicación.

Para lograr esto, AngularJS potencia el HTML tradicional ampliando su vocabulario actual, facilitando la vida de los desarrolladores.

El resultado es el desarrollo de componentes de aplicación expresivos, reutilizables y mantenibles, dejando atrás un montón de código innecesario.

Una aplicación AngularJS típica consiste principalmente en una vista, modelo y controlador. Se apoya en una serie de componentes importantes como servicios, directivas o filtros.

La vista está escrita completamente en HTML, y aprovecha el mecanismo de directivas, una extensión del vocabulario HTML que brinda la capacidad de realizar tareas de lenguaje de programación, como iterar sobre una matriz o evaluar una expresión de forma condicional.

Detrás de la vista, está el controlador. En el caso de este proyecto, las tareas características del controlador se realizan en C#, el propio de Angular solo se declara para poder utilizar las ventajas del scope. El scope es un objeto compartido que conecta de manera sencilla, utilizándose para intercambiar información relacionada con el modelo [Herramientas_10].

3.3.4. OpenLayers (vista)

OpenLayers es una biblioteca de JavaScript de código abierto 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 [Herramientas_11].

4 ESPECIFICACIÓN DE REQUISITOS DE SOFTWARE

"Caminar sobre el agua y desarrollar software en base a una especificación es fácil, si ambos elementos están congelados"

- Edward V. Berard-

Mediante la ingeniería de requisitos, se comprende qué desea el cliente de un software, analizando sus necesidades y la factibilidad de cumplirlas. Estos requisitos han de estar expresados sin ambigüedades y de forma que sea posible comprobar, objetivamente, si han sido logrados [ERS_1].

4.1. Información del dominio

Como ya se ha introducido qué es un sistema de información geográfica en el primer apartado de la memoria, se procede directamente a explicar algunos conceptos relevantes y específicos del dominio y a valorar las soluciones ya existentes en el mercado.

4.1.1. Glosario

Para comprender los requisitos de este proyecto, es necesario aclarar algunos conceptos propios del dominio de los sistemas de información geográfica.

- **Visor** Pantalla desde la que un usuario accede a la información geográfica de forma cartográfica (sobre un mapa). La mayor parte de la pantalla es el propio mapa, y en los laterales se suelen incluir barras de **herramientas** con distintas utilidades que facilitan el análisis de la información, o permiten seleccionar qué información es la que se quiere ver.
- **Entidad** Es la unidad básica de un mapa, del dato que se quiere modelar. Para representarlo, se utilizan tres elementos geométricos: el punto, la línea y el polígono.
- **Capa** Las capas son el mecanismo que se utiliza para visualizar un conjunto de datos geográficos del mismo tipo. En ella se hace referencia al conjunto de datos y a cómo han de representarse.
- **Mapa** Representación gráfica obtenida al ensamblar un conjunto de capas.
- **Mapa base** Capa "inferior" de cualquier mapa, en la que se representa la superficie del terreno. Actúa como lienzo del mapa, es decir, no consta de información dinámica como el resto de capas.
- **Clúster** En un SIG, un clúster es una agrupación de varias entidades, cuando el nivel de zoom no es lo suficientemente alto como para diferenciar uno de otro. En el clúster aparecerá un número, el número de entidades que se están representando con él, número que irá descendiendo al ir aproximando el zoom.
- **Shapefile** Fichero utilizado para almacenar la ubicación geométrica y la información de atributos de las entidades geográficas [ERS_2]. Las entidades geográficas de un shapefile se pueden representar, como las entidades creadas en nuestro propio sistema, por medio de puntos, líneas o polígonos.

- **Modelos vectoriales vs ráster** Los modelos vectoriales se representan mediante figuras geométricas (puntos, líneas, polígonos), mientras que los ráster se tratan de una imagen dividida en celdillas en la que cada una tiene información. En este proyecto, solo se trabajará con modelos vectoriales.

4.1.2. Mercado actual

A continuación, se presentan las dos soluciones más relevantes del mercado de los SIG, siendo una software propietario y la otra libre.

4.1.2.1. ArcGIS

La plataforma ArcGIS es un conjunto de productos de software en el campo de los Sistemas de Información Geográfica producida y comercializada por la empresa ESRI.

Existen distintas aplicaciones para la captura, edición, análisis, tratamiento, diseño, publicación e impresión de información geográfica. Estas aplicaciones se engloban en familias temáticas como ArcGIS Server, para la publicación y gestión web, o ArcGIS Móvil para la captura y gestión de información en campo.

Una de las más ampliamente utilizadas es ArcGIS Desktop, probablemente el software GIS de escritorio más importante, tomándose incluso como estándar de facto.

ArcGIS Desktop permite visualizar y administrar los datos, realizar análisis avanzados, crear modelos y automatizar los flujos de trabajo, mostrando sus resultados en mapas profesionales [ERS_3].

4.1.2.2. QGIS

QGIS es un SIG de código libre para plataformas GNU/Linux, Unix, Mac OS, Microsoft Windows y Android. Tiene soporte para la extensión espacial de PostgreSQL (PostGIS), y maneja tanto archivos vectoriales (como Shapefile) como archivos ráster (varios tipos) [ERS_4].

4.1.2.3. Comparativa

	ArcGIS [ERS_5] [ERS_6]	QGIS [ERS_5] [ERS_6]
Documentación	Muy elaborada	Insuficiente
Sistema operativo	Windows	Multiplataforma
Licencia	Según el nivel, se acceden a más herramientas de geoprocesamiento	No hay
Plugins	Soluciones para cada cosa, más fácil encontrar especialización	Muchos, desarrollados por usuarios
Consumo de datos	Pocos y específicos	Infinitad de formatos ráster y vectorial
Simplicidad	Un único botón de “Añadir dato”	Un botón por cada tipo de dato
Catálogo de datos reales	Inmenso	Poco, con el plugin de OpenLayers
Herramientas para geoestadísticas	Muy guiadas	Difíciles de usar si no se conoce cómo
Edición de topologías con errores	Corrige facilmente posibles errores	Tiene herramientas de validar, pero hay que corregirlas a mano

Tabla 1: Comparativa entre ArcGIS y QGIS

4.2. Alcance

Como ya se ha introducido, la finalidad de este proyecto será desarrollar un sistema de información geográfica.

Un SIG integral, que ya estuviera listo para ser utilizado en las labores de una empresa, estaría dedicado a alguna función específica, a aquella del dominio de la empresa. Las herramientas del visor, las opciones para la creación de capas y mapas... estarían adaptadas a él.

En el caso de este proyecto, no se llevará a cabo esta concreción. La atención se centra en desarrollar una herramienta global y genérica, que serviría como base a ese software integral.

Así, es posible precisar el alcance en dos módulos:

- Un módulo de gestión de información (usuarios, características a las que puede acceder cada usuario, relaciones para agrupar elementos...), que podría ser válido también para un sistema que no fuera de información geográfica.
- Un módulo específico de información geográfica, en el cual se puedan crear y editar capas pero cuyo visor solo cuente con unas herramientas básicas (se concretan en la ERS), no incluyendo otras más específicas que dependerían de cuál fuera la aplicación final del software.

4.3. Relación con sistemas externos

No todas las herramientas introducidas en el tercer apartado forman parte del sistema a desarrollar, sino que son sistemas externos en los que será necesario apoyarse.

Para explicar estas relaciones, se recupera el gráfico de ese apartado, señalando en rojo los sistemas externos.

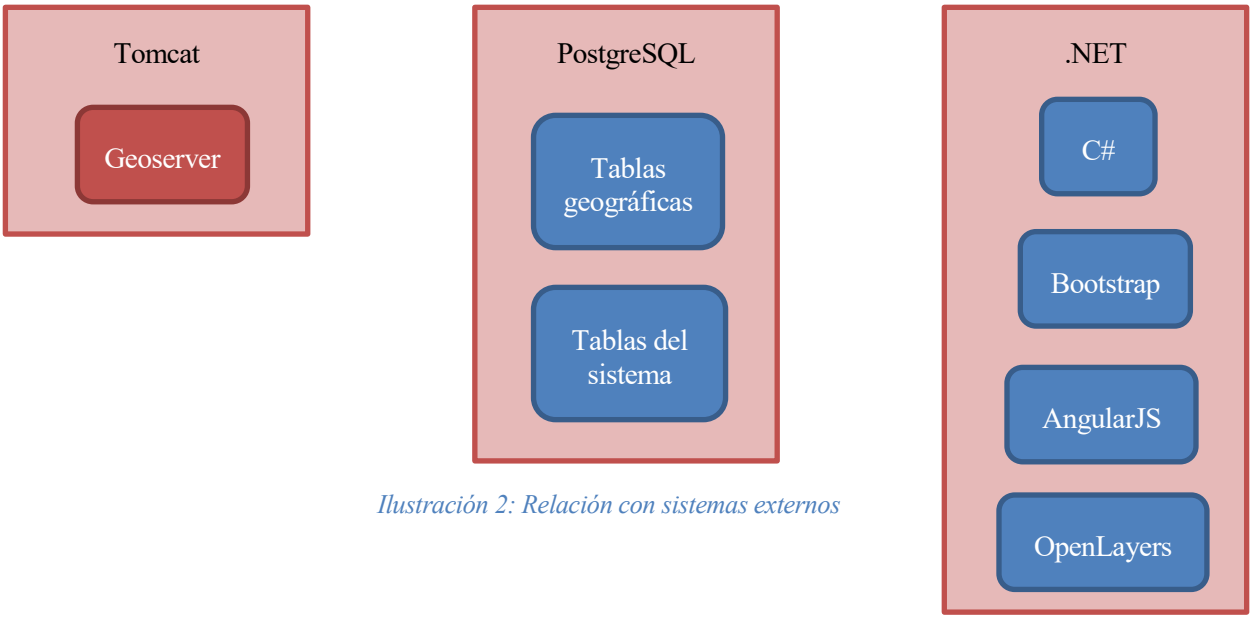


Ilustración 2: Relación con sistemas externos

Además, al margen del alcance de este proyecto, también podrían ser sistemas externos alguna de las herramientas que se utilizaran para especializar el sistema.

4.4. Descripción de subsistemas a desarrollar

Es posible dividir el software a desarrollar en tres subsistemas independientes:

- Administración de usuarios: Subsistema cuya función es administrar a los usuarios del sofware. Otorga la capacidad de, por un lado, añadir y eliminar usuarios y, por otro, modificar el rol de cada uno de ellos.
- Administración geográfica: Su función es administrar la información geográfica del sistema. En él, es posible crear, eliminar o editar tanto mapas como capas, y asignar qué mapas podrá consultar cada usuario.
- Visor: Subsistema en el que se visualizarán los mapas que los administradores hayan asignado. No se podrá modificar el sistema, únicamente consultarlo (con la ayuda de distintas herramientas).

4.5. Actores del sistema

Se cuenta con tres tipos de actores en el sistema.

Cada uno de estos actores se identifica con cada uno de los subsistemas.

- Administrador de usuarios: Solo puede haber uno en todo el sistema. Tiene la capacidad de acceder al módulo de "administración de usuarios". Está creado desde el inicio, no siendo posible editarlo ni eliminarlo. Su alias es "webmaster".
- Administradores geográficos: Puede haber varios. Pueden acceder tanto al módulo de "administración geográfica" como al "visor".
- Usuarios: Es el rol más numeroso. Pueden acceder al "visor", encontrando en él la información que los administradores de mapas hayan determinado.

4.6. Requisitos generales

Los requisitos generales describen a alto nivel las características básicas del sistema, que luego se detallarán mediante los otros tipos de requisitos.

RG-001	Administración de usuarios
Versión	2.0
Dependencias	RF-INF-001 Información de un usuario RF-CU-001 Autenticacion RF-CU-002 Crear usuario RF-CU-003 Eliminar usuario RF-CU-004 Modificar rol de usuario RF-CU-008 Asignación de mapas a usuarios RF-CON-001 Autenticación RF-CON-002 Gestionar usuarios
Descripción	El sistema deberá ofrecer un panel web desde el que gestionar a sus usuarios.
Prioridad	Alta

Tabla 2: RG-001 Administración de usuarios

RG-002	Administración de mapas
Versión	2.0
Dependencias	RF-INF-004 Información de un mapa RF-INF-005 Listado de mapas base RF-CU-005 Creación/Edición de mapa RF-CU-008 Asignación de mapas a usuarios RF-NEG-001 Mapa base de cada mapa RF-NEG-002 Capas de un mapa RF-CON-003 Gestionar mapas
Descripción	El sistema dispondrá de un sitio web desde el que administrar sus mapas.
Prioridad	Alta

Tabla 3: RG-002 Administración de mapas

RG-003	Gestión de capas
Versión	2.0
Dependencias	RF-INF-002 Información de una capa RF-INF-003 Información de una entidad RF-INF-007 Recurso RF-CU-006 Creación/Edición de capa RF-CU-007 Creación/Edición de capa mediante shapefile RF-CON-004 Creación y edición de capas RF-CON-005 Asociación de recursos a entidades
Descripción	El sistema tendrá una interfaz en la que gestionar las capas (y las entidades que las forman).
Prioridad	Alta

Tabla 4: RG-003 Gestión de capas

RG-004	Visualización de mapas
Versión	2.0
Dependencias	RF-INF-006 Información de la leyenda RF-CU-009 ... RF-CU-019 Sucesión de casos de uso del visor de mapas RF-NEG-003 Clusterización de polígonos RF-CON-007 ... RF-CON-26 Sucesión de requisitos de conducta del visor
Descripción	El sistema tendrá un visor en el que consultar los mapas de cada usuario.
Prioridad	Alta
RG-005	Personalización del visor
Versión	2.0
Dependencias	RF-CON-006 Maximizar tamaño del mapa
Descripción	Deberá ser posible mostrar u ocultar las distintas herramientas del visor, siendo todos los menús del mismo colapsables.
Prioridad	Alta

Tabla 5: RG-004 Visualización de mapas

4.7. Requisitos funcionales

Los requisitos funcionales definen las funcionalidades o capacidades que debe ofrecer el sistema a los usuarios para alcanzar sus objetivos de su negocio.

4.7.1. Requisitos de información

Se presentan en sucesivas tablas los distintos requisitos de información del sistema.

RF-INF-001	Metadatos
Versión	3.0
Dependencias	RF-INF-002 Información de una capa RF-INF-004 Información de un mapa
Descripción	Campo de información en el que se podrá incluir metadatos que faciliten identificar y buscar la capa o el mapa deseado
Campos de la información	• Texto libre
Prioridad	Alta

Tabla 6: RF-INF-001 Metadatos

RF-INF-002	Tipo de polígono
Versión	3.0
Dependencias	RF-INF-002 Información de una capa
Descripción	Tipo de figura geométrica que tendrán las entidades de una capa
Campos de la información	A elegir entre: • Puntos • Líneas • Círculos
Prioridad	Alta

Tabla 7: RF-INF-002 Tipo de polígono

RF-INF-003	Entidades
Versión	2.0
Dependencias	RG-003 Gestión de capas RF-INF-007 Recurso RF-CU-006 Creación/Edición de capa RF-CU-007 Creación/Edición de capa mediante shapefile RF-CU-011 Identificación de una entidad RF-CU-012 Localizar entidad por metadatos RF-CON-004 Creación y edición de capas RF-CON-024 Filtrado de las entidades de una capa
Descripción	El sistema deberá almacenar la información correspondiente a cada una de las entidades de una capa.
Campos de la información	• Identificador • Alias • ... (al crear una capa, el usuario del sistema podrá elegir el nombre y la cantidad de los campos de las entidades de la capa)
Prioridad	Alta

Tabla 8: RF-INF-003 Entidades

RF-INF-004	Información de un usuario
Versión	2.0
Dependencias	RG-001 Administración de usuarios RF-CU-001 Autenticación RF-CU-002 Crear usuario RF-CU-003 Eliminar usuario RF-CU-004 Modificar rol de usuario RF-CON-001 Autenticación RF-CON-002 Gestionar usuarios
Descripción	El sistema deberá almacenar una información básica de cada usuario registrado en él.
Campos de la información	• Identificador • Alias • Rol • Contraseña
Prioridad	Alta

Tabla 9: RF-INF-004 Información de un usuario

RF-INF-005	Información de una capa
Versión	2.0
Dependencias	RG-003 Gestión de capas RF-CU-006 Creación/Edición de capa RF-CU-007 Creación/Edición de capa mediante shapefile RF-CU-010 Visualizar capas RF-CON-004 Creación y edición de capas RF-CON-010 Tabla de contenidos RF-CON-023 Consulta de elementos de una capa RF-CON-024 Filtrado de las entidades de una capa
Descripción	El sistema deberá almacenar la información correspondiente a cada capa.
Campos de la información	• Identificador • Nombre • Metadatos • Tipo de polígono • Identificador de las entidades que lo conforman
Prioridad	Alta

Tabla 10: RF-INF-005 Información de una capa

RF-INF-006	Información de un mapa
Versión	2.0
Dependencias	RG-002 Administración de mapas RF-INF-005 Listado de mapas base RF-CU-005 Creación/Edición de mapa RF-CU-008 Asignación de mapas a usuarios RF-CU-010 Visualizar capas RF-NEG-001 Mapa base de cada mapa RF-NEG-002 Capas de un mapa RF-CON-003 Gestionar mapas RF-CON-010 Tabla de contenidos
Descripción	El sistema deberá almacenar información de cada mapa existente en él.
Campos de la información	• Identificador • Nombre • Metadatos • Identificador de su mapa base • Identificador de las capas que lo conforman • Identificador de los usuarios que pueden visualizarlo
Prioridad	Alta

Tabla 11: RF-INF-006 Información de un mapa

RF-INF-007	Listado de mapas base
Versión	2.0
Dependencias	RG-002 Administración de mapas RF-INF-004 Información de un mapa RF-CU-005 Creación/Edición de mapa RF-NEG-001 Mapa base de cada mapa RF-CON-003 Gestionar mapas
Descripción	El sistema deberá almacenar el listado de posibles mapas base para un mapa.
Campos de la información	• Nombre • Metadatos • Fuente
Prioridad	Alta

Tabla 12: RF-INF-007 Listado de mapas base

RF-INF-008	Información de la leyenda
Versión	2.0
Dependencias	RG-004 Visualización de mapas RF-CON-011 Visualización de leyenda
Descripción	En cada mapa, se deberá mostrar una leyenda que indique qué tipo de información representa cada dato.
Campos de la información	Conjunto de pares símbolo-descripción de cada símbolo (de cada entidad o conjunto de entidades) que aparezca en el mapa.
Prioridad	Alta

Tabla 13: RF-INF-008 Información de la leyenda

RF-INF-009	Recurso
Versión	2.0
Dependencias	RG-003 Gestión de capas RF-INF-003 Información de una entidad RF-CU-006 Creación/Edición de capa RF-CU-015 Descarga de recursos RF-CON-005 Asociación de recursos a entidades RF-CON-020 Búsqueda alfanumérica de una ciudad RF-CON-022 Descarga de recursos asociados a la entidad
Descripción	Una entidad podrá tener adjunto algún recurso en forma de fichero pdf, imagen...
Campos de la información	• Identificador de la entidad • Ruta en el servidor del recurso
Prioridad	Baja

Tabla 14: RF-INF-009 Recurso

4.7.2. Casos de uso

Los casos de uso, al igual que próximos tipos de requisitos, se dividen en función del tipo de usuario (actores del sistema) que puede realizarlos.

Antes de realizar cualquiera de ellos, es necesario autenticarse en el sistema.

RF-CU-001	Autenticación	
Versión	2.0	
Dependencias	RG-001 Administración de usuarios RF-INF-001 Información de un usuario RF-CON-001 Autenticación	
Descripción	Pantalla inicial del sistema en que el cliente se autenticará.	
Precondición	Ninguna	
Secuencia	Paso	Acción
	1	El cliente accede al sistema.
	2	Introduce sus credenciales
Postcondición	En función del rol del cliente, se le dirigirá a una u otra pantalla.	
Prioridad	Alta	

Tabla 15: RF-CU-001 Autenticación

4.7.2.1. Administración de usuarios (rol: administrador de usuarios)

Los casos de uso propios de la administración de usuarios están relacionados con la gestión de los usuarios del sistema.

RF-CU-002	Crear usuario	
Versión	2.0	
Dependencias	RG-001 Administración de usuarios RF-INF-001 Información de un usuario RF-CON-002 Gestionar usuarios	
Descripción	El sistema permitirá la creación de un usuario y la asignacion de su rol de forma sencilla.	
Precondición	Acceder al panel de administración de usuarios.	
Secuencia	Paso	Acción
	1	Se selecciona la opción de crear usuario.
	2	Se elige su alias y su rol (entre administrador de mapas y usuario).
	3 A	Si se ha elegido un alias que ya tiene otro usuario (el alias debe ser único), se notifica por pantalla de que no ha podido crearse el usuario.
	3 B	En caso exitoso, se confirman los cambios.
Postcondición	El sistema cargará de nuevo la lista de usuarios, incluyendo el nuevo.	
Prioridad	Alta	

Tabla 16: RF-CU-002 Crear usuario

RF-CU-003	Eliminar usuario	
Versión	2.0	
Dependencias	RG-001 Administración de usuarios RF-INF-001 Información de un usuario RF-CON-002 Gestionar usuarios	
Descripción	El sistema permitirá el eliminado de un usuario.	
Precondición	Acceder al panel de administración de usuarios.	
Secuencia	Paso	Acción
	1	Se selecciona selecciona la X roja que habrá junto al usuario que se desea eliminar.
	2	Se confirmará la acción mediante un aviso emergente.
	3 A	Si se trata de un usuario con mapas asignados, se notifica por pantalla de que no ha podido eliminarse el usuario.
	3 B	En caso exitoso, se confirman los cambios.
Postcondición	El sistema cargará de nuevo la lista de usuarios, descartando el seleccionado.	
Prioridad	Alta	

Tabla 17: RF-CU-003 Eliminar usuario

RF-CU-004	Modificar rol de usuario	
Versión	2.0	
Dependencias	RG-001 Administración de usuarios RF-INF-001 Información de un usuario RF-CON-002 Gestionar usuarios	
Descripción	El sistema permitirá la edición del rol de un usuario.	
Precondición	Acceder al panel de administración de usuarios.	
Secuencia	Paso	Acción
	1	Se elige el rol deseado en el desplegable ubicado junto al nombre del usuario.
	2	Se selecciona el *check* verde que habrá junto al usuario que se desea modificar.
	3 A	Si se trata de un usuario con mapas asignados, se notifica por pantalla de que no ha podido modificarse el usuario.
	3 B	En caso exitoso, se confirman los cambios.
Postcondición	El sistema cargará de nuevo la lista de usuarios, con el rol modificado.	
Restricciones	No se podrá cambiar el rol de uno mismo (si se cambiara, el sistema se quedaría sin administrador de usuario).	
	La modificación será entre administrador de mapas y usuario, no se podrá elegir otro administrador de usuarios.	
Prioridad	Alta	

Tabla 18: RF-CU-004 Modificar rol de usuario

4.7.2.2. Administración geográfica (rol: administrador geográfico)

RF-CU-005	Creación/Edición de mapa	
Versión	2.0	
Dependencias	RG-002 Administración de mapas RF-INF-004 Información de un mapa RF-INF-005 Listado de mapas base RF-NEG-001 Mapa base de cada mapa RF-NEG-002 Capas de un mapa RF-CON-003 Gestionar mapas	
Descripción	El sistema permitirá la creación de un nuevo mapa o la edición de uno ya existente.	
Precondición	Acceder al panel de administración geográfica.	
Secuencia	Paso	Acción
	1 A	Seleccionar la opción de “Crear mapa”.
	1 B	Seleccionar la opción de “Editar” junto al mapa que se desee.
	2	Elegir un mapa base.
	3	Elegir las capas existentes que quieren incluirse.
	4	Introducir el nombre del mapa, y metadatos para facilitar su identificación.
Postcondición	El sistema cargará de nuevo la lista de mapas, con el mapa nuevo/editado ya incluído.	
Restricciones	No se podrá crear un mapa sin mapa base	
Prioridad	Alta	

Tabla 19: RF-CU-005 Creación/Edición de mapa

RF-CU-006	Creación/Edición de capa	
Versión	2.0	
Dependencias	RG-003 Gestión de capas RF-INF-002 Información de una capa RF-INF-003 Información de una entidad RF-INF-007 Recurso RF-CON-004 Creación y edición de capas	
Descripción	El sistema permitirá la creación de una nueva capa o la edición de una ya existente. Una capa puede estar ligada a más de un mapa, por lo que éste caso de uso es independiente de los mapas, será únicamente en la creación/edición de mapa donde podrán vincularse capas con mapas.	
Precondición	Acceder al panel de administración geográfica.	
Secuencia	Paso	Acción
	1 A	Seleccionar la opción de “Crear capa”.
	1 B	Seleccionar la opción de “Editar” junto a la capa que se desee.
	2	Seleccionar el tipo de figura geométrica que se desea para las entidades de la capa.
	3	Introducir los campos que se desee que pueda tener cada entidad.
	4 A	Dibujar o marcar las entidades que se desee añadir.
	4 B	Eliminar, opción que aparecerá al clicar sobre una entidad existente, las que se deseen.
	4 C	Editar los metadatos de cada entidad (datos alfanuméricos o recursos adjuntos), opción que aparecerá al clicar sobre una entidad existente.
	5	Introducir el nombre de la capa, y metadatos para facilitar su identificación.
Postcondición	El sistema cargará de nuevo la lista de capas, con la capa nueva/editada ya incluída.	
Restricciones	No se podrá crear una capa sin ninguna entidad.	
Prioridad	Media	

Tabla 20: RF-CU-006 Creación/Edición de capa

RF-CU-007	Creación/Edición de capa mediante shapefile	
Versión	2.0	
Dependencias	RG-003 Gestión de capas RF-INF-002 Información de una capa RF-INF-003 Información de una entidad RF-CON-004 Creación y edición de capas	
Descripción	El sistema permitirá la creación de una nueva capa o la edición de una ya existente mediante la subida de un fichero shapefile. Una capa puede estar ligada a más de un mapa, por lo que éste caso de uso es independiente de los mapas, será únicamente en la creación/edición de mapa donde podrán vincularse capas con mapas.	
Precondición	Acceder al panel de administración geográfica.	
Secuencia	Paso	Acción
	1 A	Seleccionar la opción de “Crear capa”.
	1 B	Seleccionar la opción de “Editar” junto a la capa que se desee.
	2	Subir el fichero correspondiente.
	3	Introducir el nombre de la capa, y metadatos para facilitar su identificación.
Postcondición	El sistema cargará de nuevo la lista de capas, con la capa nueva/editada ya incluído, si la validación del fichero resulta satisfactoria.	
Restricciones	No se podrá crear una capa sin ninguna entidad o cuyo fichero no sea válido.	
Prioridad	Baja	

Tabla 21: RF-CU-007 Creación/Edición de capa mediante shapefile

RF-CU-008	Asignación (o desasignación) de mapas a usuarios	
Versión	2.0	
Dependencias	RG-001 Administración de usuarios RG-002 Administración de mapas RF-INF-004 Información de un mapa RF-CON-003 Gestionar mapas	
Descripción	El sistema permitirá asignar a cada usuario los mapas que se deseen, en función de las tareas que tenga que desempeñar éste.	
Precondición	Acceder al panel de administración geográfica.	
Secuencia	Paso	Acción
	1	Buscar el usuario deseado en el listado de usuarios del módulo.
	2 A	Seleccionar un nuevo mapa y clicar el botón de añadir.
	2 B	Clicar el botón de eliminar que haya junto al mapa que se desee borrar de un determinado usuario. En este caso, aparecerá una ventana emergente para la confirmación.
Prioridad	Alta	

Tabla 22: RF-CU-008 Asignación (o desasignación) de mapas a usuarios

4.7.2.3. Visor (rol: usuario)

Los casos de uso de un usuario se tratan, principalmente, de las herramientas de las que disponen para mejorar la visualización del mapa.

RF-CU-009	Inicializar visor	
Versión	1.0	
Dependencias	RG-004 Visualización de mapas	
Descripción	El sistema ofrecerá un visor que permitirá a los usuarios de éste la visualización de diferentes mapas con sus correspondientes capas, así como las entidades de estas.	
Precondición	El usuario accederá al visor.	
Secuencia	Paso	Acción
	1	El usuario accede al visor.
	2	El sistema cargará el visor con los mapas que tenga disponibles en una barra de herramientas.
	3	Al seleccionar un mapa, se cargará éste con la configuración básica de las capas que contenga.
Postcondición	El sistema cargará el visor.	
Prioridad	Alta	

Tabla 23: RF-CU-009 Inicializar visor

RF-CU-010	Visualizar capas	
Versión	1.0	
Dependencias	RG-004 Visualización de mapas RF-INF-002 Información de una capa RF-INF-004 Información de un mapa RF-CON-010 Tabla de contenidos	
Descripción	Herramienta que contiene todas las capas en el mapa.	
Precondición	Abrir la herramienta de “Tabla de contenidos”. Por defecto, estará abierta.	
Secuencia	Paso	Acción
	1	El usuario selecciona el mapa, de los que dispone, que desea visualizar.
	2	El usuario podrá mostrar u ocultar las capas del mapa seleccionado, o modificar la trasparencia de alguna de ellas.
Postcondición	La capa seleccionada se hará activa o modificará su estado.	
Prioridad	Alta	

Tabla 24: RF-CU-010 Visualizar capas

RF-CU-011	Identificación de una entidad	
Versión	1.0	
Dependencias	RG-004 Visualización de mapas RF-INF-003 Información de una entidad RF-CON-021 Identificación y consulta de entidades	
Descripción	Herramienta que obtiene información de las entidades de un mapa.	
Precondición	Se abre la herramienta de "Identificador".	
Secuencia	Paso	Acción
	1	El usuario clicará en el mapa sobre la entidad de la que desea obtener información.
Postcondición	Se rellenará el grid de información con el elemento identificado.	
Prioridad	Alta	

Tabla 25: RF-CU-011 Identificación de una entidad

RF-CU-012	Localizar entidad por metadatos	
Versión	1.0	
Dependencias	RG-004 Visualización de mapas RF-INF-003 Información de una entidad RF-CON-020 Búsqueda alfanumérica de una entidad	
Descripción	Herramienta que localiza espacialmente una entidad según su nombre o sus metadatos.	
Precondición	Abrir la herremienta "Buscador".	
Secuencia	Paso	Acción
	1	Escribir algún dato del elemento que se quiere localizar.
	2	El usuario hace clic en el botón "Buscar".
	3	Se realizará zoom a la zona del mapa en que se sitúe la entidad.
Prioridad	Media	

Tabla 26: RF-CU-012 Localizar entidad por metadatos

RF-CU-013	Localizar zona por coordenadas	
Versión	1.0	
Dependencias	RG-004 Visualización de mapas RF-CON-016 Seleccionar ubicación según coordenadas	
Descripción	Herramienta que permite la localización rápida de una entidad según sus coordenadas de longitud y latitud.	
Precondición	Se abre la herramienta denominada “Localizador”.	
Secuencia	Paso	Acción
	1	Se introducen las coordenadas deseadas.
	2	El visor hará zoom a las coordenadas indicadas.
Prioridad	Media	

Tabla 27: RF-CU-013 Localizar zona por coordenadas

RF-CU-014	Dibujar y medir	
Versión	1.0	
Dependencias	RG-004 Visualización de mapas RF-CON-015 Mediciones	
Descripción	Herramienta de dibujo y medición del visor.	
Precondición	Se ha de abrir la herramienta de “Dibujar y medir”.	
Secuencia	Paso	Acción
	1	El usuario seleccionará el tipo de geometría a dibujar en el mapa.
	2	El usuario podrá configurar las características de cada del elemento a dibujar: color, grosor, trasparencia y tipo de borde.
	3	El usuario dibujará el elemento en el mapa
Postcondición	Se mostrará en pantalla el elemento pintado en el mapa. En el mapa aparecerán los datos de medición del elemento pintado. Además, se podrá guardar el dibujo.	
Prioridad	Media	

Tabla 28: RF-CU-014 Dibujar y medir

RF-CU-015	Descarga de recursos	
Versión	1.0	
Dependencias	RG-004 Visualización de mapas RF-INF-007 Recurso RF-CON-022 Descarga de recursos asociados a la entidad	
Descripción	Herramienta que contiene un conjunto de recursos asociadas a una entidad.	
Precondición	Existencia de recursos en la entidad seleccionada.	
Secuencia	Paso	Acción
	1	Seleccionar un elemento
	2	Abrir el widget “Recursos”
	3	Se mostrará el fichero.
Prioridad	Baja	

Tabla 29: RF-CU-015 Descarga de recursos

RF-CU-016	Añadir sitio predeterminado	
Versión	1.0	
Dependencias	RG-004 Visualización de mapas RF-CON-019 Añadir y consultar favoritos	
Descripción	Herramienta tipo “Favorito” que almacena localizaciones para un rápido acceso.	
Secuencia	Paso	Acción
	1	Abrir la herramienta “Favoritos”.
	2	Seleccionar la pestaña “añadir favorito”.
	3	Ingresar una etiqueta para la localización o elemento.
	4	Seleccionar el botón “Añadir Favorito” y pulsar en su ubicación en el mapa.
Prioridad	Media	

Tabla 30: RF-CU-016 Añadir sitio predeterminado

RF-CU-017	Ir a sitio predeterminado	
Versión	1.0	
Dependencias	RG-004 Visualización de mapas RF-CON-019 Añadir y consultar favoritos	
Descripción	Herramienta tipo "Favorito" que almacena localizaciones y elementos para un rápido acceso.	
Secuencia	Paso	Acción
	1	Abrir la herramienta "Favoritos".
	2	Seleccionar la localización a hacer zoom del listado de favoritos existentes.
	3	Se hará zoom en el elemento.
Prioridad	Media	

Tabla 31: RF-CU-017 Ir a sitio predeterminado

RF-CU-018	Imprimir	
Versión	1.0	
Dependencias	RG-004 Visualización de mapas RF-CON-017 Impresión de mapas	
Descripción	Herramienta de impresión.	
Precondición	Se se selecciona la herramienta de "Imprimir".	
Secuencia	Paso	Acción
	1	El usuario seleccionará el tipo de tamaño de salida.
	2	El usuario presionará el botón de impresión.
Postcondición	Se abrirá el marco de impresión del sistema operativo para seleccionar la impresora y sus características de impresión.	
Prioridad	Baja	

Tabla 32: RF-CU-018 Imprimir

RF-CU-019	Clusterización de los elementos de una capa	
Versión	1.0	
Dependencias	RG-004 Visualización de mapas RF-NEG-003 Clusterización de polígonos RF-CON-026 Clústers	
Descripción	Posibilidad de, en una capa cuyos elementos son puntos, observarlos en forma de clúster, es decir, agrupando los puntos que se encuentren cercanos y mostrando sobre el punto resultando el número de elementos que se han agrupado.	
Precondición	Contar con alguna capa cuyos elementos sean puntos. Abrir la herramienta "Tabla de contenidos".	
Secuencia	Paso	Acción
	1	Se tendrá que marcar la casilla "Ver clúster" situada junto al nombre de la capa en la herramienta "Tabla de contenidos".
Prioridad	Media	

Tabla 33: RF-CU-019 Clusterización de los elementos de una capa

4.7.3. Requisitos de reglas de negocio

RF-NEG-001	Mapas base de cada mapa
Versión	2.0
Dependencias	RG-002 Administración de mapas RF-INF-004 Información de un mapa RF-INF-005 Listado de mapas base RF-CU-005 Creación/Edición de mapa RF-CON-003 Gestionar mapas
Descripción	El sistema no permitirá que se cree un mapa sin un mapa base.
Implementación	Se comprobará que se haya seleccionado un mapa base cuando se vaya a guardar uno nuevo o los cambios de uno existente.
Prioridad	Alta

Tabla 34: RF-NEG-001 Mapas base de cada mapa

RF-NEG-002	Capas de un mapa
Versión	2.0
Dependencias	RG-002 Administración de mapas RF-INF-004 Información de un mapa RF-CU-005 Creación/Edición de mapa RF-CON-003 Gestionar mapas
Descripción	El sistema no permitirá que se cree un mapa sin, al menos, una capa.
Implementación	Se comprobará que se haya seleccionado al menos una capa cuando se vaya a guardar uno nuevo o los cambios de uno existente.
Prioridad	Alta

Tabla 35: RF-NEG-002 Capas de un mapa

RF-NEG-003	Clusterización de polígonos
Versión	2.0
Dependencias	RG-004 Visualización de mapas RF-CU-019 Clusterización de los elementos de una capa RF-CON-026 Clústers
Descripción	El sistema no permitirá que se active la visualización en modo clúster en una capa cuyas entidades sean polígonos (solo se podrá si son puntos).
Implementación	La casilla para mostrar las entidades en clúster aparecerá bloqueada si las entidades de la capa son polígonos.
Prioridad	Media

Tabla 36: RF-NEG-003 Clusterización de polígonos

4.7.4. Requisitos de conducta

Estos requisitos describen los servicios que debe ofrecer el sistema para que los usuarios puedan realizar sus tareas de negocio [ERS_7].

RF-CON-001	Autenticación
Versión	1.1
Dependencias	RG-001 Administración de usuarios RF-INF-001 Información de un usuario RF-CU-001 Autenticación
Descripción	Para acceder al sistema, un usuario tendrá que autenticarse con usuario y contraseña.
Implementación	Pantalla que se mostrará inicialmente o cuando no se dispongan de permisos para visualizar la solicitada.
Prioridad	Alta

Tabla 37: RF-CON-001 Autenticación

4.7.4.1. Administrador de usuarios

RF-CON-002	Gestionar usuarios
Versión	1.1
Dependencias	RG-001 Administración de usuarios RF-INF-001 Información de un usuario RF-CU-002 Crear usuario RF-CU-003 Eliminar usuario RF-CU-004 Modificar rol de usuario
Descripción	Si el usuario es administrador, podrá crear, eliminar o modificar los permisos (qué mapas puede ver, editar…) de un usuario.
Implementación	Pantalla a la que se accederá con el rol de “administrador de usuarios”, que ofrecerá las opciones descritas en un menú con el listado de usuarios.
Prioridad	Alta

Tabla 38: RF-CON-002 Gestionar usuarios

4.7.4.2. Administrador geográfico

RF-CON-003	Gestionar mapas
Versión	1.1
Dependencias	RG-002 Administración de mapas RF-INF-004 Información de un mapa RF-INF-005 Listado de mapas base RF-CU-005 Creación/Edición de mapa RF-CU-008 Asignación de mapas a usuarios RF-NEG-001 Mapa base de cada mapa RF-NEG-002 Capas de un mapa
Descripción	Si el usuario es administrador, podrá crear y editar mapas, seleccionando las capas, carpetas… que formarán parte del mismo.
Implementación	Pantalla a la que se accederá con el rol de "administrador geográfico", que ofrecerá las opciones descritas en distintos menús: lista de mapas, panel de creación/edición...
Prioridad	Alta

Tabla 39: RF-CON-003 Gestionar mapas

RF-CON-004	Creación y edición de capas
Versión	2.0
Dependencias	RG-003 Gestión de capas RF-INF-002 Información de una capa RF-INF-003 Información de una entidad RF-CU-006 Creación/Edición de capa RF-CU-007 Creación/Edición de capa mediante shapefile
Descripción	El GIS debe proporcionar herramientas para la creación de capas y la edición de las entidades que la componen.
Implementación	Pantalla a la que se accederá con el rol de "administrador geográfico", en la que se podrá editar dinámicamente y sobre un mapa base las entidades que forman parte de una capa: dibujando las nuevas y seleccionando las que se quieren eliminar.
Prioridad	Media

Tabla 40: RF-CON-004 Creación y edición de capas

RF-CON-005	Asociación de recursos a entidades
Versión	1.1
Dependencias	RG-003 Gestión de capas RF-INF-007 Recurso
Descripción	El GIS debe proporcionar herramientas que permitan la asociación de recursos relacionados con la entidad. Podrán asociarse tanto información contenida en ficheros (pdf, png, doc) como recursos externos publicados en una url.
Implementación	Durante la creación/edición de capas, al seleccionar una entidad existente, se presentará la opción de subir un fichero al sistema o introducir una url.
Prioridad	Baja

Tabla 41: RF-CON-005 Asociación de recursos a entidades

4.7.4.3. Usuario

RF-CON-006	Maximizar tamaño del mapa
Versión	1.1
Dependencias	RG-005 Personalización del visor
Descripción	El visor debe maximizar el tamaño del mapa, eliminando el banner superior. Todos los menús deben ser colapsables.
Implementación	Un botón en la esquina del visor permitirá maximizar (y devolver a su estado original) el visor.
Prioridad	Alta

Tabla 42: RF-CON-006 Maximizar tamaño del mapa

RF-CON-007	Navegación sobre el mapa
Versión	1.1
Dependencias	RG-004 Visualización de mapas
Descripción	El sistema permitirá el desplazamiento del fragmento de mapa que se ve en la pantalla.
Implementación	Se desplazará el mapa manteniendo pulsado en algún punto del mismo y arrastrando el cursor hacia un lado.
Prioridad	Alta

Tabla 43: RF-CON-007 Navegación sobre el mapa

RF-CON-008	Zoom sobre el mapa
Versión	1.1
Dependencias	RG-004 Visualización de mapas
Descripción	El sistema permitirá el desplazamiento sobre los mapas.
Implementación	El sistema permitirá hacer zoom tanto con unos controles situados en una esquina, como con el scroll del ratón.
Prioridad	Alta

Tabla 44: RF-CON-008 Zoom sobre el mapa

RF-CON-009	Barras de herramientas
Versión	1.1
Dependencias	RG-004 Visualización de mapas
Descripción	Las distintas herramientas para la navegación, citadas principalmente en los casos de uso, se situarán en sendas barras de herramientas.
Implementación	Se situarán en los laterales superior e inferior, y serán colapsables.
Prioridad	Alta

Tabla 45: RF-CON-009 Barras de herramientas

RF-CON-010	Tabla de contenidos
Versión	1.1
Dependencias	RG-004 Visualización de mapas RF-INF-002 Información de una capa RF-INF-004 Información de un mapa RF-CU-010 Visualizar capas
Descripción	En la barra del lateral izquierdo se encontrará la tabla de contenidos. En ella, se mostrarán los posibles mapas y capas a los que puede acceder el usuario.
Implementación	Se situará en la barra del lateral izquierdo. Se eligirá un mapa y, dentro de él, las capas que se desee mostrar. Además, en cada capa, aparecerán las opciones: • Transparencia: se podrá modificar el nivel de transparencia del servicio seleccionado. • Acercar: realiza un encuadre a la extensión total del servicio. • Clúster: si se tratan de puntos, mostrarlos en forma de clúster.
Prioridad	Alta

Tabla 46: RF-CON-010 Tabla de contenidos

RF-CON-011	Visualización de leyenda
Versión	1.1
Dependencias	RG-004 Visualización de mapas RF-INF-006 Información de la leyenda
Descripción	Existirá una herramienta donde se centralizarán todas las leyendas de las capas.
Implementación	Se mostrarán en una de las esquinas del visor. Éstas serán activadas dinámicamente en función de las capas que se muestren en el visor en ese momento.
Prioridad	Alta

Tabla 47: RF-CON-011 Visualización de leyenda

RF-CON-012	Visualización de escala
Versión	1.1
Dependencias	RG-004 Visualización de mapas
Descripción	El sistema deberá permitir visualizar la escala del mapa según el zoom actual.
Implementación	En una de las esquinas del visor aparecerá esta información.
Prioridad	Alta

Tabla 48: RF-CON-012 Visualización de escala

RF-CON-013	Posición del cursor
Versión	1.1
Dependencias	RG-004 Visualización de mapas
Descripción	El sistema deberá permitir visualizar dinámicamente la longitud y la latitud de la posición actual del cursor.
Implementación	En una de las esquinas del visor aparecerá esta información.
Prioridad	Alta

Tabla 49: RF-CON-013 Posición del cursor

RF-CON-014	Mapa guía
Versión	1.1
Dependencias	RG-004 Visualización de mapas
Descripción	El sistema deberá permitir mostrar un mapa general con un recuadro indicando el zoom actual sobre el mapa del visor.
Implementación	En una de las esquinas del visor aparecerá esta información gráfica.
Prioridad	Media

Tabla 50: RF-CON-014 Mapa guía

RF-CON-015	Mediciones
Versión	1.1
Dependencias	RG-004 Visualización de mapas RF-CU-014 Dibujar y medir
Descripción	El sistema permitirá realizar mediciones de longitudes y áreas sobre los mapas.
Implementación	En la barra de herramientas aparecerán las distintas opciones de medición (área, longitud).
Prioridad	Alta

Tabla 51: RF-CON-015 Mediciones

RF-CON-016	Seleccionar ubicación según coordenadas
Versión	1.1
Dependencias	RG-004 Visualización de mapas RF-CU-013 Localizar zona por coordenadas
Descripción	El usuario podrá hacer zoom a una región en función de unas coordenadas que introduzca manualmente.
Implementación	La opción estará en la barra de herramientas, con una caja para la latitud y otra para la longitud.
Prioridad	Media

Tabla 52: RF-CON-016 Seleccionar ubicación según coordenadas

RF-CON-017	Impresión de mapas
Versión	1.1
Dependencias	RG-004 Visualización de mapas RF-CU-018 Imprimir
Descripción	El sistema deberá permitir imprimir la vista actual del mapa que se esté mostrando.
Implementación	La opción estará en la barra de herramientas. Se seleccionará el tamaño y se encuadrará el visor en la zona que desee imprimirse.
Prioridad	Baja

Tabla 53: RF-CON-017 Impresión de mapas

RF-CON-018	Exportación a Excel
Versión	1.1
Dependencias	RG-004 Visualización de mapas
Descripción	El sistema deberá permitir exportar a un fichero Excel los datos de los elementos del mapa seleccionados.
Implementación	La opción estará en la barra de herramientas. Se requerirá marcar en el mapa los elementos de los que se desee obtener los datos, y se exportará a continuación. En el Excel, se podrán ver los metadatos de cada entidad.
Prioridad	Baja

Tabla 54: RF-CON-018 Exportación a Excel

RF-CON-019	Añadir y consultar favoritos
Versión	1.1
Dependencias	RG-004 Visualización de mapas RF-CU-016 Ir a sitio predeterminado RF-CU-016 Añadir sitio predeterminado
Descripción	Se podrá añadir un elemento o una localización geográfica a favoritos, y consultar la lista de estos.
Implementación	La opción estará en la barra de herramientas. En la parte superior de la herramienta, se mostrará un botón de “Añadir”, y debajo aparecerá la lista de favoritos anteriormente guardados junto a un botón con el que se hará zoom en su ubicación.
Prioridad	Media

Tabla 55: RF-CON-019 Añadir y consultar favoritos

RF-CON-020	Búsqueda alfanumérica de una entidad
Versión	1.1
Dependencias	RG-004 Visualización de mapas RF-INF-007 Recurso RF-CU-012 Localizar entidad por metadatos
Descripción	El GIS debe permitir realizar búsquedas de los recursos y los metadatos de una entidad mediante criterios alfanuméricos.
Implementación	Tras introducir la búsqueda en una caja para tal efecto, el sistema mostrará el listado de elementos resultantes de la consulta. El usuario podrá seleccionar uno los resultados y realizar un encuadre de dicho elemento sobre el mapa.
Prioridad	Media

Tabla 56: RF-CON-020 Búsqueda alfanumérica de una entidad

RF-CON-021	Identificación y consulta de entidades
Versión	1.1
Dependencias	RG-004 Visualización de mapas RF-CU-011 Identificación de una entidad
Descripción	El sistema permitirá identificar un elemento de una capa a partir de su selección sobre el mapa.
Implementación	Se mostrará en la barra de herramientas su nombre y metadatos.
Prioridad	Baja

Tabla 57: RF-CON-021 Identificación y consulta de entidades

RF-CON-022	Descarga de recursos asociados a la entidad
Versión	1.1
Dependencias	RG-004 Visualización de mapas RF-INF-007 Recurso RF-CU-015 Descarga de recursos
Descripción	El GIS debe proporcionar herramientas que permitan consultar los recursos asociados a una entidad.
Implementación	Al clicar sobre una entidad, aparecerán los recursos asociados a ella y la opción de descargarla.
Prioridad	Baja

Tabla 58: RF-CON-022 Descarga de recursos asociados a la entidad

RF-CON-023	Consulta de elementos de una capa
Versión	1.1
Dependencias	RG-004 Visualización de mapas RF-INF-002 Información de una capa
Descripción	El sistema permitirá consultar el conjunto de los elementos de una capa en forma de listado.
Implementación	Se podrá mostrar en la barra de herramientas el nombre y metadatos de cada una de las entidades de la capa.
Prioridad	Baja

Tabla 59: RF-CON-023 Consulta de elementos de una capa

RF-CON-024	Filtrado de las entidades de una capa
Versión	1.1
Dependencias	RG-004 Visualización de mapas RF-INF-002 Información de una capa RF-INF-003 Información de una entidad
Descripción	El sistema permitirá filtrar las entidades de una capa en función de alguno de sus parámetros.
Implementación	Se presentará un filtro en la barra de herramientas, adaptado a los campos que tengan los metadatos de esa capa. Solo aparecerán en el mapa las entidades que cumplan ese filtro.
Prioridad	Baja

Tabla 60: RF-CON-024 Filtrado de las entidades de una capa

RF-CON-026	Clústers
Versión	1.0
Dependencias	RG-004 Visualización de mapas RF-NEG-003 Clusterización de polígonos RF-CU-019 Clusterización de los elementos de una capa
Descripción	El GIS podrá mostrar los elementos en una zona a modo de clústers que irán descomponiéndose a medida que se vaya haciendo zoom sobre los mismos.
Implementación	Se podrá activar la opción en la "Tabla de contenidos", en capas cuyo tipo de entidad sea un punto.
Prioridad	Baja

Tabla 61: RF-CON-026 Clústers

4.8. Requisitos no funcionales

Se trata de requisitos que especifican condiciones que se le imponen al sistema a desarrollar relacionadas con aspectos principalmente de calidad y operación, en lugar de sus comportamientos específicos (Taxonomía de requisitos).

4.8.1. Requisitos de fiabilidad

RNF-FIA-001	Tiempo de recuperación
Versión	2.0
Dependencias	Todo el sistema.
Descripción	El sistema deberá tardar un máximo de 10 minutos para la recuperación de un fallo de caída total, en el 95% de las ocasiones.
Prioridad	Media

Tabla 62: RNF-FIA-001 Tiempo de recuperación

4.8.2. Requisitos de usabilidad

RNF-USA-001	Facilidad para encontrar la opción deseada
Versión	2.0
Dependencias	Todo el sistema.
Descripción	El sistema deberá permitir en el 80% de las veces que con un máximo de 5 clicks sea suficiente para llegar a la información deseada.
Prioridad	Media

Tabla 63: RNF-USA-001 Facilidad para encontrar la opción deseada

4.8.3. Requisitos de eficiencia

RNF-EFI-001	Rapidez en la respuesta
Versión	2.0
Dependencias	Todo el sistema.
Descripción	El sistema deberá tener un tiempo máximo de respuesta de 5 segundos para cualquier operación de consulta.
Prioridad	Media

Tabla 64: RNF-EFI-001 Rapidez en la respuesta

4.8.4. Requisitos de mantenibilidad

RNF-MAN-001	Cumplimiento de estándares
Versión	2.0
Dependencias	Todo el sistema.
Descripción	El código fuente que se implemente deberá cumplir las recomendaciones de cada lenguaje y estar debidamente documentado (para qué sirve cada código) para que un tercero pueda entenderlo.
Prioridad	Media

Tabla 65: RNF-MAN-001 Cumplimiento de estándares

4.8.5. Requisitos de portabilidad

RNF-POR-001	Multiplataforma en escritorio
Versión	2.0
Dependencias	Todo el sistema.
Descripción	El sistema deberá proporcionar la misma experiencia en plataformas Windows, Unix y MacOS.
Prioridad	Media

Tabla 66: RNF-POR-001 Multiplataforma en escritorio

RNF-POR-002	Dispositivos móviles
Versión	2.0
Dependencias	Todo el sistema.
Descripción	El sistema deberá poder utilizarse en dispositivos Android e iOS, aunque la experiencia pueda no estar completamente adaptada.
Prioridad	Media

Tabla 67: RNF-POR-002 Dispositivos móviles

4.8.6. Requisitos de seguridad

RNF-SEG-001	Protección ante ataques
Versión	2.0
Dependencias	Todo el sistema.
Descripción	El sistema deberá ser capaz de evitar ataques de inyección de SQL sistemáticos.
Prioridad	Media

Tabla 68: RNF-SEG-001 Protección ante ataques

5 ARQUITECTURA

"Programar sin una arquitectura o diseño en mente es como explorar una gruta solo con una linterna: no sabes dónde estás, dónde has estado ni hacia dónde vas"

- Danny Thorpe-

Como ya se ha podido ver en la Especificación de Requisitos, es posible dividir el proyecto en tres grandes módulos: administración de usuarios, administración geográfica y visor. Además, necesitaremos de algunos módulos extra, como la autenticación en el sistema.

La arquitectura de la aplicación se dividirá en estos mismos módulos, contando cada uno con:

- Un controlador, que se divide en dos archivos: uno para retornar la vista de cada componente, y otro para interactuar con el modelo.
- Un conjunto de modelos.
- Un grupo de vistas, que utilizarán varios componentes. Componente es el nombre de cada mini-programa en que se puede dividir el sistema. Los componentes estarán codificados como una directiva de AngularJS, en las cuáles puede incluirse en una vista HTML. El uso de este concepto facilita la granularidad del sistema, diviendo los problemas en otros más pequeños y específicos, y permite usar las facilidades que proporciona AngularJS frente HTML+JS.

El uso del patrón modelo-vista-controlador resulta evidente.

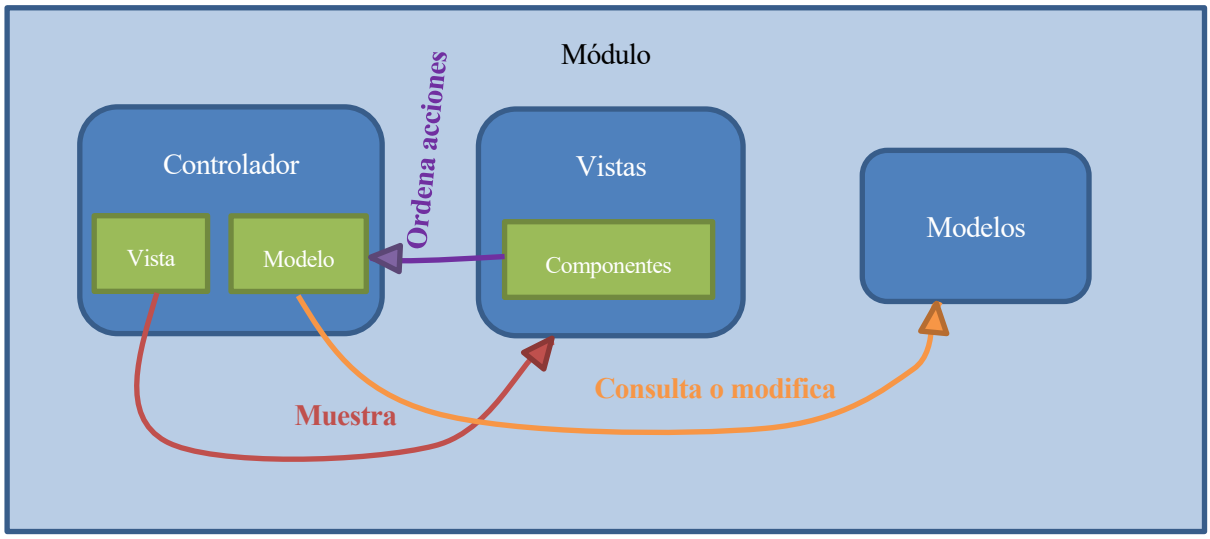


Ilustración 3: Arquitectura MVC

5.1. Módulos principales del sistema

5.1.1. Autenticación

El módulo de autenticación es la puerta al resto del sistema.

Su funcionalidad se limita a presentar un sistema de acceso que solicita alias y contraseña. Si la combinación introducida es correcta, se redirigirá al usuario al módulo correspondiente. En caso contrario, se le notificará del error.

5.1.1.1. Controlador

El fichero encargado de devolver las vistas tendrá una Acción por cada vista existente. Estas acciones simplemente devolverán la vista solicitada (no como en otros módulos, en los que habrá que comprobar si el usuario está autorizado para acceder a ellas.

El fichero encargado de interactuar con el modelo contará con un método que recibirá la combinación alias-contraseña en formato JSON. Tras descodificarlo, lanzará una consulta a la base de datos, buscando una coincidencia en la tabla de Usuarios con los datos introducidos.

5.1.1.2. Modelo

Para poder realizar esta consulta de existencia del usuario, necesitará la tabla Usuarios.

5.1.1.3. Vistas

Existe una única vista, Index (es decir, aparece por defecto), en la que se muestra el panel para la autenticación.

5.1.2. Administración de usuarios

Se trata de uno de los tres módulos principales, siendo el encargado de crear, eliminar y editar el rol de los usuarios del sistema.

5.1.2.1. Controlador

De nuevo, el fichero encargado de interactuar con las vistas tendrá una Acción por cada vista existente. En este caso, en cada acción, se comprobará que el usuario sea administrador de usuarios (usando el manager de seguridad, que veremos más adelante) y, según el resultado de la verificación, se presentará la vista solicitada o se devolverá al usuario a la vista de autenticación.

En cuanto al fichero encargado de interactuar con el modelo, contará con un método por cada consulta o modificación que ofrezca la vista. Éstas son:

- Consulta del conjunto de usuarios.
- Creación de un nuevo usuario.
- Edición de un usuario existente.
- Eliminado de un usuario.

5.1.2.2. Modelo

Se reproducirán las tablas de la base de datos relacionadas con la administración de usuarios.

En este caso, solo es una, Usuarios, a partir de la cual se crea un objeto usuario.

5.1.2.3. Vistas

Existen dos vistas:

- Index: Vista principal, en la que se muestra el listado de todos los usuarios.
- Creación de usuario: Vista en la que aparecerán las cajas necesarias para completar la información del usuario (nombre, contraseña, rol).

La edición y la eliminación de usuarios se realizarán en la misma vista Index.

En este caso, cada vista estará compuesta por un único componente (consulta de usuarios y creación de usuarios).

5.1.3. Administración geográfica

Se trata del módulo encargado de gestionar todo lo que rodea a los mapas: ellos mismos, sus capas y los usuarios que pueden utilizarlos.

5.1.3.1. Controlador

Al igual que en el módulo de administración de usuarios, el controlador estará encargado de comprobar que el usuario que trata de acceder tiene el rol de administrador geográfico.

Los métodos que tendrá el fichero encargado de interactuar con el modelo son:

- Consulta del conjunto de mapas.
- Consulta del conjunto de capas.
- Consulta de los mapas de cada usuario.
- Creación o edición de un mapa.
- Eliminación de un mapa.
- Creación o edición de una capa.
- Eliminación de una capa.
- Creación de una relación usuario-mapa.
- Eliminación de una relación usuario-mapa.

Sin duda, se trata del módulo que más se relaciona con el modelo de la aplicación.

5.1.3.2. Modelo

Para las tareas acometidas por este controlador, será necesario tener el modelo de las siguientes tablas, tanto para consultarlas como para crear sus respectivos objetos a partir de ellas:

- Mapas.
- Capas.
- Cada capa.
- Usuarios-mapas.

5.1.3.3. Vistas

Las vistas de este módulo serán tres:

- Index: Vista principal en la que se mostrarán los listados de mapas, capas, y usuarios-mapas. Cada uno de ellos serán un componente (una directiva) independiente de AngularJS. Mientras que las dos primeras solo serán listados y eliminado, el componente de usuarios-mapas realizará todo su negocio en esta misma vista.
- Nuevo/Edición mapa: Vista en la que se seleccionarán los parámetros (nombre, mapa base, capas) de un mapa.
- Nuevo/Edición capa: Vista en la que se seleccionarán los parámetros (tipo de polígono, representarlo sobre mapa) de una capa.

5.1.4. Visor

Módulo al que tendrá acceso el usuario y en el que visualizará los mapas que el administrador le haya otorgado, con la posibilidad de usar una relación de herramientas que le ayuden al análisis de los mapas.

5.1.4.1. Controlador

Al igual que en anteriores módulos, el controlador comprobará el rol del usuario que intenta acceder.

El fichero encargado de interactuar con el modelo contará con dos métodos: uno devolverá el listado de mapas, capas y usuarios relacionados; el otro los datos de un elemento de una capa.

5.1.4.2. Modelo

Se reproducirán las tablas de la base de datos relacionadas con la visualización de los mapas.

En este caso, solo es una, Vista usuarios-mapas-capas, ya que para la obtención de los los datos de una capa no es posible parametrizar un modelo.

5.1.4.3. Vistas

Solo contará con una vista, Index, pero ésta estará compuesta por multitud de componentes, uno por cada herramienta de interacción con el mapa.

5.2. Módulos auxiliares del sistema

Se trata de herramientas que son utilizadas por otros controladores para realizar sus tareas.

5.2.1. Shared

Se trata de un componente de AngularJS que se incluye en la vista Layout (que contiene el diseño básico de la aplicación web y que se muestra en todas sus ventanas). Este componente muestra la barra de navegación superior.

Esta barra incluye un mensaje que informa del alias del usuario que está conectado al sistema y de un botón que le permite cerrar sesión en él. Para ambas cosas, es necesario un controlador, el cual consulta y modifica el objeto HttpSessionState de la sesión HTTP, donde se guarda el alias y el rol del usuario una vez se autentica. Para salir del sistema, se borran estos valores, con lo que los controladores del resto de módulos no permitirán el acceso a sus vistas.

5.2.2. Manager de seguridad

Los controladores de los tres módulos principales, antes de devolver la correspondiente vista, consultan que el usuario tenga los permisos pertinentes. Para ello, utilizan el manager de seguridad.

Éste es un controlador con dos métodos:

- CheckUserInSession: Comprueba que el usuario siga en sesión. Esto resulta útil porque, transcurrido un tiempo sin realizar solicitudes HTTP, la sesión expira. Con esta comprobación, se asegura que el sistema no se quede abierto.
- CheckUserHasProfile: Comprueba que el usuario tenga el rol adecuado a la vista a la que desee acceder. Recibe como parámetro el nombre del módulo y comprueba que éste se corresponda con el rol almacenado en el objeto HttpSessionState.

5.2.3. Database Context

El mapeador objeto-relacional (técnica para convertir datos entre la base de datos y el lenguaje de programación) más extendido en .NET es Entity Framework. Pero, dado que éste no soporta de forma nativa PostgreSQL, se usará ADO.NET.

Para lograr la conexión con PostgreSQL, también se requerirá una librería, Npgsql.

La clase que se encarga de esta conexión es DatabaseContext. Esta clase se encarga tanto de abrir y cerrar la conexión sobre la base de datos, como realizar la sentencia que se desee (consulta, adición...).

En cada método de los controladores de los módulos principales, se crea una clase DatabaseContext, ejecutándose la acción que se desee sobre ella.

5.3. Modelo de datos

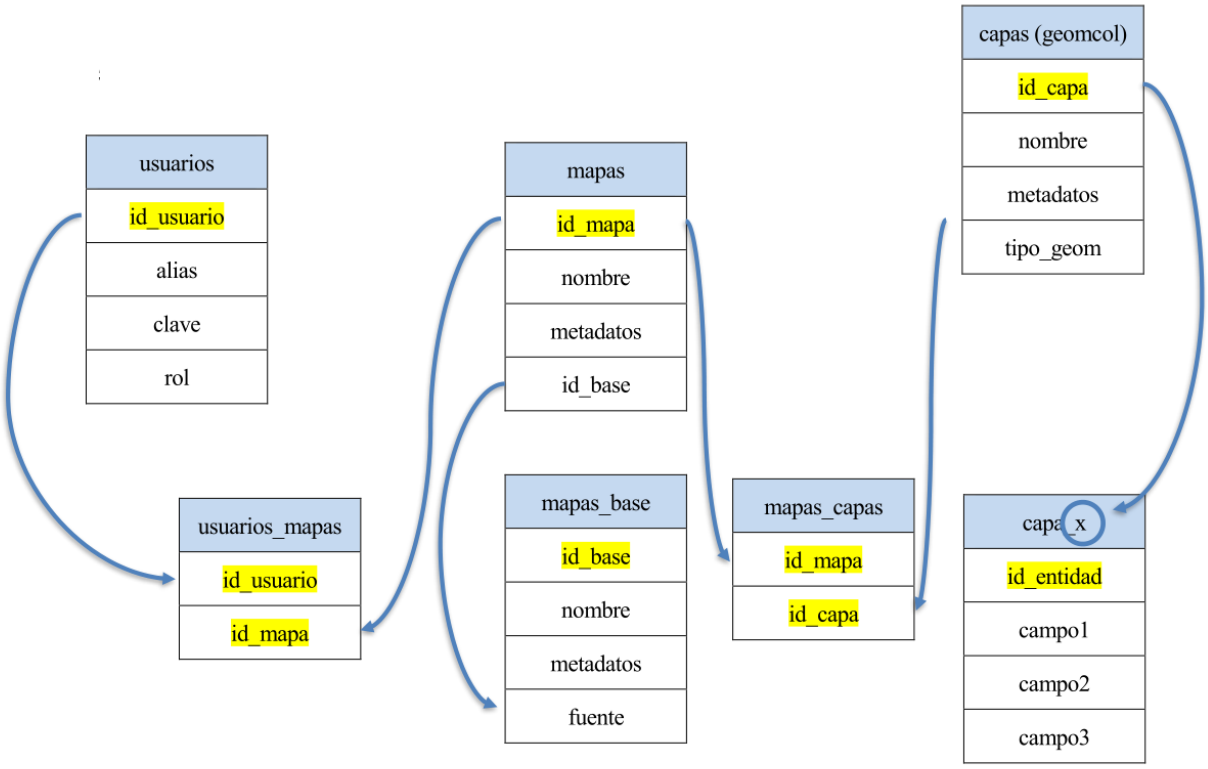


Ilustración 4: Modelo de datos

6 IMPLEMENTACIÓN

"Está bien investigar y resolver misteriosos asesinatos,
pero no deberías necesitar hacerlo con el código,
simplemente deberías poder leerlo"

- Steve McConnell -

En este apartado se contará cómo se ha trabajado en el entorno de desarrollo con la arquitectura explicada anteriormente. De igual modo, para facilitar su comprensión, se dividirá en los distintos módulos en que se puede distribuir la aplicación.

Todos los módulos estarán compuestos por, al menos, cinco ficheros:

/Views/Modulo/Submodulo.cshtml

/Controllers/ModuloController.cs

/Controllers/ModuloRestController.cs

/Components/modulo/submodulo/submodulo.js

/Components/modulo/submodulo/templates/submodulo.html

Siendo Modulo: Home, AdminUsu, AdminGeo, o Mapas. Y siendo Submodulo cada una de las herramientas que lo componen y se irán introduciendo.

6.1. Autentificación

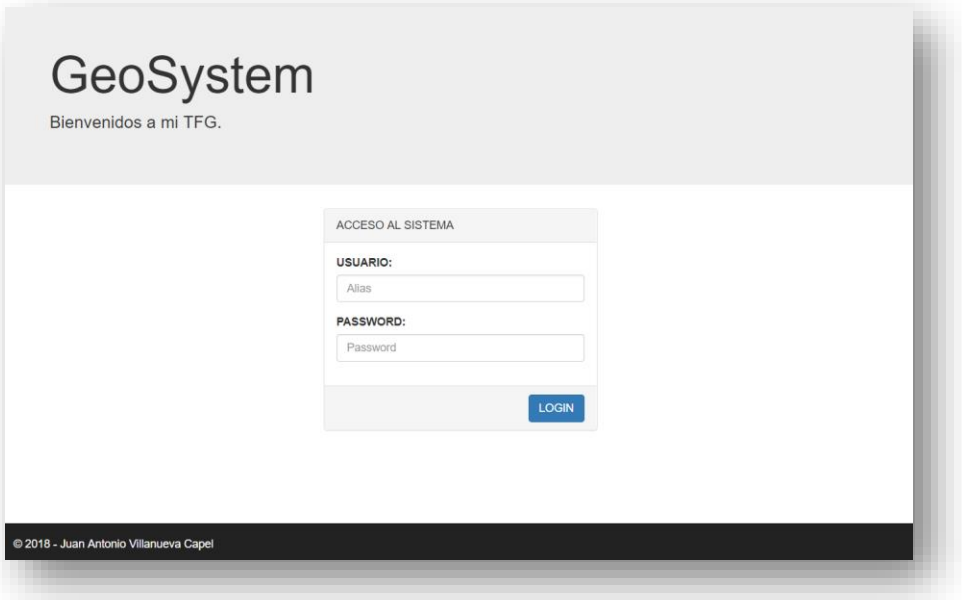


Ilustración 5: Módulo de Autentificación

6.1.1. /Views/Home/Index.cshtml

El fichero de la vista solo contiene como tal la información estática de la ventana.

Ilustración 6: Clase 'jumbotron' de Bootstrap

Para dar este formato, se utiliza la clase `jumbotron` de Bootstrap, que muestra el contenido que se introduzca dentro de ella en una caja grande gris.

En cuanto al panel de autenticación, tal y como sucederá en el resto de módulo, está incluído en un componente. Por ello, en la vista, lo único que se realiza es invocarlo.

```
<w-login></w-login>
```

Como ya se ha comentado en la arquitectura, lo que en este proyecto se está denominando como componentes, está codificado como directivas de AngularJS. Con `w-modulo`, se invoca a la directiva `wModulo`, la cual trataremos al adentrarnos en el fichero que la contenga (en el caso de `wModulo`, éste es /Components/home/login/login.js).

6.1.2. /Controllers/HomeController.cs

Se limita a retornar la vista Index de /Views/Home/Index.cshtml

```
public ActionResult Index()
{
    return View();
}
```

6.1.3. /Controllers/HomeRestController.cs

Este controlador está formado por dos métodos, `CheckUser` y `GetUser`.

```
public string CheckUser(string strParam)
```

Este método recibe un string en formato JSON que contiene el nombre del usuario y la contraseña que se hayan introducido en el formulario. Lo primero que hace es decodificarlo para poder utilizar esta información.

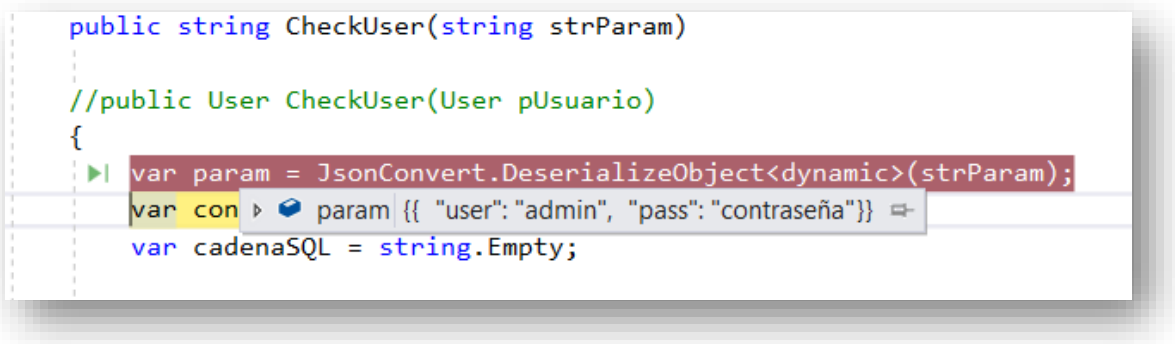


Ilustración 7: Decodificación de usuario y contraseña

A continuación, se crean las variables que almacenarán la cadena SQL que se lanzará al servidor de la base de datos y el resultado de esta consulta. Esta última variable sera de tipo User, definido en el modelo de la aplicación.

```
public class User
{
    [Key]
    public int id { get; set; }

    public string name { get; set; }
    public string role { get; set; }

}
```

Teniendo listas estas variables, se abrirá la conexión con la base de datos.

```
var db = conexion.OpenConnection()
```

Se darán más detalles sobre esta conexión en el apartado correspondiente de la implementación.

La cadena SQL a utilizar será: `"SELECT * FROM usuarios WHERE alias = '" + param.user+"'"` para param.user el nombre de usuario introducido en el formulario.

Se rellenará la variable de tipo User con la información encontrada.

```
resultado.id = (int)lector["id_usuario"];
resultado.name = (string)lector["alias"];
resultado.role = (string)lector["rol"];
```

De no haber encontrado ningún resultado, se devuelve el objeto con el identificador vacío.

Si el resultado sí ha sido satisfactorio, se procederá a comprobar si la contraseña introducida es la correcta. Como ya se tiene el identificador del usuario, se realizará una consulta SQL similar a la anterior pero esta vez buscando la coincidencia de la pareja idenficador-contraseña. Si esta comprobación da un resultado fallido, se informará al usuario de que la contraseña que ha introducido no es correcta.

Se realizan dos consultas SQL por dos motivos:

- Si en la primera consulta la cláusula WHERE incluyera tanto alias como contraseña, la respuesta obtenida del servidor (0 resultados) no nos permitiría saber si el error ha sido que el usuario no existe o que la contraseña es errónea.
- Si en la primera consulta “trajésemos” del servidor la contraseña del usuario y comprobáramos que ésta es correcta en el cliente, en Javascript, sería fácil manipular esta comprobación para un usuario.

En esta versión de la aplicación, al tratarse de un entorno de desarrollo, no se ha dado excesiva importancia a la seguridad de la misma (al margen de unas mínimas medidas, como la relatada justo anteriormente). Por ello, las contraseñas se almacenan tal cual son introducidas, sin ningún tipo de encriptado. Esto sería algo a desarrollar en una versión futura e implantable a un cliente de la misma.

Además de rellenar la variable que será devuelta al cliente con el resultado de la consulta, también se almacenan el nombre y el rol del usuario en el objeto `HttpSessionState` de la sesión, para poder luego comprobar que el usuario tiene permiso para acceder al módulo al que intente penetrar.

```
public string GetUser()
```

Este método se limita a devolver el nombre del usuario autenticado en el sistema, utilizando el objeto `HttpSessionState` anteriormente nombrado.

6.1.4. /Components/home/login/login.js

Este fichero contiene la directiva `wLogin`, perteneciente al módulo `home.login`. Esta directiva cuenta con sus propios vista y controlador. Su vista, en otro fichero HTML, estará enlazada mediante el atributo `templateUrl`. Su controlador se define en el propio fichero mediente el atributo `controller`.

Este controlador recibe varios objetos:

- `"$scope"`: Compartido entre la vista y el controlador para compartir información.
- `"$http"` y `"transformRequestAsFormPost"`: Utilizados para poder conectarse con servidores HTTP remotos, en este caso, el controlador de la aplicación.
- `"$window"`: Objeto de la ventana del navegador, para poder redirigirlo desde el propio conrolador.

En primer lugar, se definen el principal objeto que se utilizará.

```
$scope.data = {
       user: "",
       pass: ""
};
```

Además, está formado por dos métodos, `$scope.check` y `$scope.login`. Estos también se definen como `$scope` para poder invocarlos desde la vista HTML.

El primero de ellos se utiliza para comprobar si un usuario que acceda a la Home está autenticado en el sistema y redirigirlo en caso afirmativo, mientras que el segundo se ejecuta cuando se envía el formulario de autenticación.

6.1.4.1. Check

Se comienza realizando una petición HTTP al método `GetUser` del controlador para obtener el nombre del usuario. Si la respuesta devuelta por el método no se trata de un campo vacío, es decir, hay un usuario autenticado, se realiza una petición al método `CheckUser`.

```
$http({
    method: 'POST',                                  Método de la petición HTTP
    url: '/HomeRest/CheckUser',                      URL del método del controlador
    transformRequest: transformRequestAsFormPost,    Método que transforma el JSON
                                                     para poder enviarlo en un POST
    dataType: "json",                                Tipo del dato a enviar
    headers: {
        'Content-type': 'application/x-www-form-urlencoded'
    },
    data: {                                          Dato a enviar
        strParam: angular.toJson($scope.data)        Se transforma a JSON
    }
}).then(function (response) {                        Respuesta del controlador


});
```

Obtenida la respuesta, se redirije al usuario al módulo al que tenga derechos de acceso.

6.1.4.2. Login

Este método es muy similar al anterior a partir de la llamada al método `CheckUser`.

Obtenido el resultado de este método, si el usuario no está en el sistema, se le notificará por pantalla de ello.

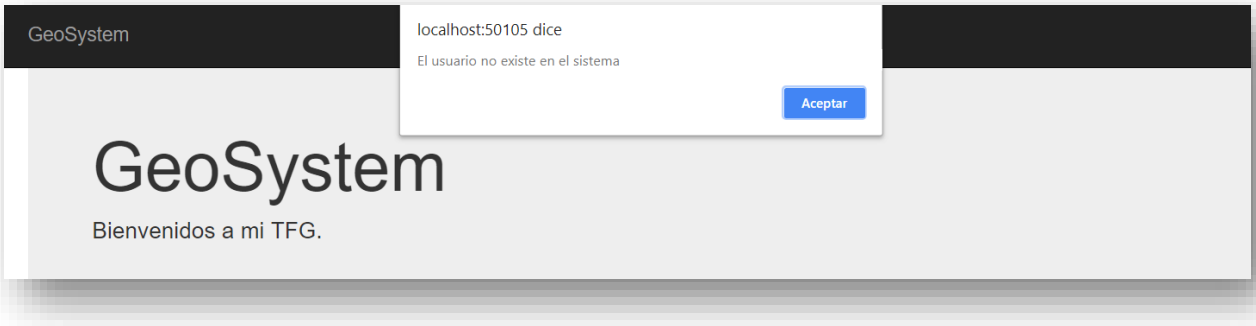


Ilustración 8: Alerta de usuario no existente en el sistema

Esto no se realiza en el caso del método Check al ejecutarse éste antes de que haya cualquier interacción del usuario con la vista.

En caso de que el usuario sí esté en el sistema, se le redirije al módulo correspondiente.

6.1.5. /Components/home/login/templates/login.html

En cuanto al fichero HTML, al margen del propio código de la vista, es en él en el que se invoca a los métodos del fichero Javascript.

Con el uso de la directiva `ng-init` es posible llamar a un método en el momento en que un bloque (div, por ejemplo) es mostrado.

Así, se llama al método `check`:

```
<div class="col-sm-4" ng-init="check();">

</div>
```

También es posible llamar a un método cuando se pulse un botón con la directiva ng-click:

```
<button class="btn btn-primary" id="btn_login" ng-click="login();"> LOGIN </button>
```

En cuanto a cómo se obtiene la información introducida en el formulario, se utiliza la directiva `ng-model`, pasándole como parámetro la misma ruta (del objeto $scope) que se utiliza en el fichero Javascript.

```
<form>
    <div class="form-group">
        <label for="user">USUARIO:</label>
        <input type="text" class="form-control" placeholder="Alias" ng-model="data.user">
    </div>
    <div class="form-group">
        <label for="pwd">PASSWORD:</label>
        <input type="password" class="form-control" placeholder="Password"
                                                              ng-model="data.pass">
    </div>

</form>
```

6.2. Administración de usuarios

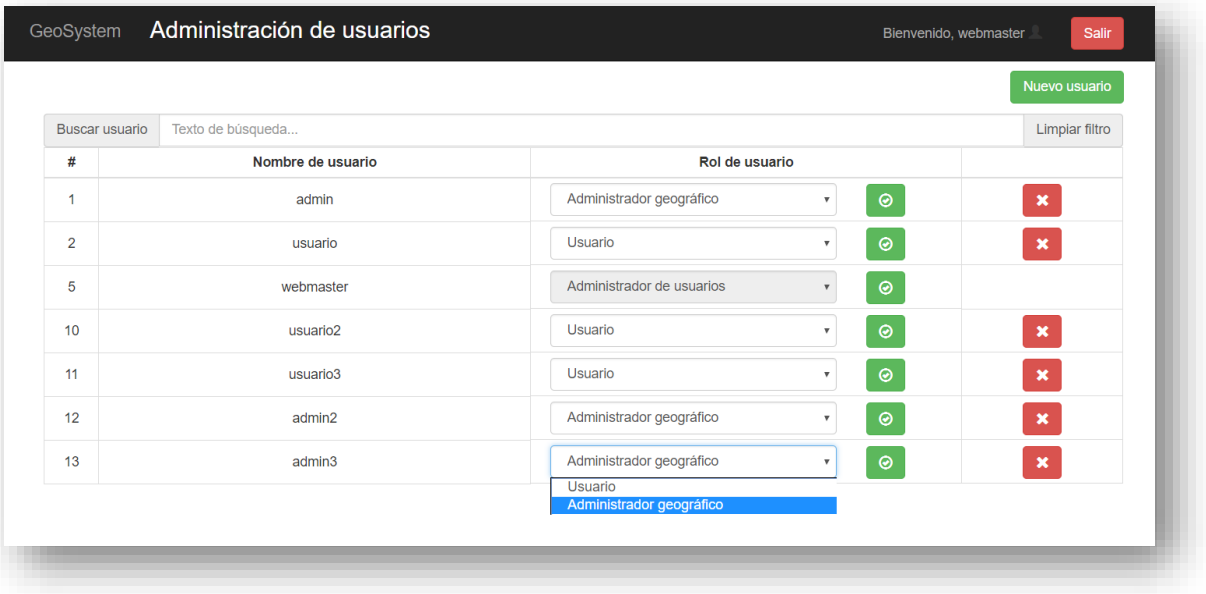


Ilustración 9: Módulo de administración de usuarios

6.2.1. /Views/AdminUsu/Index.cshtml

Incluye el componente `w-consulta-usuarios`.

6.2.2. /Views/AdminUsu/NuevoUsuario.cshtml

Incluye el componente `w-nuevo-usuario`.

6.2.3. /Controllers/AdminUsuController.cs

Está formado por las acciones `Index` y `NuevoUsuario`.

En este caso, a diferencia del controlador de Home, no se limitan a devolver la vista correspondiente, si no que también consultan que el usuario tenga los permisos pertinentes (redirigiéndole a Home en caso contrario).

Para ello, se crea un objeto de tipo `SecurityManager` (cuya definición se verá más adelante), del cual se usarán los métodos `CheckUserInSession` y `CheckUserHasProfile`. Ambos devuelven un tipo boolean.

El primero será true con que haya algún usuario autenticado.

Al segundo se le pasa como parámetro el nombre del módulo, `adminUsu` en este caso, devolviendo true o false según si ese es el rol del usuario.

```
bool user_check = manager.CheckUserInSession();
profile_check = manager.CheckUserHasProfile("adminUsu");
```

Si ambas comprobaciones dan resultados satisfactorios, se devolverá la vista del módulo de administración de usuarios.

6.2.4. /Controllers/AdminUsuRestController.cs

Está compuesto por seis métodos: `GetUsuarios`, `GetRoles`, `CambiarRol`, `CambiarRolUpdate`, `SaveUsuario`, y `DeleteUsuario`.

6.2.4.1. GetUsuarios

Este método se utiliza para obtener el listado de usuarios en el sistema.

Esta información se almacenará en una variable List (lista de objetos del mismo tipo) de tipos User.

```
List<User> usuarios = new List<User>();
```

La cadena SQL que se utilizará para solicitar esta información será `"SELECT * FROM usuarios ORDER BY id_usuario"`.

Finalmente, se devolverá el objeto List codificado en JSON.

6.2.4.2. GetRoles

La función de este método es obtener la lista de posibles roles que puede tener un usuario. Estos se almacenan en una tabla de la base de datos para facilitar la escalabilidad de la aplicación (solo es necesario añadir una tupla a la tabla si se quiere incluir un nuevo rol).

Este información se almacena en una List de tipo Rol. El tipo Rol está definido en el modelo de la aplicación como:

```
public class Rol
{
    [Key]
    public int id_rol { get; set; }

    public string codigo { get; set; }
    public string nombre { get; set; }

}
```

Con el resultado de la cadena `"SELECT * FROM roles"` se rellenará esta información, que se devolverá en formato JSON.

6.2.4.3. CambiarRol

Este método recibe como parámetro un string (`strParam`) en JSON, decodificándolo como primera acción.

Su función es cambiar el rol de un usuario, el que se le haya indicado en el parámetro recibido.

En primer lugar, es necesario comprobar si el usuario tiene mapas asignados, ya que, en caso afirmativo, no podría cambiar de rol. Esta circunstancia solo puede darse al pasar de usuario a administrador geográfico.

Para ello, se realiza la consulta `"SELECT * FROM usuarios_mapas WHERE id_usuario=" + param.id`, siendo param.id el identificador del usuario recibido.

Si se encuentran mapas, se devuelve una bandera que informa de ello, para posteriormente notificar al administrador de usuarios de que no puede realizar este cambio.

Si no se encuentran mapas, se llama al método CambiarRolUpdate, pasándole también la información del usuario.

6.2.4.4. CambiarRolUpdate

Este método es en el que propiamente se realiza el cambio de rol de un usuario.

Una vez decodificado el JSON y conectado a la base de datos, envía a la misma la siguiente sentencia `"UPDATE usuarios SET rol = '" + param.role + "' WHERE id_usuario = " + param.id;` siendo param.role el rol al que se quiere actualizar al usuario. Se devolverá si el resultado ha sido exitoso o no.

6.2.4.5. SaveUsuario

Este método se utiliza para guardar un nuevo usuario en el sistema.

Recibe como parámetro un JSON con el alias y el rol del usuario a crear. Con esta información, lanza contra la base de datos la sentencia `"INSERT INTO usuarios(alias, rol) VALUES ('" + param.nombre + "', '" + param.rol + "')"`.

Devuelve si el resultado ha sido exitoso o no.

6.2.4.6. DeleteUsuario

Este método recibe el identificador del usuario que se desea eliminar.

Tras decodificarlo y conectarse a la base de datos, ejecuta la sentencia `"DELETE FROM usuarios WHERE id_usuario = " + id_int`.

No es necesario lanzar el eliminado de los mapas de dicho usuario en la tabla correspondiente, la base de datos está configurada para realizarla automáticamente.

6.2.5. /Components/adminUsu/consultaUsuarios/consultaUsuarios.js

Este fichero contiene la directiva `wConsultaUsuarios`, perteneciente al módulo `adminUsu.consultaUsuarios`. Esta directiva está enlazada con su correspondiente vista, en otro fichero, mediante el atributo `templateUrl`.

Su controlador se define en el propio fichero mediente el atributo `controller`. Éste recibe los objetos `"$scope"`, `"$http"` y `"transformRequestAsFormPost"`. En este caso, no requiere el atributo `"$window"`, ya que no tiene ninguna interacción directa con la ventana del navegador (como, por ejemplo, redirigirla).

6.2.5.1. getUsuarios

Este método se encarga de llamar al método `GetUsuarios` del controlador y rellena la variable `$scope.usuarios` con el resultado obtenido. Gracias al uso del $scope, la vista puede mostrar estos valores directamente.

Tras realizar esto, llama al método `getRoles`.

6.2.5.2. getRoles

Este método se encarga de llamar al método `GetRoles` del controlador y rellena la variable `$scope.usuarios` con el resultado obtenido.

6.2.5.3. cambiarRol

Este método se encarga de llamar al método `CambiarRol`del controlador. Recibe como parámetro el usuario a modificar, el cual le pasa también al controlador.

Si el controlador responde que ese usuario tiene mapas asignados, muestra un mensaje que alerta de ello. En ambos casos, actualiza el listado de usuarios, ya sea para revertir el cambio en pantalla o para asegurar que se muestra el rol adecuado de cada usuario.

```
if (response.data == "user_conmapas")
{
    alert("El usuario que desea que sea administrador tiene mapas asignados. Debe
eliminar estas relaciones antes de realizar esta acción.")
    $scope.getUsuarios();
}
else
    $scope.getUsuarios();
```

Mensaje mostrado al tratar de modificar el rol de un usuario con mapas asignados:

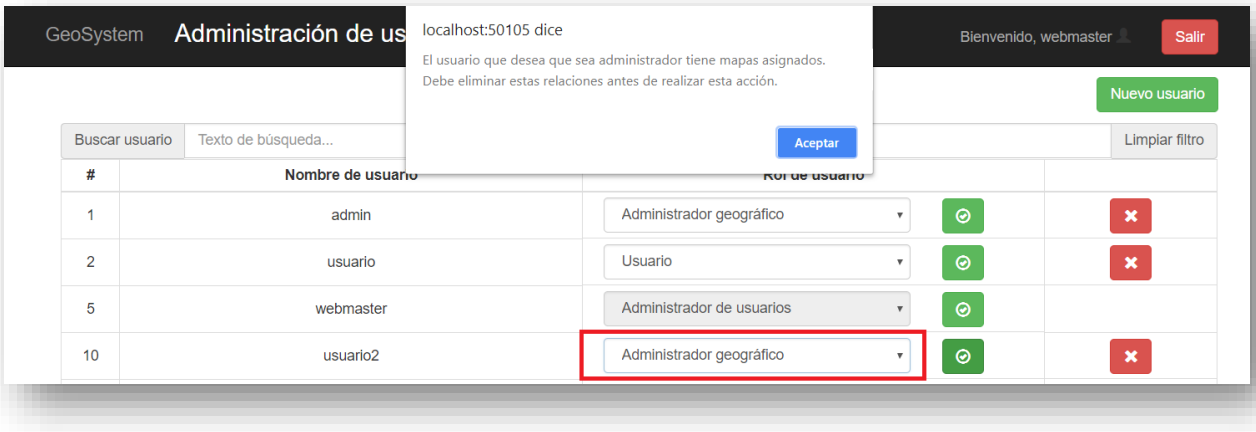


Ilustración 10: Alerta al tratar de hacer administrador a un usuario con mapas

Listado de usuarios recargado:

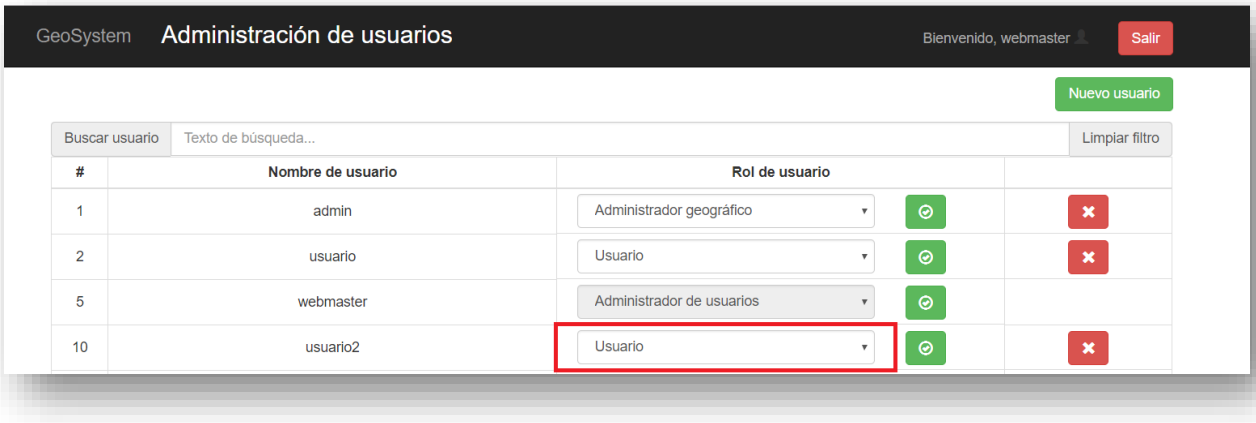


Ilustración 11: Modificación de rol de usuario exitosa

6.2.5.4. eliminarUsuario

Este método, que recibe el usuario que se desea eliminar, muestra en primer lugar un mensaje de confirmación del eliminado.

```
if (confirm("¿Está seguro de eliminar \"" + usuario.name + "\"?"))
```

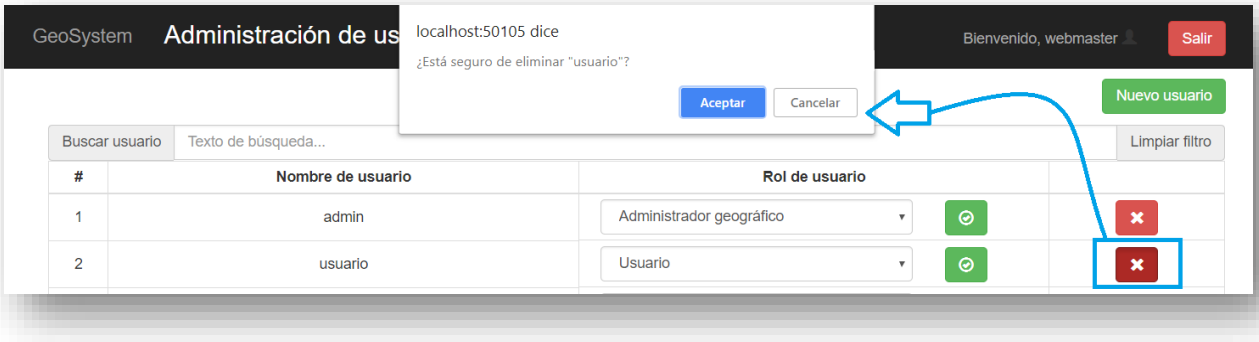


Ilustración 12: Alerta para la confirmación del eliminado de un usuario

En el caso de confirmar el eliminado, se muestra una nueva ventana informativa y se llama al método `DeleteUsuario` del controlador, pasándole el identificador del usuario a eliminar.

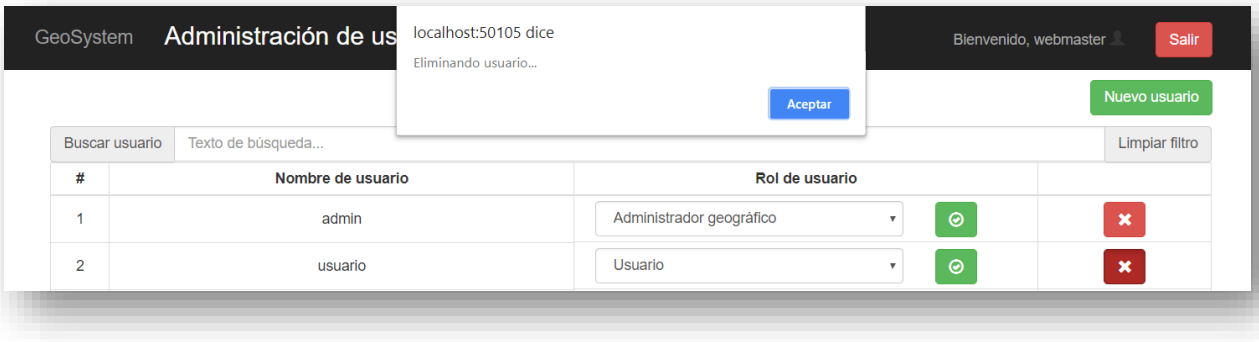


Ilustración 13: Mensaje informativo del eliminado de un usuario

Finalmente, se recarga el listado de usuarios haciendo uso del método `getUsuarios`.

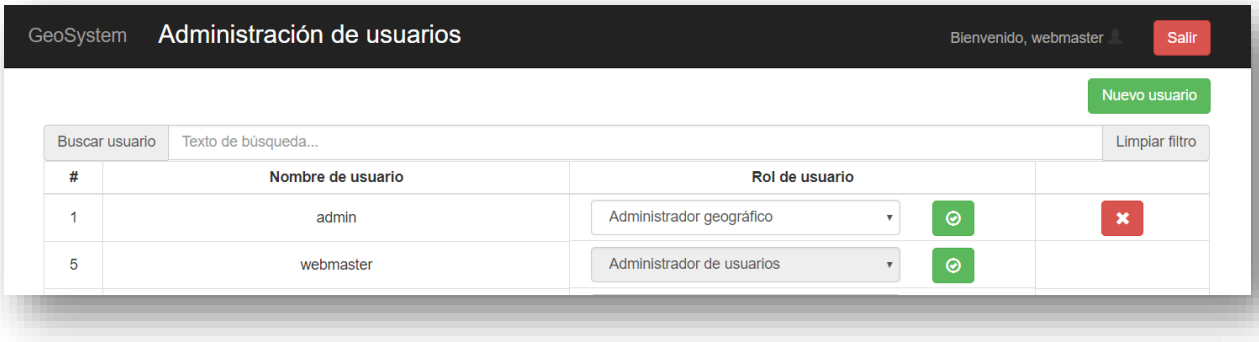


Ilustración 14: Eliminado de un usuario exitoso

6.2.5.5. resetFilter

Borra la cadena de texto que almacena el contenido que se introduzca en el formulario "Buscar usuario".

6.2.6. /Components/adminUsu/consultaUsuarios/templates/consultaUsuarios.html

En este fichero HTML está codificada la vista que se muestra por pantalla y se están los respectivos enlaces con los métodos del fichero Javascript.

Estos son los métodos a los que se llaman: getUsuarios cuando se carga el contenedor que aglutina a todos los campos, y el resto cuando se clica sobre el botón correspondiente.

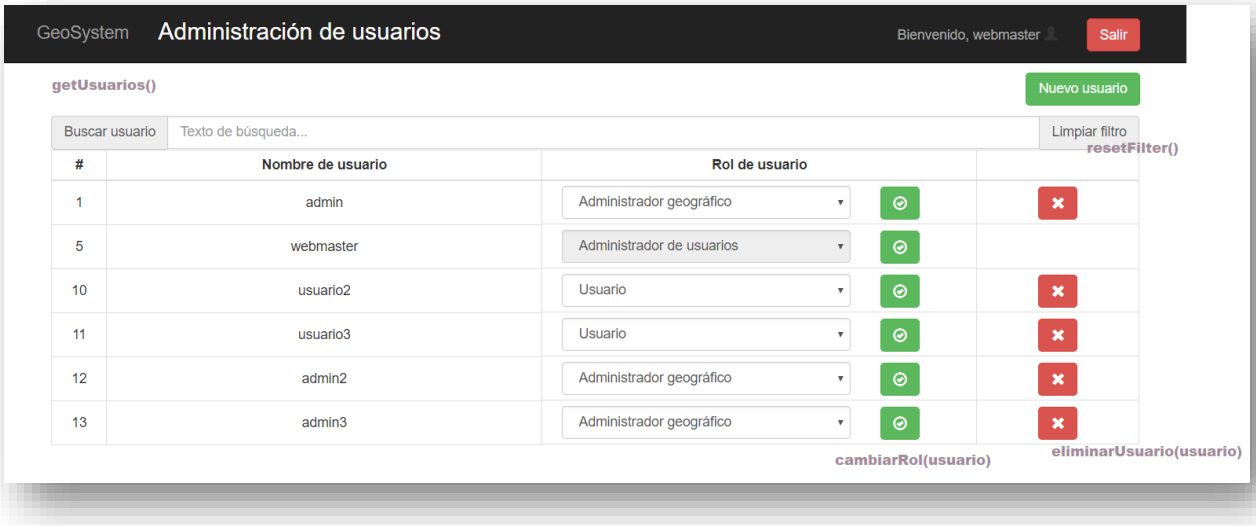


Ilustración 15: Relación entre la vista y los métodos Javascript en el panel de administración de usuarios

En cuanto al filtro, se guarda el texto introducido en el formulario en la variable $scope.searchText

```
<input type="text" class="form-control" placeholder="Texto de búsqueda..." ng-
model="searchText" />
```

Esta variable, en conjunto a la directiva ng-repeat (que repite un campo, en el caso que nos ocupa una fila de la tabla, tantas veces como elementos haya en la variable), nos permite filtrar los elementos mostrados en la tabla. La propia directiva consta del parámetro **`filter`**, en el que se le señala la variabla donde se guarda el texto a buscar.

Con `ng-repeat` se genera una fila por cada elemento, que estará formada por varias columnas.

1) Identificador del usuario
2) Nombre del usuario
3) Desplegable con el rol mostrándose y el resto de roles cuando se pulsa sobre él.

   Esta columna tiene dos desplegables dentro excepto por una directiva `ng-if`. En el caso de que el rol del usuario sea Administrador de Usuarios, el desplegable estará bloqueado, ya que solo hay uno en el sistema y no puede ser modificado.

   También es digno de mención el uso de `ng-if` en la etiqueta `option`, que evita que la opción de Administrador de Usuarios aparezca como opción.

   Por último, dentro de esta columna se encuentra el check verde que llamará al método correspondiente para cambiar el rol.
4) Botón para eliminar el usuario. También se usa `ng-if` para no mostrarlo en el caso del Administrador de Usuarios.

```
<tr ng-repeat="usuario in usuarios | filter:searchText">
    <td>{{usuario.id}}</td>
    <td>{{usuario.name}}</td>

    <td>
        <div class="col-sm-9">
            <select class="form-control" id="rol" ng-if="usuario.role!=adminUsu"
                                                        ng-model="usuario.role">

                <option ng-repeat="rol in roles" ng-value="rol.codigo"
                                ng-if="rol.codigo!=adminUsu">{{rol.nombre}}</option>
            </select>

            <select class="form-control" id="rol" ng-if="usuario.role==adminUsu"
                                                    ng-model="usuario.role" disabled>
                <option ng-repeat="rol in roles" ng-value="rol.codigo"
                                ng-if="rol.codigo==adminUsu">{{rol.nombre}}</option>
            </select>
        </div>

        <div class="col-sm-1">
            <a href="" class="btn btn-success glyphicon glyphicon-ok-circle"
                                    ng-click="cambiarRol(usuario);"></a>
        </div>
    </td>


    <td>
        <a href="#" ng-click="eliminarUsuario(usuario);" ng-if="usuario.role!=adminUsu"
                                class="btn btn-danger glyphicon glyphicon-remove"></a>
    </td>
</tr>
```

6.2.7. /Components/adminUsu/nuevoUsuario/nuevoUsuario.js

Este fichero contiene la directiva `wNuevoUsuario`, perteneciente al módulo `adminUsu.nuevoUsuario`. Esta directiva está enlazada con su correspondiente vista, en otro fichero, mediante el atributo `templateUrl`.

Su controlador se define en el propio fichero mediente el atributo `controller`. Éste recibe los objetos `"$scope"`, `"$http"` y `"transformRequestAsFormPost"`.

6.2.7.1. getRoles

Este método utiliza el método `GetRoles` del controlador para obtener el listado de posibles roles de usuario.

6.2.7.2. saveUsuario

Este método, en primer lugar, asegura que los campos de nombre y rol han sido rellenados, alertando al usuario en caso contrario.

```
if ($scope.data.nombre.length == 0) {
    alert("Introduzca un nombre");
} else if ($scope.data.rol == "") {
    alert("Seleccione un rol");
}
```

Al tratarse de tipos de formulario distintos, como se verá en el HTML, requieren de comprobaciones distintas. En un caso se comprueba que la longitud de la cadena sea mayor que 0, mientras que en otro se comprueba que no esté vacía.

Tras esto, se utiliza el método `SaveUsuario` del controlador.

6.2.8. /Components/adminUsu/nuevoUsuario/templates/nuevoUsuario.html

El fichero HTML define la vista de este formulario. Para el diseño del mismo, se utilizan las clases `panel` y `panel-default` de Bootstrap.

Estas clases permiten estructurar un panel con encabezado, cuerpo y pie, colocando en sus respectivos contenedores las clases `panel-heading`, `panel-body`, y `panel-footer`.

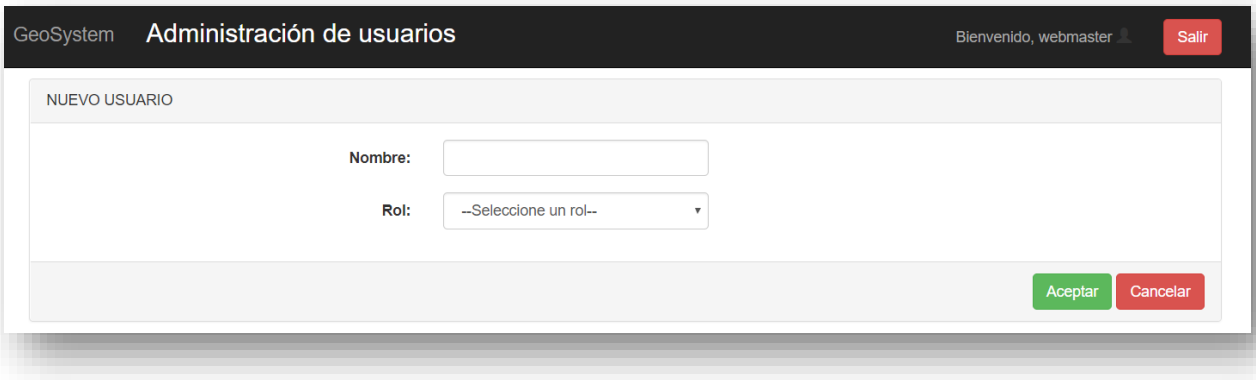


Ilustración 16: Panel con encabezado, cuerpo y pie utilizando Bootstrap

Ambos campos de información, nombre y rol, se almacenan en sendas variables para poder ser usados por el fichero Javascript con el uso de la directiva `ng-model`. Para obtener los posibles roles, se utiliza el método `getRoles()`del Javascript.

Por último, al clicar sobre Aceptar, con la directiva `ng-click`, se llama al método `saveUsuario()`.

6.3. Administración geográfica

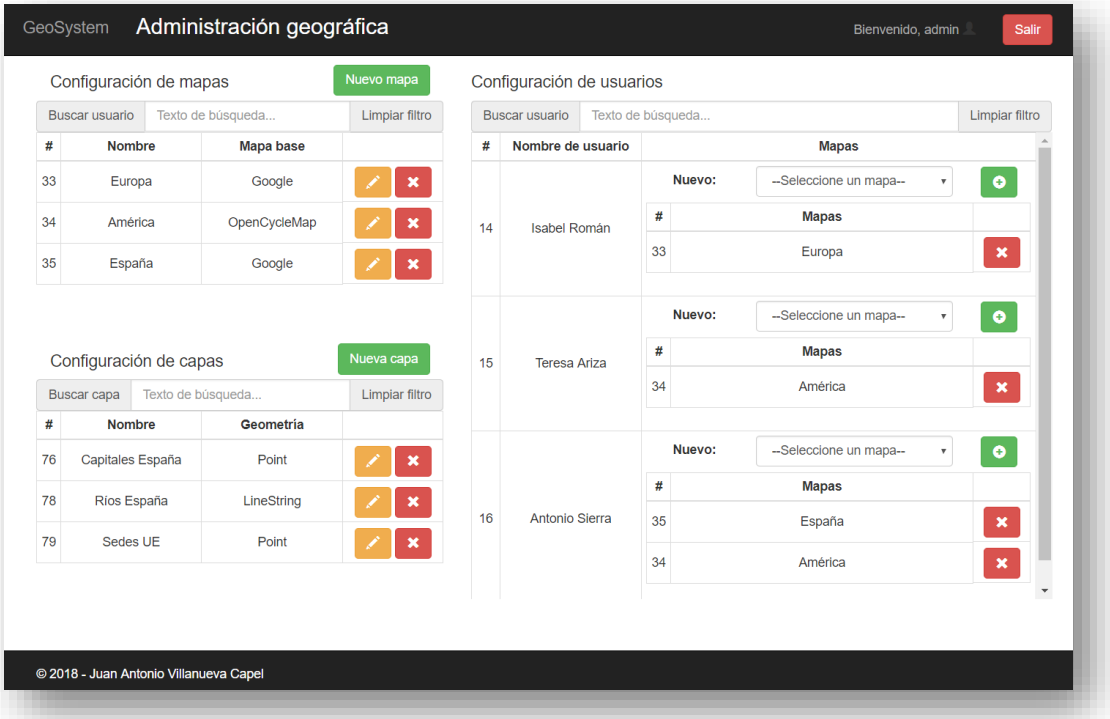


Ilustración 17: Módulo de administración geográfica

6.3.1. /Views/AdminGeo/Index.cshtml

Incluye una columna con los componentes `w-mapas` y `w-capas`, y otra con el componente `w-usuarios-mapas`.

6.3.2. /Views/AdminGeo/NuevoMapa.cshtml y /Views/AdminGeo/EdicionMapa.cshtml

Incluyen, respectivamente, los componentes `w-nuevo-mapa` y `w-edicion-mapa`.

6.3.3. /Views/AdminGeo/NuevaCapa.cshtml y /Views/AdminGeo/EdicionCapa.cshtml

Incluyen, respectivamente, los componentes `w-nueva-capa` y `w-edicion-capa`.

6.3.4. /Controllers/AdminGeoController.cs

Consta de una acción por cada vista del módulo, es decir, `Index`, `NuevoMapa`, `EdicionMapa`, `NuevaCapa` y `EdicionCapa`.

Al igual que en el controlador de `AdminUsu`, estas acciones no se limitan a devolver la vista correspondiente, si no que también consultan que el usuario tenga los permisos pertinentes (redirigiéndole a Home en caso contrario). Esta comprobación es idéntica a la antes explicada, pasando al método `CheckUserHasProfile` el parámetro `adminGeo` en este caso.

6.3.5. /Controllers/AdminGeoRestController.cs

Está compuesto por los siguientes métodos: `GetMapas`, `GetMapaById`, `GetCapasById`, `DeleteMapa`, `SaveMapa`, `GetIdMapa`, `UpdateMapa`, `GetCapas`, `SaveCapa`, `GetIdCapa`, `GetCapaById`, `GetCamposCapaById`, `SaveDato`, `UpdateCapa`, `DeleteCapa`, `GetUsuarios`, `GetUsuariosMapas`, `AddUsuarioMapa`, `DeleteUsuarioMapa` y `GetRoles`.

6.3.5.1. GetMapas

Este método se utiliza para obtener el listado de mapas en el sistema.

Esta información se almacenará en una variable List (lista de objetos del mismo tipo) de tipos Mapa.

```
List<Mapa> mapas = new List<Mapa>();
```

Tras abrir una conexión con la base de datos, la cadena SQL que se utilizará para solicitar esta información será `"SELECT * FROM mapas ORDER BY id_mapa"`.

Se recorrerá la respuesta obtenida rellenando los campos de cada objeto Mapa, que se irán añadiendo a la lista.

Finalmente, se devolverá el objeto List codificado en JSON.

6.3.5.2. GetMapaById

Este método se utiliza para obtener un mapa en concreto, aquel cuyo identificador es igual al que se la pasa como parámetro.

Siguiendo una secuencia similar al método anterior, en este caso la sentencia SQL será `"SELECT * FROM mapas WHERE id_mapa = " + id_int`, para id_int el parámetro que se le pasa al método tras transformarlo a tipo entero.

6.3.5.3. GetCapasById

Este método recibe como parámetro una cadena de caracteres (`id`) que es parseada a tipo entero.

Este valor será utilizado para obtener las capas incluídas en un determinado mapa, aquel con identificador igual al pasado como parámetro. Para ello, se utilizará la sentencia SQL `"SELECT * FROM capas_mapas WHERE id_mapa = " + id_int`.

6.3.5.4. DeleteMapa

La función de este método es eliminar del sistema aquel mapa cuyo identificador se corresponda con el que recibe como parámetro, valiéndose de la sentencia `"DELETE FROM mapas WHERE id_mapa = " + id_int`.

La base de datos está configurada para que, al eliminar un mapa, se eliminen en cascada las filas de la tabla capas_mapas correspondientes al mapa eliminado.

En cambio, si el mapa a eliminar tiene alguna relación usuario-mapa, al violarse la llave `usuarios_mapas_id_mapa_fkey`, se devolverá una bandera (`mapa_conusers`) para poder informar al usuario de que, antes de eliminar el mapa, debe desligarlo de estos usuarios.

6.3.5.5. SaveMapa

Este método recibe como parámetro un string (`strParam`) en JSON, decodificándolo como primera acción.

Este parámetro contiene: nombre, metadatos, mapabase y lista de capas.

En primer lugar, se crea el mapa en su correspondiente tabla, mediante la sentencia `"INSERT INTO mapas(nombre, metadatos, mapabase) VALUES ('" + param.nombre + "', '" + param.metadatos + "', '" + param.mapabase + "')"`.

En caso de que ya exista un mapa con el nombre introducido, se devuelve un indicador para facilitar la notificación al usuario de este error.

A continuación, se obtendrá el identificador del recién creado mapa utilizando el método `GetIdMapa` (que se describe a continuación) del controlador, al que se le pasa el nombre del mapa como parámetro.

Teniendo este identificador, ya es posible introducir en la tabla capas_mapas el listado de capas que se han seleccionado. Para ello, se recorre la lista de capas, ejecutando la siguiente sentencia en cada iteración `"INSERT INTO capas_mapas(id_capa, id_mapa) VALUES ('" + param.capas[i] + "', '" + id_mapa + "')"`.

6.3.5.6. GetIdMapa

Este método recibe como parámetro el nombre de un mapa y, mediante la consulta `"SELECT id_mapa FROM mapas WHERE nombre = '" + mapa + "'"`, obtiene y devuelve el identificador del mismo.

6.3.5.7. UpdateMapa

Este método tiene como fin actualizar los parámetros de un mapa y de sus capas vinculadas.

Recibe como parámetro un string (`strParam`) en JSON, decodificándolo como primera acción a realizar. Este incluye: nombre, metadatos, mapabase, identificador, capasNuevas y capasBorra.

En primer lugar, se actualiza la tabla mapas, mediante la sentencia `"UPDATE mapas SET nombre = '" + param.nombre + "', metadatos = '" + param.metadatos + "', mapabase = '" + param.mapabase + "' WHERE id_mapa = " + param.id_mapa`.

A continuación, se recorre la lista capasNuevas, insertándolas en la tabla capas_mapas de la base de datos utilizando `"INSERT INTO capas_mapas(id_capa, id_mapa) VALUES ('" + param.capasNuevas[i] + "', '" + param.id_mapa + "')"`.

Por último, se eliminan de la misma tabla aquellas coincidencias con la lista capasBorra, recorriéndola y utilizando `"DELETE FROM capas_mapas WHERE id_mapa = " + param.id_mapa + " AND id_capa = " + param.capasBorra[i]`.

Se devolverá un indicador de error si el nombre de mapa introducido ya existe en la base de datos.

6.3.5.8. GetCapas

Este método se utiliza para obtener el listado de capas en el sistema.

Esta información se almacenará en una variable List (lista de objetos del mismo tipo) de tipos Capa.

```
List<Capa> capas = new List<Capa>();
```

Tras abrir una conexión con la base de datos, la cadena SQL que se utilizará para solicitar esta información será `"SELECT * FROM capas ORDER BY id_capa"`.

Se recorrerá la respuesta obtenida rellenando los campos de cada objeto Capa, que se irán añadiendo a la lista.

Finalmente, se devolverá el objeto List codificado en JSON.

6.3.5.9. SaveCapa

Este método recibe como parámetro un string (`strParam`) en JSON, decodificándolo como primera acción a realizar. Este parámetro consta de: nombre, metadatos, tipo de geometría de sus elementos, campos de sus elementos y listado de elementos.

Su función es guardar en la base de datos una capa y los elementos que la componen.

En primer lugar, se creará la capa en su correspondiente tabla mediante `"INSERT INTO capas(nombre, metadatos, tipo_geom) VALUES ('" + param.nombre + "', '" + param.metadatos + "', '" + param.tipo_geom + "')";`.

A continuación, se obtendrá el identificador de la capa recién creada mediante el método `GetIdCapa` (procedimiento equivalente al llevado a cabo en SaveMapa).

Una vez obtenido este identificador, se empieza a construir la sentencia SQL mediante la cual resultará la tabla de esta capa (capa_X) [Implementación_1]. Para lograrla, se irán concatenando cadenas de texto a medida que se va recavando información sobre la misma, usando el método nativo `string.Concat(cadenaA, cadenaB)` [Implementación_2].

1. `"CREATE TABLE capa_" + id_capa + " (id_elemento integer NOT NULL PRIMARY KEY"`

   Nombre (capa_X) y primera columna de la tabla, que siempre será el identificador del elemento

2. `", " + columna + " character varying(20) DEFAULT ' '"`

   `columna` es cada campo de la tabla tras haber sustituido sus espacios por barras bajas. Este fragmento de cadena se sitúa dentro de un bucle que recorre la lista de campos.

3. `", geom geometry(#TIPO#));"`

   Tras los campos de los elementos, la última columna de la tabla es aquella en que se almacenará la geometría del dato. Esta columna es de tipo geometry(#TIPO#), siendo #TIPO# el seleccionado en el formulario.

Esta sentencia se ejecuta en la base de datos.

A continuación, se procederá a rellenar esta tabla específica de la capa con los datos geográficos introducidos en ella, para lo cual se seguirá un proceso similar de ir concatenando cadenas de texto hasta completar la sentencia SQL.

El formato de esta cadena variará según el tipo de la geometría, aunque en todos los casos comenzará con `"INSERT INTO capa_" + id_capa + " (id_elemento, geom) VALUES "` [Implementación_3].

- Punto

  En un bucle que recorre la lista de elementos geográfico, a cada iteración, se concatena el fragmento `"(" + id + ", ST_GeomFromGeoJSON('{\"type\":\"Point\",\"coordinates\":[" + x + "," + y + "]}'))"`.

  `id` es un entero que se va incrementando a cada iteración.

  `x` e `y` son las coordenadas del punto tras haber sido tratadas con el método String.Replace [Implementación_4] para que el número decimal se forme con un punto (y no con una coma) mediante `string x = "" + param.elementos[i][0]`. Esta transformación será necesaria en todos los tipos de geometría.

  `ST_GeomFromGeoJSON` es un método de PostGIS [Implementación_5] que recibe una geometría en formato JSON y devuelve un objeto geométrico compatible con la base de datos.

- Línea

  De igual manera, se trata de un bucle que recorre la lista de elementos, aunque, en este caso, cada elemento no está compuesto únicamente de un par de coordenadas X-Y.

  Así, en primer lugar, la iteración de elementos concatena el fragmento `"(" + id + ", ST_GeomFromText('LINESTRING(".`

  `ST_GeomFromText` es un método de PostGIS [Implementación_6] que devuelve un objeto geométrico habiendo recibido una representación WKT, codificación ASCII para describir objetos espaciales [Implementación_7].

  A continuación, se define un nuevo bucle. Una línea está formada por un conjunto de puntos, así que es necesario recorrer el listado de esos puntos para lograr definir la línea. Tras las transformaciones pertinentes, se concatena `x + " " + y`. A la finalización de esta iteración por los puntos de la línea, se cierra el elemento concatenando `")'))"`.

- Círculo

  En último lugar, en el caso de un círculo, se reciben las coordenadas de su centro y el radio del mismo. Recorriendo la lista de elementos y con las transformaciones necesarias, se concatenará cada elemento con la sentencia `"(" + id + ", ST_Buffer(ST_GeomFromGeoJSON('{\"type\":\"Point\",\"coordinates\":[" + x + "," + y + "]}'),"  + radio + "))"`.

  En el caso del centro, se sigue la misma sintaxis que en el caso del punto. En cuanto al radio, para éste se utiliza el método `ST_Buffer`. ST_Buffer es un método de PostGIS [Implementación_8] que devuelve una geometría que representa todos los puntos cuya distancia del centro es menos o igual a la distancia que se le pasa como parámetro.

  ```
  geometry ST_Buffer(geometry g1, float radius_of_buffer);
  ```

Finalmente, se ejecuta en la base de datos la cadena formada.

6.3.5.10. GetIdCapa

Este método recibe como parámetro el nombre de un capa y, mediante la consulta `"SELECT id_capa FROM capas WHERE nombre = '" + capa + "'"`, obtiene y devuelve el identificador de la misma.

6.3.5.11. GetCapaById

Este método se utiliza para obtener una capa en concreto, aquella cuyo identificador es igual al que se la pasa como parámetro. Esto se logrará mediante la sentencia SQL `"SELECT * FROM capas WHERE id_capa = " + id_int`, para id_int el parámetro que se le pasa al método tras transformarlo a tipo entero.

6.3.5.12. GetCamposCapaById

La funcionalidad de este método es obtener la información sobre los campos de una capa, es decir, obtener toda la información de la tabla `capa_#ID#`.

La sentencia SQL utilizada para ello es `"SELECT * FROM capa_" + id_int + " ORDER BY id_elemento"`.

A diferencia del resto de SELECT's a la base de datos, en este caso no se conoce a priori el nombre de las columnas de la tabla. Por ello, en primer lugar, se rellena un array con los nombres de las columnas, `string[] fieldnames`, utilizando el método de Npgsql `GetName()`.

A continuación, se crea un diccionario de datos (clave, valor): `dictionary = new Dictionary<string, string>()`. Por cada campo, se rellena con los nombres de cada columna y el valor de la misma.

Una vez finalizada la lectura del campo, este se almacena en una lista de diccionarios, para poder seguir recorriendo la tabla.

6.3.5.13. SaveDato

Este método guarda los datos de un elemento de una capa.

Recibe como parámetro las cadenas de texto `strIds` y `strParam`. La primera contiene los identificadores del elemento y de la capa. La segunda, los datos de este elemento.

La cadena strParam, codificada como JSON, se deserializa convirtiéndola en un diccionario de claves-elementos mediante `DeserializeObject<Dictionary<string, string>>` [Implementación_9]. Esto es necesario al desconocerse a nivel de servidor la clave de estos campos: es un campo creado por el usuario y que cambia en cada capa.

La cadena strIds sí puede ser deserializada como el resto de las vistas en otros métodos.

La sentencia SQL a ejecutar en la base de datos se irá creando concatenando los distintos elementos.

Así, como inicio de la sentencia tendremos `"UPDATE capa_" + ids.id_capa + " SET "` para fijar la tabla a la que apuntamos (la correspondiente a la capa a modificar). A continuación, se recorre el diccionario que contiene los datos, concatenando a cada iteración `entry.Key + " = '" + entry.Value`. Finalmente, se fija el elemento que se quiere modificar mediante `"' WHERE id_elemento = " + ids.id_elemento`.

Así, se logra salvar los metadatos (todas las columnas salvo el id y la geometría) de los elementos de la tabla capa_#ID#.

6.3.5.14. UpdateCapa

Este método se encarga de actualizar los metadatos propios de la capa (no los de sus elementos). Recibe un JSON, `strParam`, con esta información codificada.

Tras decodificarlo en la variable param, actualiza esta información mediante la sentencia `"UPDATE capas SET nombre = '" + param.nombre + "', metadatos = '" + param.metadatos + "', tipo_geom = '" + param.tipo_geom + "' WHERE id_capa = " + param.id_capa`.

6.3.5.15. DeleteCapa

Este método se encarga de eliminar una capa, recibiendo como parámetro su identificador.

El borrado de una capa consta de dos partes. Por un lado, hay que eliminar su registro en la tabla de capas, lo que se consigue mediante `"DELETE FROM capas WHERE id_capa = " + id_int`.

Por otro, hay que eliminar la tabla que almacena sus elementos, capa_#ID#. Para ello, se ejecuta `"DROP TABLE capa_" + id_in`.

6.3.5.16. GetUsuarios

Recupera la información de tabla de usuarios correspondiente a aquellos cuyo rol es el de 'usuario' mediante `"SELECT * FROM usuarios WHERE rol='usuario'"`. Esta información se almacena en una lista de objetos de tipo `User`.

Para rellenar esta lista, se va recorriendo el resultado de la consulta:

```
var resultado = new User();
resultado.id = (int)lector["id_usuario"];
resultado.name = (string)lector["alias"];


usuarios.Add(resultado);
```

6.3.5.17. GetUsuariosMapa

Recupera toda la información de tabla que relaciona usuarios y mapas. Para tener más información disponible, no se ejecuta la consulta sobre esta tabla (que solo tiene identificadores de usuarios y mapas), si no que se ha definido una vista SQL que también tiene el nombre del usuario y del mapa.

Así, se obtiene la información deseada mediante `"SELECT * FROM v_usuarios_mapas"`.

6.3.5.18. AddUsuarioMapa

Este método añade un elemento a la tabla que relaciona usuarios y mapas. Recibe un JSON codificado que incluye el identificador tanto del usuario como del mapa. Teniendo esta información, basta con realizar un `INSERT INTO usuarios_mapas(id_usuario, id_mapa) VALUES ('" + param.usuario + "', '" + param.mapa + "')`.

6.3.5.19. DeleteUsuarioMapa

Este método sigue una estructura similar al anterior, pero en este caso elimina un elemento de la tabla que relaciona usuarios y mapas. Teniendo ambos identificadores, se ejecuta `"DELETE FROM usuarios_mapas WHERE id_mapa = " + param.id_mapa + " AND id_usuario = " + param.id_usuario`.

6.3.5.20. GetGeometrias

La función de este método es obtener la lista de posibles geometrías que puede tener una capa. Éstas se almacenan en una tabla de la base de datos para facilitar la escalabilidad de la aplicación (solo es necesario añadir una tupla a la tabla si se quiere incluir una nueva geometría en el formulario).

Este información se almacena en una List de tipo Geo. El tipo Geo está definido en el modelo de la aplicación como:

```
public class Geo
{
    [Key]
    public int id_geo { get; set; }

    public string codigo { get; set; }
    public string nombre { get; set; }

}
```

Con el resultado de la cadena `"SELECT * FROM geometrias"` se rellenará esta información, que se devolverá en formato JSON.

6.3.6. /Components/adminGeo/mapas

Este componente está formado por los ficheros:

- /Components/adminGeo/mapas/templates/mapas.html
- /Components/adminGeo/mapas/mapas.js

En el fichero HTML, además de el código propio de su visualización, resultan especialmente dignos de mención los atributos indicados en la captura inferior:

Ilustración 18: Relación entre la vista y los métodos Javascript en el panel de configuración de mapas

El atributo href del botón de Nuevo Mapa redirige al usuario a otro componente para la creación del mismo.

Los atributos ng-click y ng-init utilizan funciones definidas en el fichero Javascript.

El atributo ng-model indica la variable del $scope en que se almacena la información introducida en esa caja.

Por último, el atributo ng-repeat se utiliza para representar cada línea del objeto que contiene los mapas, gestionando AngularJS esta iteración.

6.3.6.1. getMapas

Utiliza el método `GetMapas` del controlador para obtener el listado de mapas del sistema.

6.3.6.2. editarMapas

Utilizando el objeto `window.location` [Implementación_10], este método redirige al navegador a otra página web.

Dado que recibe el identificador del mapa a editar, la URL a la que redirige es `"/AdminGeo/EdicionMapa?id="` + `id`.

6.3.6.3. eliminarMapa

Se encarga de eliminar un mapa mediante una llamada al método `DeleteMapa` del controlador.

Antes de ello, informa al usuario, en primer lugar mediante una ventana de confirmación y a continuación mediante una alerta a modo informativo. Para esto se utilizan, respectivamente, los métodos `confirm` y `alert`.

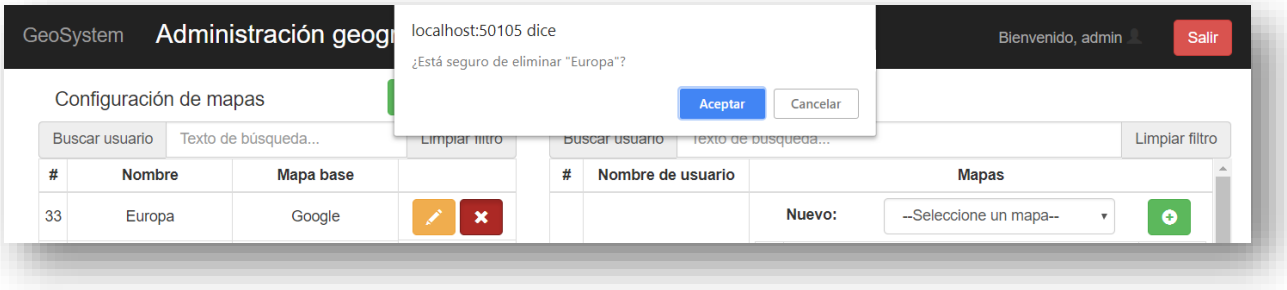


Ilustración 19: Alerta para la confirmación del eliminado de un mapa

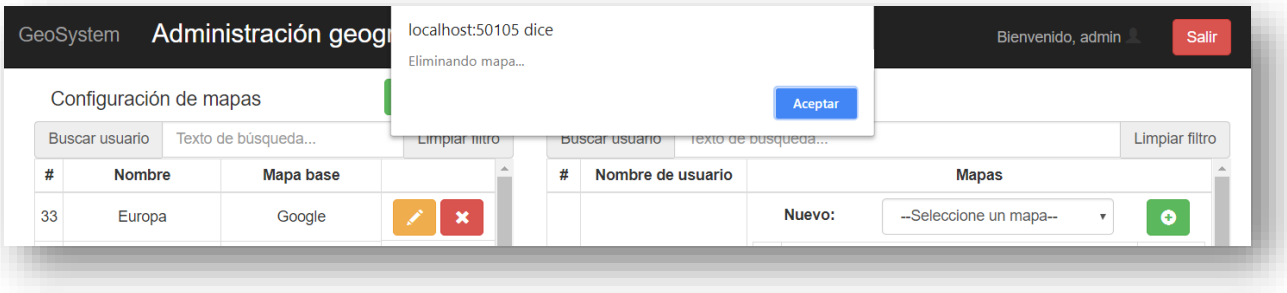


Ilustración 20: Ventana informativa del eliminado de un mapa

En función de la respuesta del método del controlador, existen dos posibilidades.

Si éste devuelve la bandera `"mapa_conusers"`, se notificará al usuario de que no es posible eliminar el mapa mientras éste esté vinculado a un usuario.

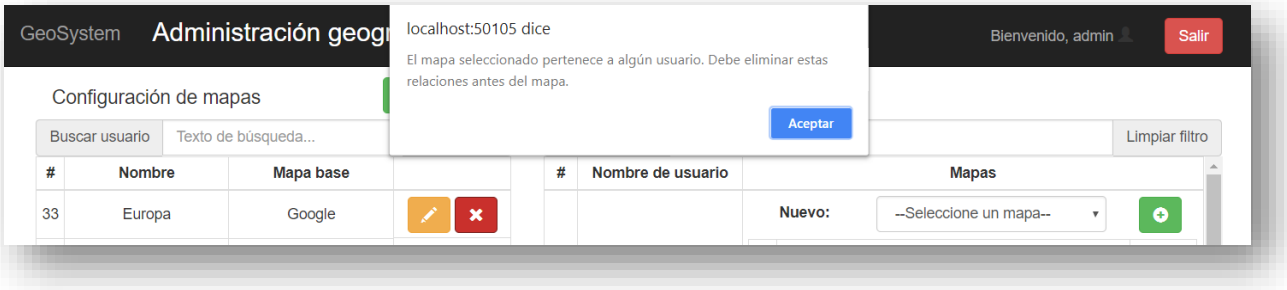


Ilustración 21: Alerta al tratar de eliminar un mapa relacionado con algún usuario

En caso contrario, se llamará al método getMapas para recargar ésta lista ya sin el mapa eliminado.

6.3.6.4. resetFilter

Este método se limita a vaciar la cadena `$scope.searchText` en la que se almacena el texto a buscar en el filtro.

6.3.7. /Components/adminGeo/nuevoMapa

Este componente, encargado de gestionar la creación de un nuevo mapa, está formado por los ficheros:

- /Components/adminGeo/ nuevoMapa /templates/nuevoMapa.html
- /Components/adminGeo/mapas/ nuevoMapa.js

En el fichero HTML, además de el código propio de su visualización, resultan especialmente dignos de mención los atributos indicados en la captura inferior:

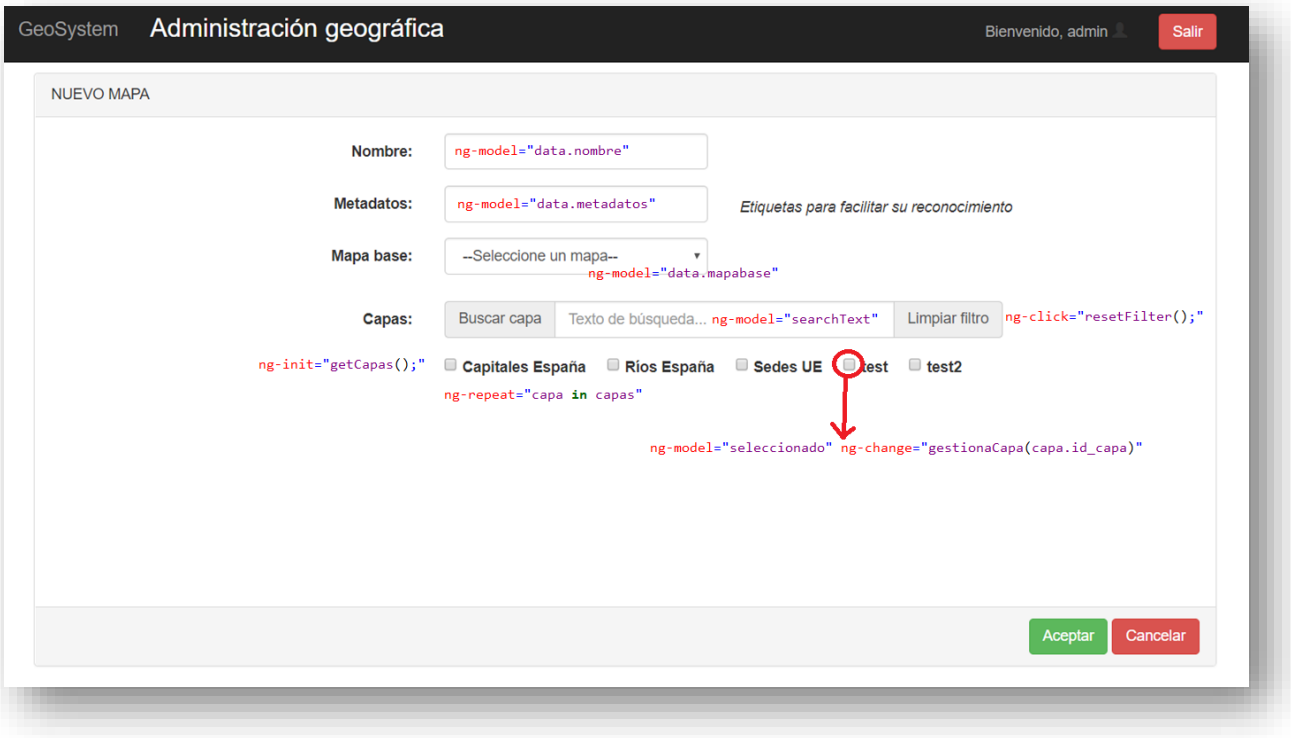


Ilustración 22: Relación entre la vista y los métodos Javascript en el panel de creación de un mapa

Los ng-model de nombre, metadatos y mapabase se utilizan para almacenar cada uno de los elementos rellenados en la variable $scope.data.

ng-init y ng-repeat se encaran de rellenar y de iterar en la variable $scope.capas.

En cuanto a los atributos de la caja de check, con ng-model se guarda en $scope.seleccionado el valor del mismo y con ng-change se llama al método gestionaCapa del fichero Javascript, pasándole el identificador de la capa, cada vez que el estado del check varía.

6.3.7.1. data

Definición de la variable en la que se guarda la información del nuevo mapa:

```
$scope.data = {
    nombre: "",
    metadatos: "",
    mapabase: "",
    capas: []
};
```

6.3.7.2. getCapas

Obtiene el listado de capas utilizando el método `GetCapas` del controlador.

6.3.7.3. saveMapa

Tras comprobar que todos los elementos del formulario están rellenos y que se ha seleccionado, al menos, una capa (y alertando al usuario en caso de que alguno de los campos esté vacío), guarda la información sobre el mapa utilizando el método `SaveMapa` del controlador.

Si éste devuelve la bandera `"nombre_repe"`, se notificará al usuario de que no es posible crear un mapa con este nombre por existir ya otro. En caso de éxito, se devolverá al usuario a la ventana principal del módulo mediante el método `window.history.back()`.

6.3.7.4. gestionaCapa

Este método recibe como parámetro el idenficador de la capa cuyo botón de check acaba de ser modificado.

Para saber qué acción realizar, es necesario conocer en qué estado se encontraba la capa, es decir, si ya estaba o no seleccionada. Para ello, se utiliza el método find:

```
var comprueba = $scope.data.capas.find(function (element) {
    return element == id;
});
```

Así, se busca en el array que almacena las capas asignadas al mapa el identificador de la capa que ha variado.

Si no se encuentra su identificador, se añade a éste array mediante el método push: `$scope.data.capas.push(id);`

En campo, si se encuentra, éste es borrado mediante el método splice: `$scope.data.capas.splice($scope.data.capas.indexOf(id), 1);`

6.3.8. /Components/adminGeo/edicionMapa

Este componente, encargado de gestionar la creación de un nuevo mapa, está formado por los ficheros:

- /Components/adminGeo/edicionMapa/templates/edicionMapa.html
- /Components/adminGeo/edicionMapa/edicionMapa.js

El fichero HTML resulta idéntico al del componente nuevoMapa salvo por dos salvedades:

- En los primeros elementos del fichero se incluyen los atributos `ng-init="getMapaById();"` y `ng-init="getParams();"`. En cambio, getCapas no es llamado desde el HTML (ya que ésta llamada se incluye al final del propio getMapaById).
- Se gestiona de forma distinta el check de cada capa. En este caso, se define como `<input type="checkbox" ng-model="capa.check">`.

6.3.8.1. getParams y parseQueryString

Su funcionalidad es extraer el identificador del mapa de la URL, que tiene la forma `http://#server#/AdminGeo/EdicionMapa?id=#IDENTIFICADOR#`.

`getParams` llama al método `parseQueryString` y asigna a la variable $scope.id el parámetro obtenido.

`parseQueryString` obtiene los parámetros de la URL mediante el objeto `window.location.search`. Mediante diversas transformaciones con expresiones regulares, se consigue extraer el identificador del mapa.

6.3.8.2. getMapaById

Obtiene los parámetros del mapa mediante el método `GetMapaById` del controlador y llama al método `getCapasById` del Javascript.

6.3.8.3. getCapas

Obtiene el listado de todas las capas del sistema mediante el método `GetCapas` del controlador y llama al método getCapas del Javascript.

6.3.8.4. getCapasById

En primer lugar, obtiene el listado de capas que actualmente tiene asignadas este mapa mediante el método `GetCapasById` del controlador.

A continuación, construye una variable `$scope.capasOriginales` con los identificadores de las mismas.

Por último, recorre el listado de todas las capas del sistema y el de las capas del mapa, comprobando a cada iteración si los idenficadores son iguales. En caso afirmativo, marca el atributo 'check' de la lista de todas las capas como true.

6.3.8.5. saveMapa

Este método se encarga de salvar el mapa editado mediante una llamada al método `UpdateMapa` del servidor.

Antes de ello, comprueba que el nombre y los metadatos no estén vacíos y contruye dos variables en las que gestionar qué capas se han añadido y cuáles se han eliminado. Esto se consigue recorriendo el listado de todas las capas y buscando estos identificadores en la variable `$scope.capasOriginales` que se rellenó al comienzo.

6.3.9. /Components/adminGeo/capas

Este componente está formado por los ficheros:

- /Components/adminGeo/capas/templates/capas.html
- /Components/adminGeo/capas/capas.js

En cuanto al fichero HTML, destacan:

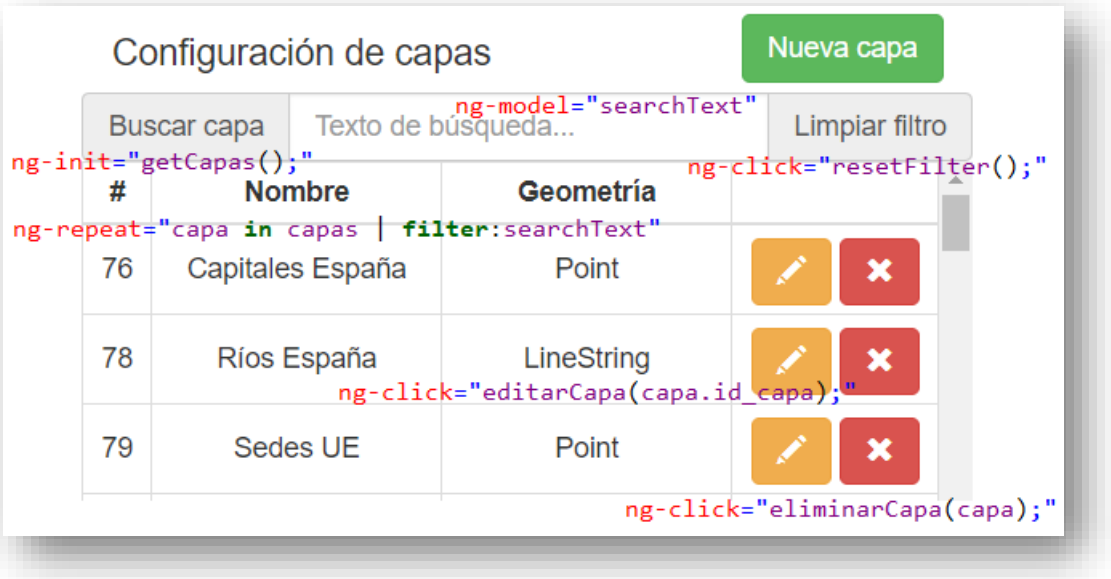


Ilustración 23: Relación entre la vista y los métodos Javascript en el panel de configuración de capas

Ya hemos visto una estructura similar en adminGeo/mapas. Los atributos ng-click de los botones redirigiran a sendos métodos.

Aprovechando que ya hemos visto cómo interaccionan HTML y Javascript en un componente de este tipo, también es digno de mención el estilo de éste fichero. Toda su estructura se define con clases Bootstrap. Así, por ejemplo, para el botón de editar se utiliza `class="btn btn-warning glyphicon glyphicon-pencil"`, clases que significan que se trata de un botón, de color amarillo, del juego Glyphicon y con un lápiz, respectivamente.

6.3.9.1. getCapas

Obtiene el listado de capas del sistema mediante una llamada al método `GetCapas` del controlador.

6.3.9.2. editarCapa

Recibe como parámetro el identificador de la capa que el usuario desea editar, y lo redirige a su correspondiente URL mediante `window.location = "/AdminGeo/EdicionCapa?id=" + id`.

6.3.9.3. eliminarCapa

Tras mostrar una ventana emergente para la confirmación del eliminado, realiza éste mediante una llamada al método `DeleteCapa` del controlador.

6.3.10. /Components/adminGeo/nuevaCapa

Este componente, encargado de gestionar la creación de una nueva capa, está formado por los ficheros:

- /Components/adminGeo/nuevaCapa/templates/nuevaCapa.html
- /Components/adminGeo/nuevaCapa/nuevaCapa.js

En el fichero HTML, además de el código propio de su visualización, resultan especialmente dignos de mención los atributos indicados en la captura inferior:

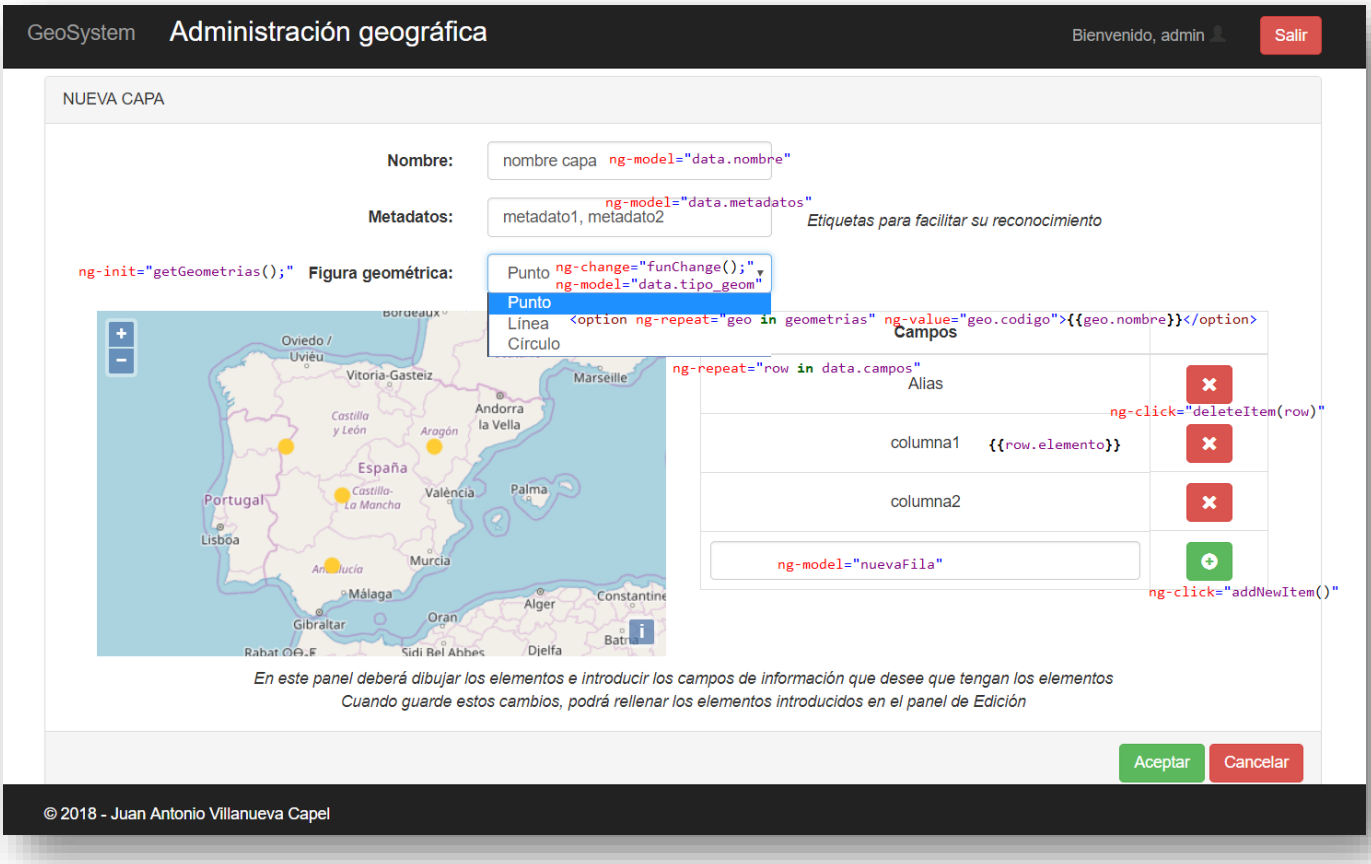


Ilustración 24: Relación entre la vista y los métodos Javascript en el panel de creación de capas

Los atributos ng-model de nombre, metatatos y tipo_geom almacenan estos valores en la variable $scope.data.

En cuanto al desplegable de figuras geométricas, está incluído en un contenedor que, al cargarse, llama al método getGeometrias() del Javascript, a través del cual obtiene la lista de posibles geometrías. Esta lista se despliega mediante el uso de un atributo ng-repeat que la recorre. Además, cuando varía su valor seleccionado, se llama al método funChange(), cuya funcionalidad se describirá a continuación.

A continuación, el componente se divide un panel para la administración de su geografía y otro para la de sus datos.

En el lado derecho se encuentra el de sus datos, con el listado de campos que se quiere tener para cada elemento de la capa. Éste se muestra, de nuevo, mediante un ng-repeat. Por defecto, se encuentra precargado el campo Alias, aunque éste puede ser eliminado, al igual que el resto, mediante el botón a su derecha, que llama al método deleteItem(). Siempre se sitúa en la última fila un campo de entrada vacío, cuyo valor se almacena en la variable $scope.nuevaFila y que puede incluirse en el listado si se pulsa el botón + de la derecha, que llama al método addNewItem().

En cuanto a la parte geográfica, en el lado izquierdo, se desarrolla dentro del mapa que se presenta usando OpenLayers.

Para la definición del mapa, se utiliza un mapa base de Open Street Maps y una capa de tipo vector sobre la cual se dibujará y almacenerán los elementos dibujados.

```
var raster = new ol.layer.Tile({
    source: new ol.source.OSM()
});

var source = new ol.source.Vector();
var vector = new ol.layer.Vector({
    source: source,
    style: new ol.style.Style({
        fill: new ol.style.Fill({
            color: 'rgba(255, 255, 255, 0.2)'
        }),
        stroke: new ol.style.Stroke({
            color: '#ffcc33',
            width: 2
        }),
        image: new ol.style.Circle({
            radius: 7,
            fill: new ol.style.Fill({
                color: '#ffcc33'
            })
        })
    })
});
```

Éste se centra sobre Madrid utilizando sus coordenadas y se vincula al contenedor de identificador `'map'` del fichero HTML.

```
var madridLonLat = [-3, 40];   //     40°24’59”N    03°42’09”W
var madridWebMercator = ol.proj.fromLonLat(madridLonLat);
var map = new ol.Map({
    layers: [raster, vector],
    target: 'map',
    view: new ol.View({
        center: madridWebMercator,
        zoom: 5
    })
});
```

Para que sea posible dibujar en él, se reutilizará el modelo de Draw and Modify Features de la documentación de OpenLayers [Implementación_11]. Usando las facilidades de AngularJS, podemos simplicar el código de ese ejemplo. Una vez definido el mapa, se logra poder dibujar en él añadiéndole una interacción.

Las variables draw y snap almacenan la definición de ésta interacción.

```
var draw, snap;
const typeSelect = document.getElementById('type');
```

La función addInteractions asigna valores a éstas variables. El tipo de figura geométrica de la interacción se introduce como parámetro en el atributo `type` de la variable `draw`. En el ejemplo recupera este valor mediante un `getElementById`, pero con AngularJS es posible almacenarlo en la variable `$scope.data.tipo_geom`. Además, como solo queremos un tipo de geometría en cada capa, se limpiará el contenido del mapa (se borrarán los elementos dibujados) a cada ejecución de la función, mediante `vector.getSource().clear()` [Implementación_12].

```
function addInteractions() {
  vector.getSource().clear();
  draw = new ol.interaction.Draw({
    source: source,
    type: $scope.data.tipo_geom
  });
  map.addInteraction(draw);
  snap = new ol.interaction.Snap({source: source});
  map.addInteraction(snap);

}
```

6.3.10.1. getGeometrias

Este método obtiene el listado de posibles formas geométricas mediante el uso del método `GetGeometrias` del controlador.

6.3.10.2. saveCapa

Este método guarda la capa creada mediante el uso del método `SaveCapa` del controlador, tras realizar previamente una serie de comprobaciones y transformaciones.

En primer lugar, se revisa que los campos de nombre, metadatos y figura geométrica están rellenos, notificando al usuario mediante una alerta en caso contrario.

A continuación, se recuperan las características del mapa mediante `vector.getSource().getFeatures()` [Implementación_13].

En el caso de que el formato de los elementos de la capa sean puntos o líneas (conjunto de puntos), se podrá obtener la geometría de cada uno de ellos directamente mediante `feature.getGeometry().A` [Implementación_14]. Las geometría de estas figuras se obtienen en el formato de proyección EPSG:3857 [Implementación_15 y Implementación_16], con lo que los valores X e Y que nos proporcionan no son coordenadas geográficas. Si queremos consultar en un mapa estos valores (con longitud y latitud), es necesario convertirlos EPSG:3857->EPSG:4326, lo cual puede realizarse en un conversor [Implementación_17] o transformándolas desde el propio OpenLayers [Implementación_18].

En cambio, si éstos son círculos, es necesario transformar estas variables, ya que lo que se almacena es su centro y su radio. Para ello, se utilizan los métodos `getRadius()` y `getCenter()`.

Una vez guardada la capa, se redirige el navegador del usuario a la ventana de edición de capa mediante `window.location = "/AdminGeo/EdicionCapa?id=" + response.data`.

6.3.10.3. funChange

Esta función elimina las interacciones de dibujo del mapa y llama al método `addInteractions()`, explicado anteriormente, para definir nuevas.

6.3.10.4. addNewItem

Este método incluye la nueva fila en la variable `$scope.data.campos` y luego limpia `$scope.nuevaFila` para dejarla preparada para incluir un nuevo campo.

6.3.10.5. deleteItem

Este método busca en `$scope.data.campos` el elemento que se desea borrar, y lo elimina mediante el uso del método JS splice [Implementación_19].

6.3.11. /Components/adminGeo/edicionCapa

Este componente, encargado de gestionar la edición de una capa, está formado por los ficheros:

- /Components/adminGeo/edicionCapa/templates/edicionCapa.html
- /Components/adminGeo/edicionCapa/edicionCapa.js

Actualmente, no es posible utilizar este módulo inmediatamente después de haber creado la capa, es necesario un paso manual para subir la tabla PostGIS creada en el servidor a Geoserver [Implementación_20]. Una vez subida, sí es posible utilizarlo con normalidad.

En el fichero HTML, además de el código propio de su visualización, resultan especialmente dignos de mención los atributos indicados en la captura inferior:

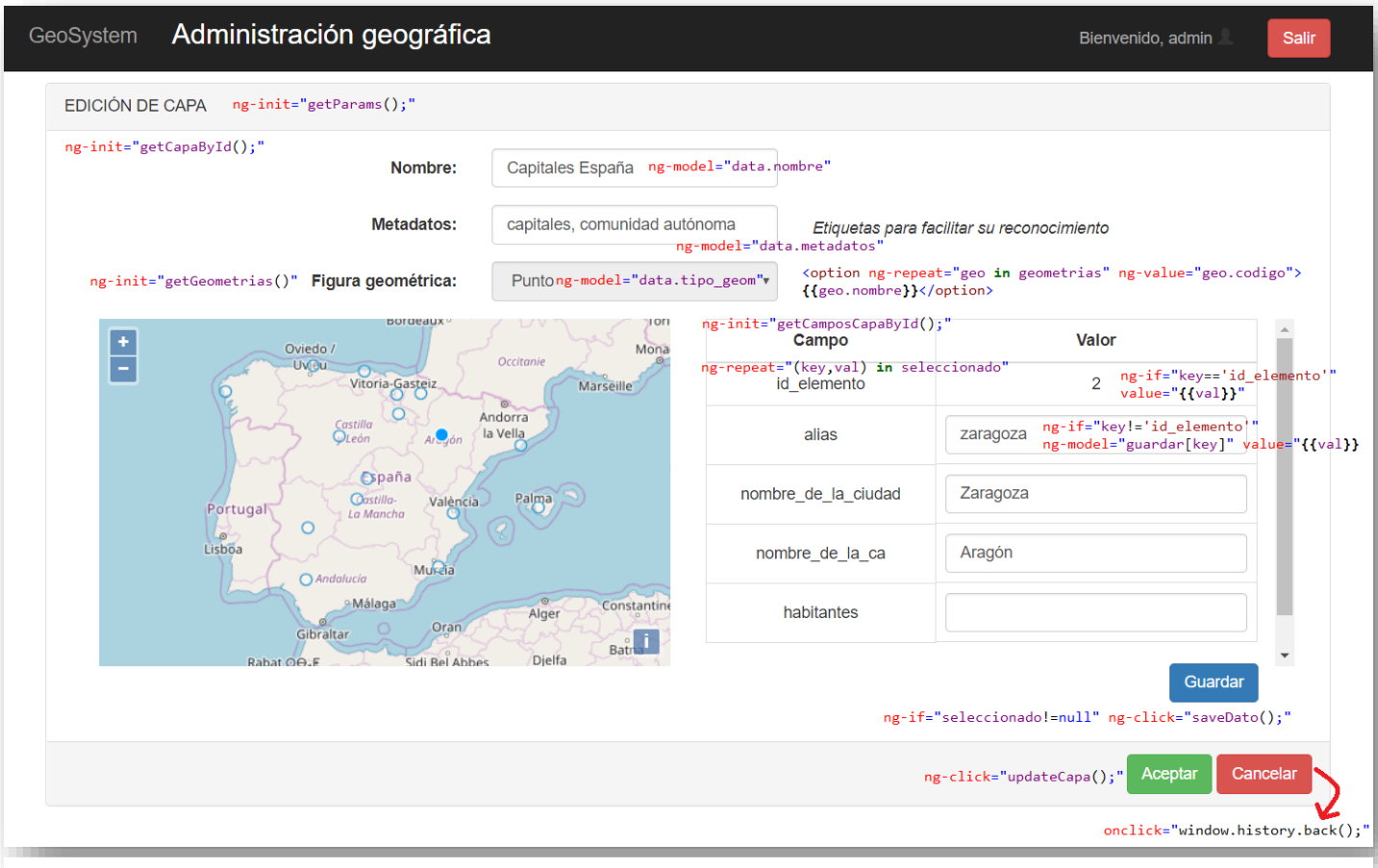


Ilustración 25: Relación entre la vista y los métodos Javascript en el panel de edición de capas

El aspecto más reseñable es la gestión de la tabla en que se muestran los metadatos de cada elemento. Esta tabla se despliega y activa cada vez que se pulsa sobre un elemento del mapa. Estos valores se almacenan en la variable $scope.seleccionado en cada ocasión.

6.3.11.1. getParams

Obtiene el identificador de la capa de la URL y monta la variable $scope.url, en la cual se almacena la ruta de la que obtener el mapa. Ésta tiene el formato `"http://localhost:8080/geoserver/tfg/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=tfg:capa_" + $scope.id + "&outputFormat=application%2Fjson"`.

6.3.11.2. getCapaById

Obtiene los atributos de la capa mediante una llamada al método `GetCapaById` del controlador.

Cuando obtiene la respuesta del controlador, llama al método `$scope.configMapa()` del fichero JS.

6.3.11.3. getCamposCapaById

Obtiene los atributos de cada elemento de la capa mediante una llamada al método `GetCamposCapaById` del controlador.

6.3.11.4. getGeometrias

Obtiene el listado de posibles geometrías para los elementos de la capa mediante una llamada al método `GetGeometrias` del controlador.

6.3.11.5. updateCapa

Actualiza en la base de datos los valores modificados en nombre y metadatos (los de los elementos se gestionan en saveDato) mediante una llamada al método `UpdateCapa` del controlador.

6.3.11.6. configMapa

Define el mapa, utilizando la URL obtenida en getParams y le añade una interacción para que se carguen los datos de cada elemento cuando es pulsado.

```
select_interaction.getFeatures().on("add", function (e) {
    var feature = e.element; //the feature selected

    // https://www.w3schools.com/jsref/jsref_substring.asp
    var id = feature.a.substring(feature.a.indexOf(".")+1);

    for (var i = 0; i < $scope.campos.length; i++)
    {
        if ($scope.campos[i].id_elemento == id)
        {
            $scope.seleccionado = $scope.campos[i];
            $scope.$apply();
            break;
        }
    }

    $scope.guardar = $scope.seleccionado;
    $scope.$apply();

});
```

A cada interacción, se busca en el listado de campos aquel que corresponda con el identificador del que ha sido pulsado, y se asigna su información a la variable $scope.seleccionado. Al tratarse de una modificación en tiempo de ejecución, es necesario ejecutar $scope.$apply() para que este cambio se vea reflejado en la vista.

6.3.11.7. saveDato

Utilizando la variable `$scope.guardar`, en la que se almacenan los cambios que se realizan sobre el formulario, y los identificadores de capa y elemento, se actualiza esta información en la base de datos mediante una llamada al método `SaveDato` del controlador.

6.3.12. /Components/adminGeo/usuariosMapas

Este componente, encargado de gestionar la tabla que relaciona usuarios y mapas, está formado por los ficheros:

- /Components/adminGeo/usuariosMapas/templates/usuariosMapas.html
- /Components/adminGeo/usuariosMapas/usuariosMapas.js

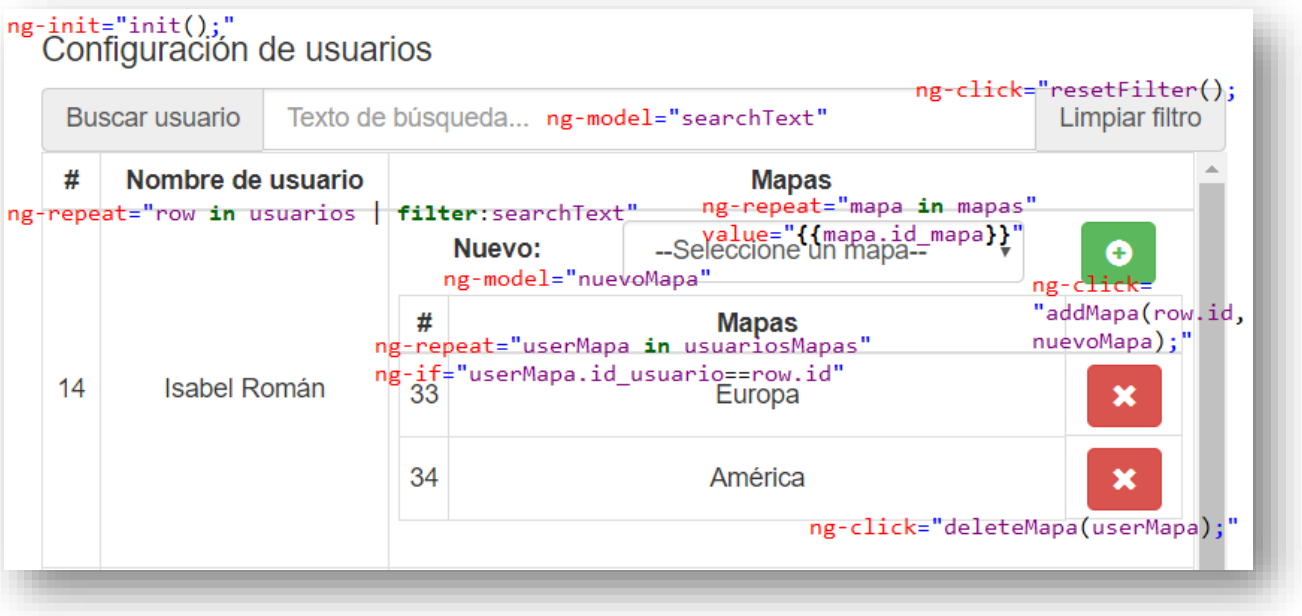


Ilustración 26:Relación entre la vista y los métodos Javascript en el panel de configuración de usuarios

En el caso del listado de mapas de cada usuario, se recorre la lista de usuariosMapas filtrando para que solo se muestren aquellos correspondientes al identificador de cada usuario.

6.3.12.1. init

Este método se encarga de llamar a getUsuarios(), getUsuariosMapas() y getMapas().

6.3.12.2. getUsuarios

Este método se encarga de recuperar el listado de usuarios mediante una llamada al método `GetUsuarios` del controlador.

6.3.12.3. getUsuariosMapas

Este método se encarga de recuperar el listado de relaciones usuario-mapa mediante una llamada al método `GetUsuariosMapas` del controlador.

6.3.12.4. getMapas

Este método se encarga de recuperar el listado de relaciones usuario-mapa mediante una llamada al método `GetMapas` del controlador.

6.3.12.5. addMapa

Este método construye una variable en la que almacenar la pareja usuario-mapa creada y lo añade en la base de datos mediante una llamada al método `AddUsuarioMapa` del controlador.

6.3.12.6. deleteMapa

Este método recibe como parámetro una variable en la que se encuentra la pareja usuario-mapa que se desea eliminar y lo borra en la base de datos mediante una llamada al método `DeleteUsuarioMapa` del controlador.

6.4. Visor

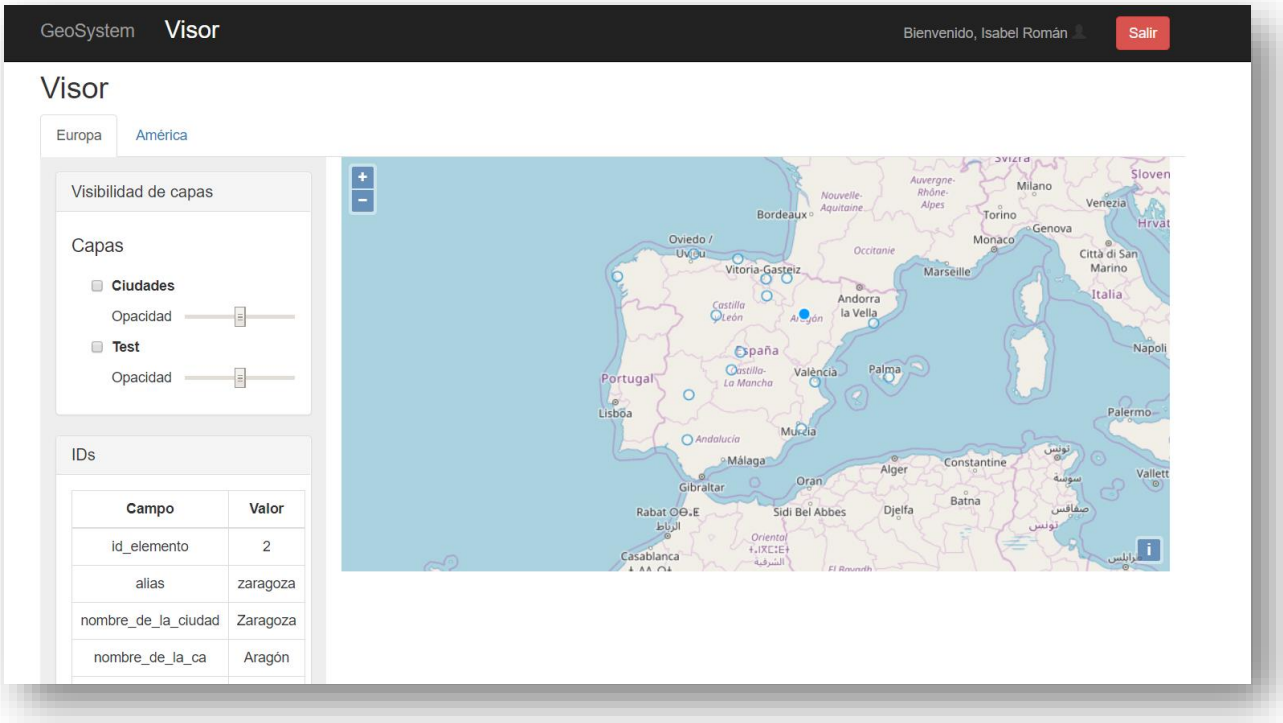


Ilustración 27: Módulo del visor

6.4.1. /Views/Maps/Visor.cshtml

Se trata del único módulo del sistema en cuya vista hay código HTML, y no solo la inclusión de otros componentes. Esto se debe a que, como el elemento mapa es común a todos los componentes, se ha decidido incluirlo en la vista (lo mismo sucederá con su código Javascript).

En primer lugar, se define la barra superior, en la que se despliegan los mapas disponibles para el usuario.

```
<ul class="nav nav-tabs" ng-init="getParams();">

    <li ng-class="{active: $index == id }" ng-click="click($index)"
        ng-repeat="row in mapas track by $index"><a ref="#">{{row.nombre_mapa}}</a>
    </li>


</ul>
```

Al iniciar la sesión, no hay ningún elemento preseleccionado (y, por tanto, no se muestra ningún mapa).

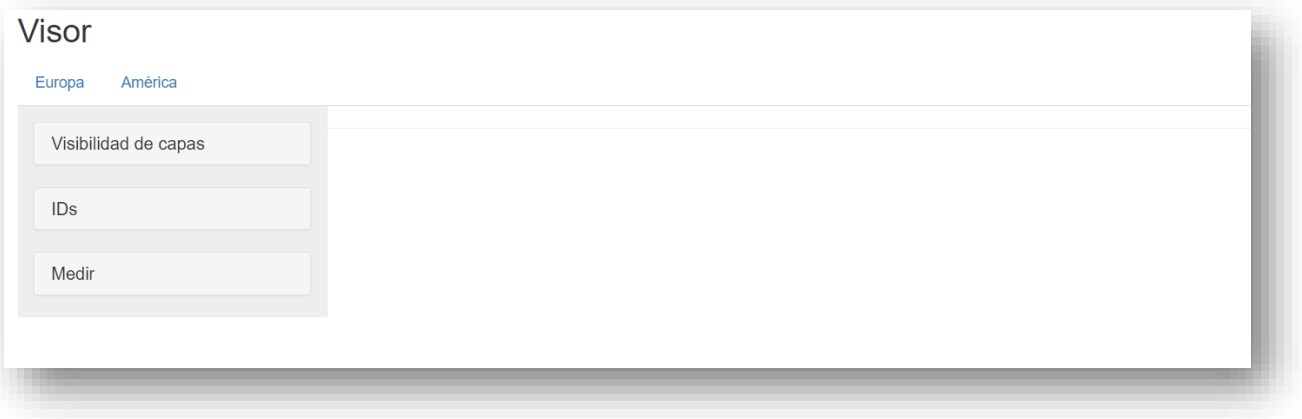


Ilustración 28: Módulo visor antes de seleccionar ningún mapa

En cuanto a las directivas usadas, su función es:

- ng-repeat: Recorre los elementos de un array. En este caso, incluye un parámetro no usado hasta ahora, `ng-repeat="row in mapas track by $index"`. Este parámetro permite utilizar el índice de la iteración [Implementación_21], con la sintaxis $index, en otras directivas.
- ng-class: Permite usar condiciones a la hora de asignar una clase [Implementación_22]. En este caso, si se cumple la condición de que el índice $index de la iteración coincide con el id de la página [Implementación_23], el elemento tendrá la clase 'active'.
- ng-click: Al clicar en el elemento, se llama al método click() con el índice `click` como parámetro.

Las herramientas laterales son incluidas mediante su respectivo componente:

- `<w-layers></w-layers>`
- `<w-identify></w-identify>`
- `<w-measure></w-measure>`

Por último, el mapa a mostrar se selecciona dinámicamente gracias al uso de una directiva Angular:

```
<div ng-attr-id="{{ 'map' + id }}" class="col-sm-9" crossOrigin='anonymous' ></div>
```

Los mapas de OpenLayers se muestran vinculando el atributo 'target' de su definición en Javascript con su 'id' en HTML. Así, desde el lado de la vista, la directiva `ng-attr-id` [Implementación_24] permite generar el identificador de este contenedor en función de la variable $scope.id de la página [Implementación_25].

6.4.2. /Controllers/MapsController.cs

Consta de una sola acción, `Visor`, al tener el módulo una única vista.

Al igual que en el resto de controladores, esta acción no se limita a devolver la vista correspondiente, si no que también consulta que el usuario tenga los permisos pertinentes (redirigiéndole a Home en caso contrario).

Para ello, como anteriormente, se usa el método `CheckUserHasProfile`, que recibe en este caso el parámetro `usuario`.

6.4.3. /Controllers/MapsRestController.cs

Está compuesto por los métodos `GetUsuariosMapasById` y `GetCamposCapaById`.

6.4.3.1. GetUsuariosMapasById

Este método obtiene la lista de mapas y capas de un usuario a partir de la información de la tabla `VUsuarioMapaCapa`.

A diferencia de en otros métodos, en este caso el identificador del usuario no se recibe como parámetro, si no que se obtiene del atributo de la sesión `System.Web.HttpContext.Current.Session["userId"]`.

Teniendo este parámetro, se realiza la consulta `"SELECT * FROM v_usuarios_mapas_capas WHERE id_usuario = " + id_int` a la base de datos.

Se recorre la información obtenido rellenando una variable de tipo `List<VUsuarioMapaCapa>`.

El objeto `VUsuarioMapaCapa` del modelo consta de `id_mapa`, `id_capa`, `nombre_mapa`, `mapabase`, `nombre_capa` y `tipo_geo`.

6.4.3.2. GetCamposCapaById

Este método recibe como parámetro los identificadores de capa y de elemento de la capa.

Con ellos, realiza una consulta a la tabla capa_#ID# de la forma `"SELECT * FROM capa_" + id_int + " WHERE id_elemento = "+item_int`.

Como sucedía en el módulo de administración geográfica, no es posible definir un modelo para recuperar estos elementos, ya que su nombre se fija por el usuario durante la creación de la capa.

Por ello, se utiliza una matriz con formato `List<Dictionary<string, string>>`, en la cual se guarda como clave el nombre de la columna mediante el método `lector.GetName(i)` [Implementación_26], que se almacena en la variable fieldnames. Esta misma variable es utilizada para encontrar el valor pareja de cada clave, obteniéndolo con el uso de `(string)lector[fieldnames[j]]`.

6.4.4. /Components/visor/app.js

El fichero app.js es aquel en el que se define el controlador de AngularJS y el método `transformRequestAsFormPost` para interactuar con la base de datos.

Todos los módulos del sistema tienen un fichero app.js, pero es el del módulo Visor el que cobra una mayor importancia, ya que, en este caso, no solo se utiliza para estas definiciones internas de AngularJS. Como todos del módulo interactúan con el mapa, éste es definido en este fichero.

Está formado por los métodos `getParams`, `click`, `getUsuariosMapas`, `construyeMapa` y `getCamposCapaById`.

Antes de la definición de estos métodos, se definen las variables globales `madridLonLat` y `madridWebMercator`, que se utilizarán para situar en las coordenadas de Madrid el centro de los mapas a crear:

```
var madridLonLat = [-3, 40];   //      40°24’59”N    03°42’09”W
var madridWebMercator = ol.proj.fromLonLat(madridLonLat);
```

El método fromLonLat convierte de coordenadas a la proyección Web Mercator usada por OpenLayers por defecto [Implementación_27].

6.4.4.1. getParams

Extrae de la URL, mediante el atributo `window.location.search`, el identificador del mapa a visualizar, y lo guarda en la variable $scope.id.

6.4.4.2. click

Recibe un identificador de mapa como parámetro, y envía el navegador del usuario a la URL en que visualizarlo mediante el uso de `window.location = "/Maps/Visor?id=" + id`.

6.4.4.3. getUsuariosMapas

Este método, en primer lugar, obtiene el listado de mapas y capas del usuario autenticado a través de una llamada al método `GetUsuariosMapasById` del controlador.

Obtenida esta respuesta, itera con los datos hasta conseguir un array de variables con el siguiente formato:

```
{ id_mapa: $scope.datos[i].id_mapa, nombre_mapa: $scope.datos[i].nombre_mapa,
    capas: [$scope.datos[i].id_capa] }
```

La forma de lograrlo es ir comprobando si en cada línea de los datos obtenidos el nombre del mapa es o no el mismo que el de la anterior línea.

A la finalización de esta tarea, llama al método `construyeMapa()`.

6.4.4.4. construyeMapa

Este método recorre la variable `$scope.mapas`, obtenida en el anteior método, para construir cada uno de los métodos asignados al usuario autenticado.

En cada iteración:

1. Se define el mapa:

```
var mapa = new ol.Map({
    layers: [raster],           // Mapa base (OSM o Google)
    target: 'map' + j,          // Atributo id de HTML al que se vincula
    view: new ol.View({         // Centro y zoom de la vista
        center: madridWebMercator,
        zoom: 5
    })
});
```

2. Se recorre el listado de capas, asociando URL:

```
for (var k = 0; k < $scope.mapas[j].capas.length; k++) {
    var capa = new ol.layer.Vector({            // Capa de tipo vector
        source: new ol.source.Vector({          // Fuente del vector
            format: new ol.format.GeoJSON(),    // Formato de la info del vector
            url:
"http://localhost:8080/geoserver/tfg/ows?service=WFS&version=1.0.0&request=GetFeature&typ
eName=tfg:capa_" + $scope.mapas[j].capas[k] + "&outputFormat=application%2Fjson"
            // URL de la información en Geoserver

        })
    });
    mapa.addLayer(capa);        // Adición de la capa al mapa
}
```

3. Añadir interacción al mapa:

```
var select_interaction = new ol.interaction.Select();

select_interaction.getFeatures().on("add", function (e) {

    $timeout(function () {   // Refresca la vista a la finalización de la función
        var feature = e.element;        // Datos del elemento seleccionado
        var capa = feature.a.split("_")[1].split(".")[0];   // ID Capa
        var item = feature.a.split("_")[1].split(".")[1];   // ID Elemento
        $scope.getCamposCapaById(capa, item);   // Método que obtiene la info
    });

});


mapa.addInteraction(select_interaction);    // Adición al mapa
```

6.4.4.5. getCamposCapaById

Método que recibe como parámetro el identificador de la capa y del elemento y obtiene su información mediante la llamada al método `GetCamposCapaById` del controlador.

Asigna a la variable `$rootScope.seleccionado` ($rootScope para que sea heredado por el resto de $scope's del controlador [Implementación_28]) la información obtenida para que pueda ser mostrada en la herramienta de identificación.

6.4.5. /Components/visor/identify

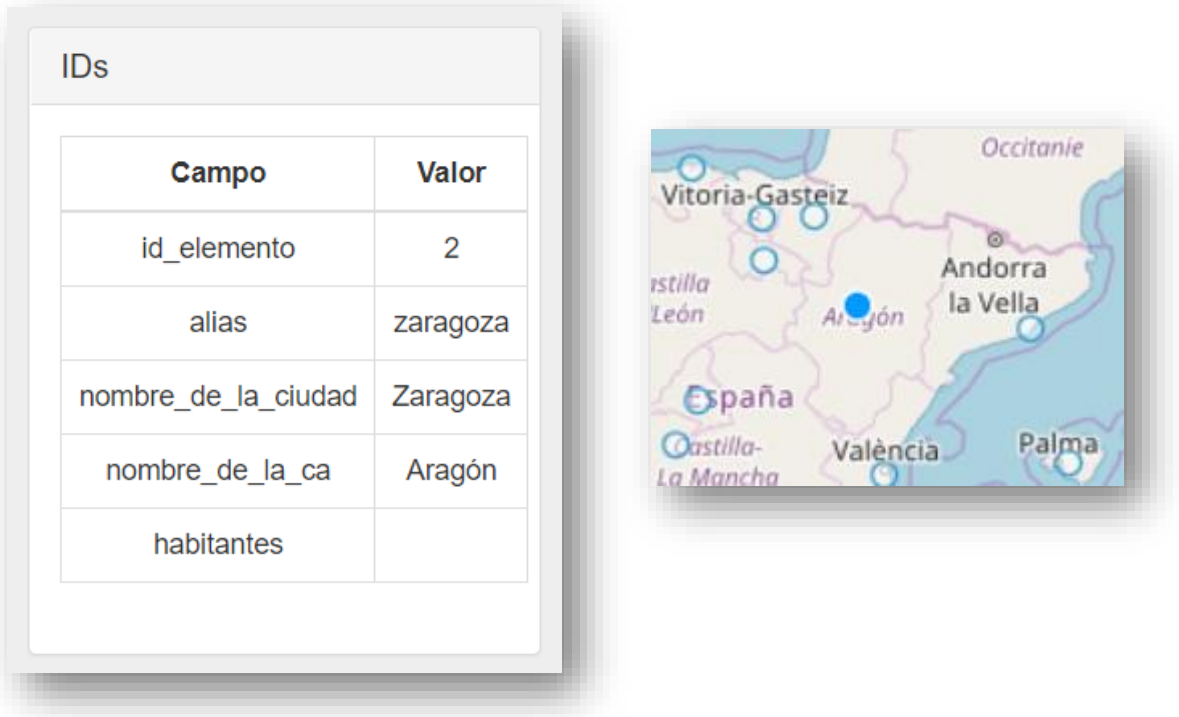


Ilustración 29: Herramienta para identificar elementos del visor

En su HTML se define un panel colapsable usando clases de Bootstrap. Dentro de él, se recorren las claves y valores de la variable $scope.seleccionado:

```
<tr ng-repeat="(key,val) in seleccionado">
    <td>{{key}}</td>
    <td >{{val}}</td>

</tr>
```

No utiliza un código JavaScript propio, ya que esta variable se rellena en el propio app.js.

6.4.6. /Components/visor/layers

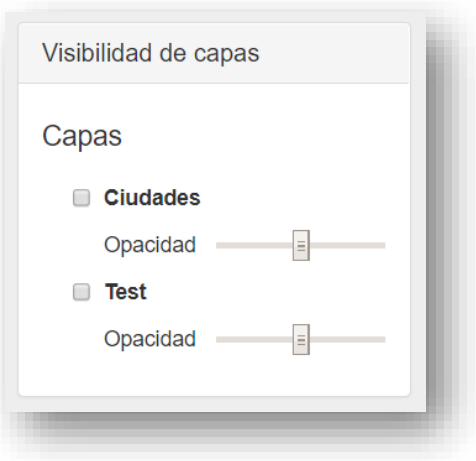


Ilustración 30: Herramienta para gestionar las capas visibles en el visor

En su HTML se define un panel colapsable usando clases de Bootstrap. Dentro de él, se recorre el listado de capas del mapa.

En su fichero Javascript, se configuran eventos para controlar cambios en la vista.

- Visibilidad

```
visibilityInput.on('change', function () {
    layer.setVisible(this.checked);
});
```

- Nivel de opacidad

```
opacityInput.on('input change', function () {
    layer.setOpacity(parseFloat(this.value));
});
```

6.4.7. /Components/visor/measure

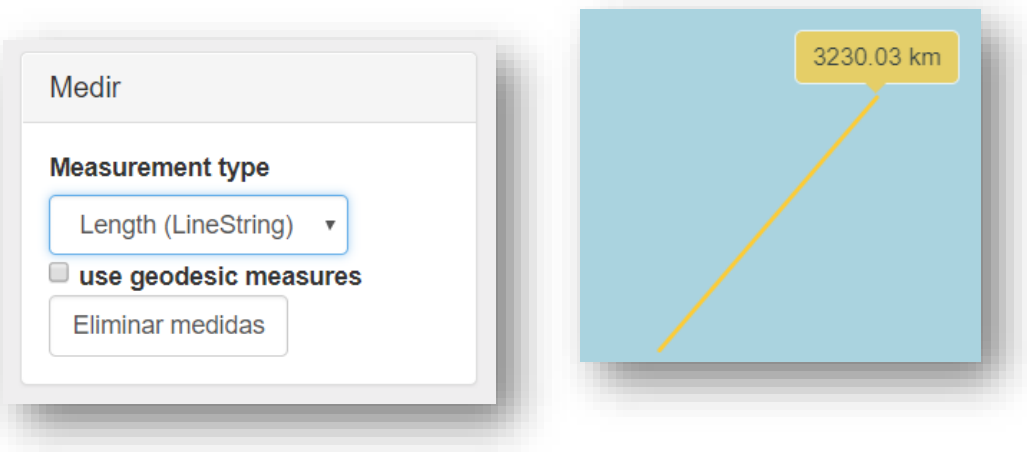


Ilustración 31: Herramienta para realizar medidas en el visor

En su HTML se define un panel colapsable usando clases de Bootstrap. Dentro de él, se incluye un desplegable que, a cada cambio de su valor (ng-change), llama a la función `tipo()` y que guarda su valor (ng-model) en la variable `"valueTipo"`.

También se define un botón que, al ser pulsado (ng-click), llama al método `eliminar()`.

En cuanto al fichero Javascript, se definen al inicio sendas interacciones para poder realizar estas medidas.

Los métodos tipo() y eliminar() eliminan estas interacciones y los valores ya guardados (dibujados) en ella, cambiando además el primero de ellos el tipo de interacción a definir.

7 PRUEBAS

"No he fallado, solo he encontrado 10.000 maneras que no funcionan"

- Thomas A. Edison -

Una vez implementado el proyecto, se ha realizado la batería de pruebas tanto unitarias como de sistema que a continuación se enumera.

7.1. Autentificación

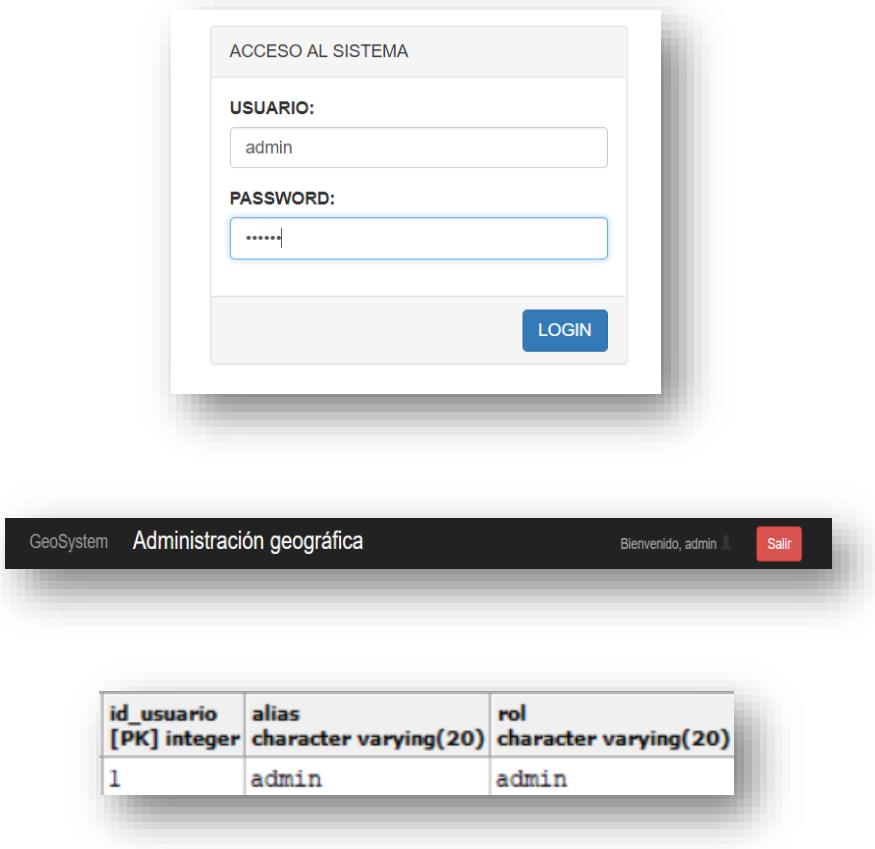
PR-AUT-001	Autenticación correcta
Descripción	Se introducen en el formulario de autenticación los datos de un usuario existente y se le dirige al módulo adecuado.
Comprobación 1	

Ilustración 32: Prueba 1 de Autenticación correcta

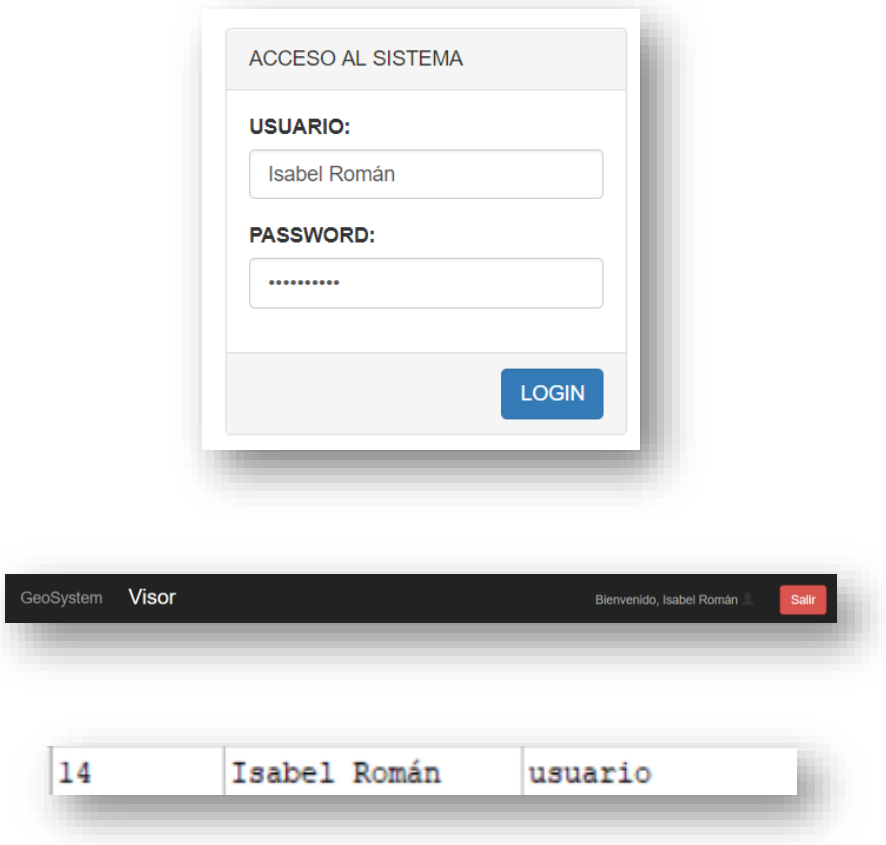
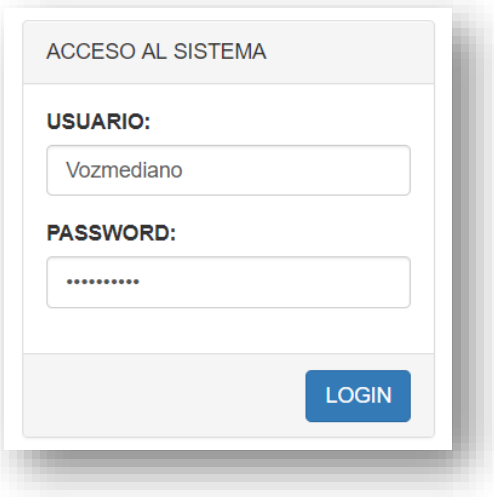
Comprobación 2	

Ilustración 33: Prueba 2 de Autenticación correcta

PR-AUT-002	Autenticación errónea
Descripción	Se introducen en el formulario de autenticación los datos de un usuario no existente y se le notifica del error.
Comprobación	

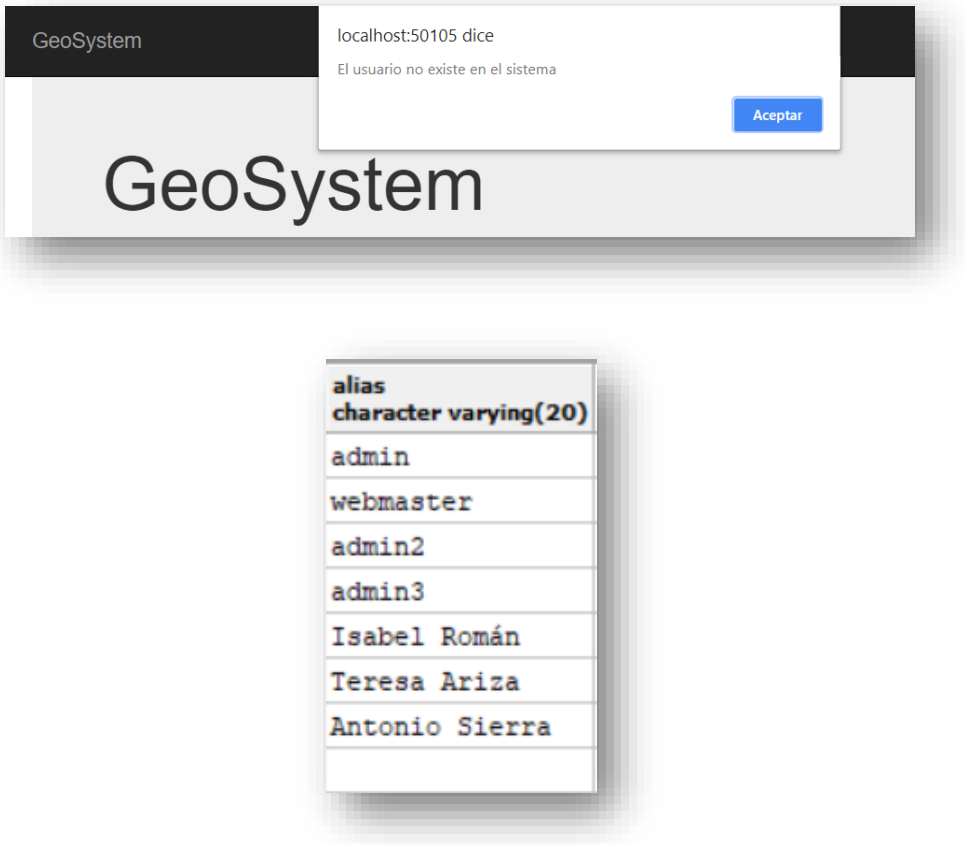
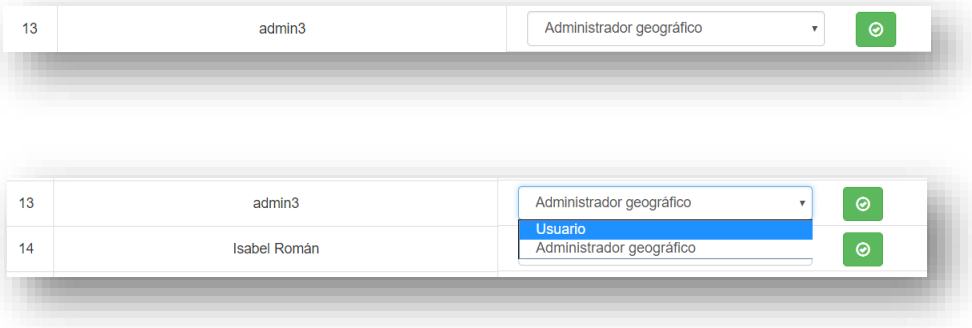
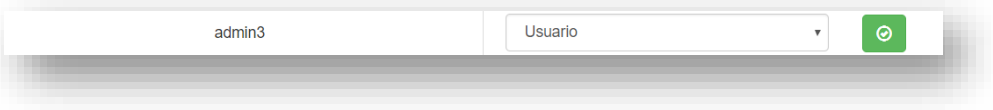
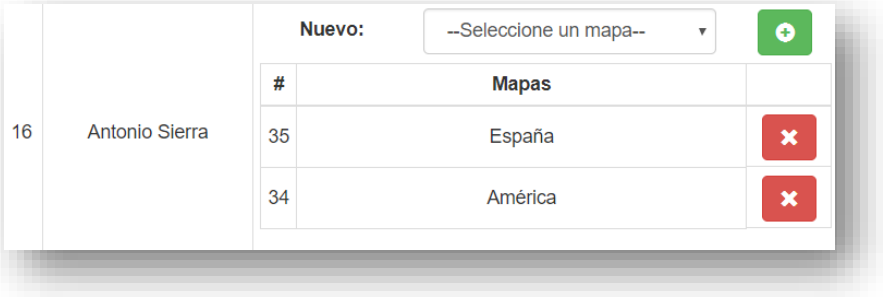
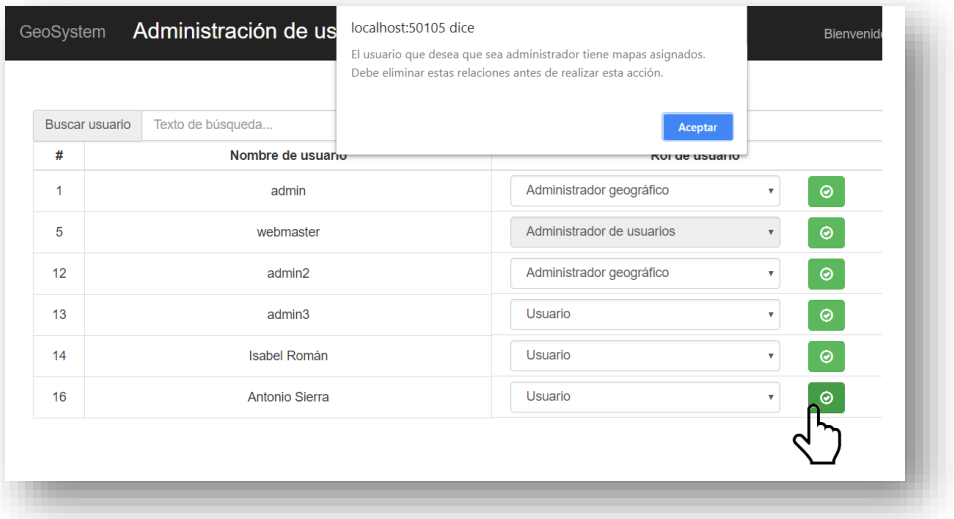
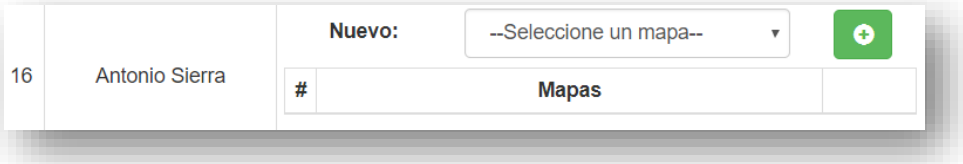


Ilustración 34: Prueba de Autenticación errónea

7.2. Administración de usuarios

PR-ADU-001	Cambiar el rol de un usuario
Descripción	Seleccionar un rol distinto para un usuario, validar el cambio, y que éste se guarde en el sistema.
Comprobación	

	Tras refrescar la ventana:  *Ilustración 35: Prueba de Cambiar el rol de un usuario*

PR-ADU-002	Cambiar el rol de un usuario con mapas asignados
Descripción	Seleccionar un usuario con rol Usuario, tratar de cambiarlo a Administrador y obtener un mensaje de error de que no puede cambiarse su rol por tener mapas asignados. Comprobar que, si deja de tener mapas asignados, ya sí puede ser modificado su rol.
Comprobación	  

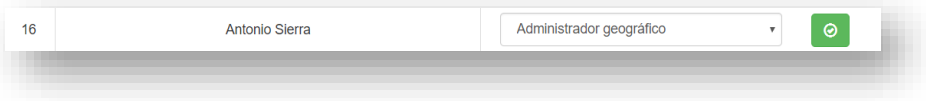
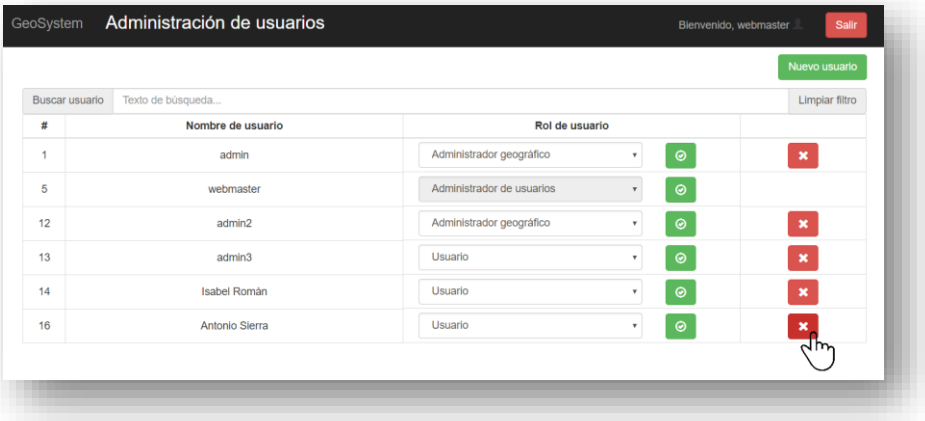


Ilustración 36: Prueba de Cambiar el rol de un usuario con mapas asignados

PR-ADU-003	Eliminar un usuario
Descripción	Eliminar un usuario y, en caso de que sea Usuario, comprobar que se borran sus relaciones en cascada
Comprobación	

	id_usuario integer	nombre_usuario character varying(20)	id_mapa integer	nombre_mapa character varying(20)
1	14	Isabel Román	33	Europa
2	14	Isabel Román	34	América
3	16	Antonio Sierra	33	Europa
4	16	Antonio Sierra	34	América



	id_usuario integer	nombre_usuario character varying(20)	id_mapa integer	nombre_mapa character varying(20)
1	14	Isabel Román	33	Europa
2	14	Isabel Román	34	América

Ilustración 37: Prueba de Eliminar un usuario

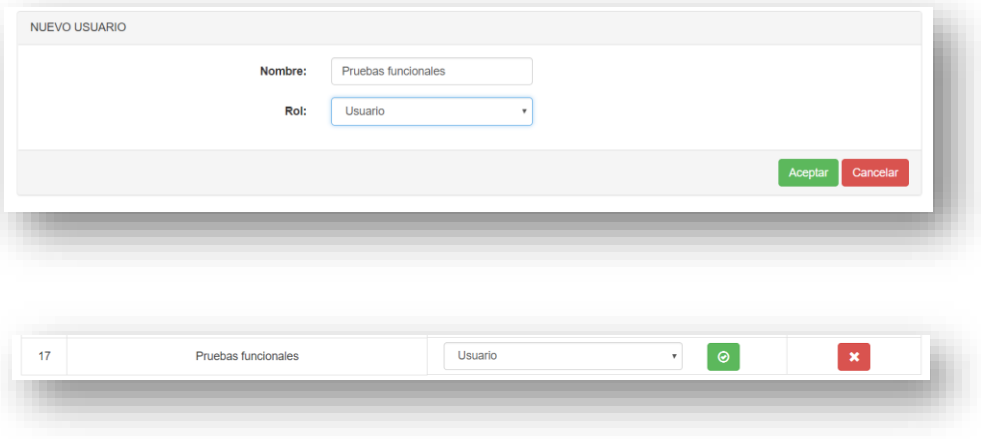
PR-ADU-004	Crear un usuario exitosamente
Descripción	Crear un usuario con todos los parámetros correctos
Comprobación	

Ilustración 38: Prueba de Crear un usuario exitosamente

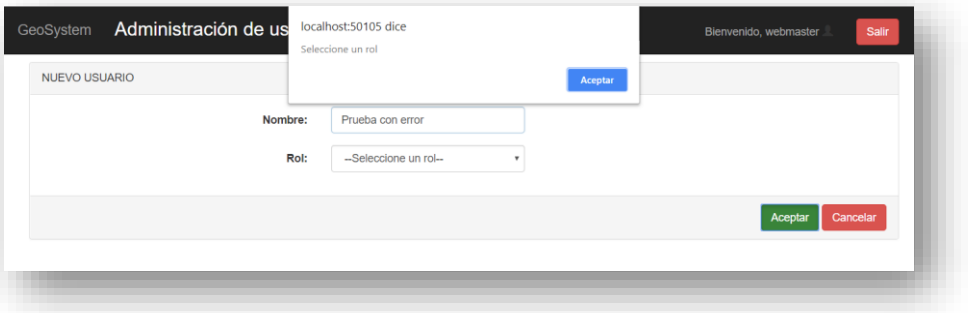
PR-ADU-005	Crear un usuario con campos sin rellenar
Descripción	Tratar de crear un usuario sin seleccionar su rol y comprobar que aparece un mensaje de error y no se permite crearlo
Comprobación	

Ilustración 39: Prueba de Crear un usuario con campos sin rellenar

7.3. Administración geográfica

7.3.1. Mapas

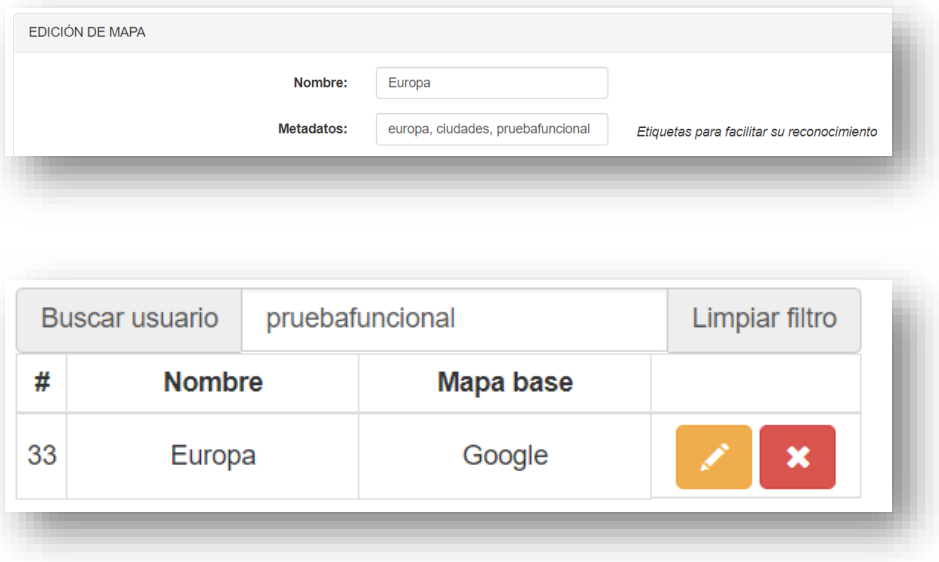
PR-ADG-001	Filtrado según metadatos del mapa
Descripción	Comprobar que el listado de mapas se filtra en función de los metadatos que se hayan introducido en éste, aunque este campo no aparezca en la tabla
Comprobación	

Ilustración 40: Prueba de Filtrado según metadatos del mapa

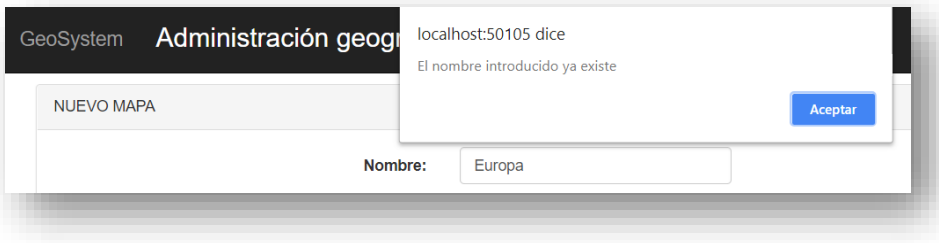
PR-ADG-002	Nombres únicos de mapas
Descripción	Tratar de crear un mapa con el mismo nombre de uno ya existente y comprobar que se obtiene un mensaje de error
Comprobación	

Ilustración 41: Prueba de Nombres únicos de mapas

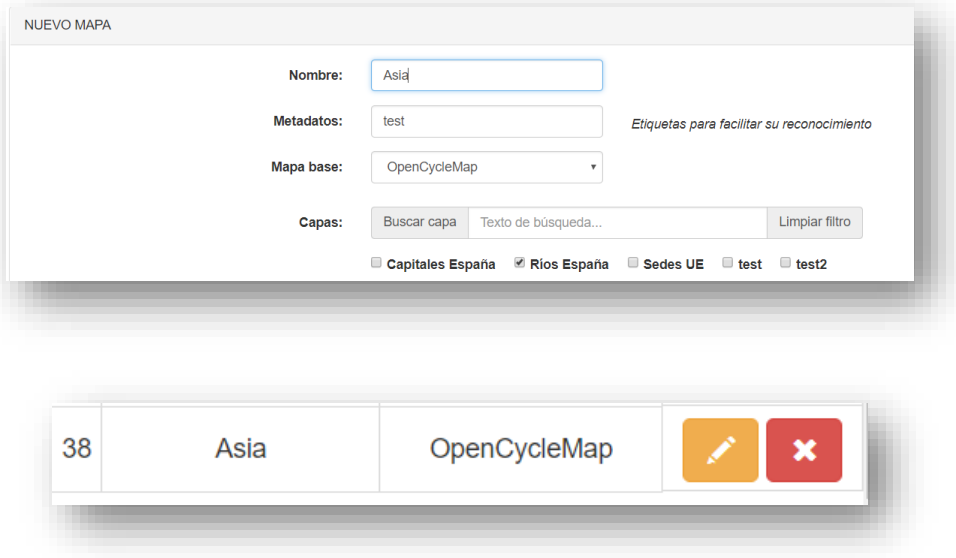
PR-ADG-003	Creación de mapa exitosa
Descripción	Comprobar que se crea un mapa cuando todos los campos introducidos son correctos
Comprobación	

Ilustración 42: Prueba de creación de mapa exitosa

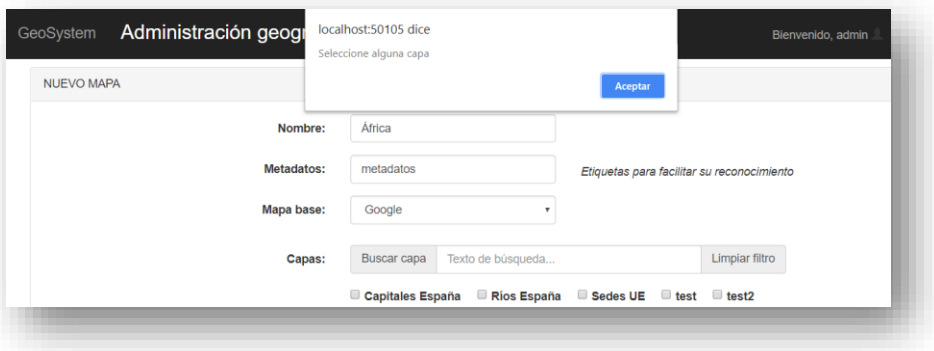
PR-ADG-004	Creación de mapa con campos sin rellenar
Descripción	Al tratar de crear un mapa sin completar todos los campos necesarios, comprobar que aparece un mensaje de error que no permite guardarlo
Comprobación	

Ilustración 43: Prueba de Creación de mapa con campos sin rellenar

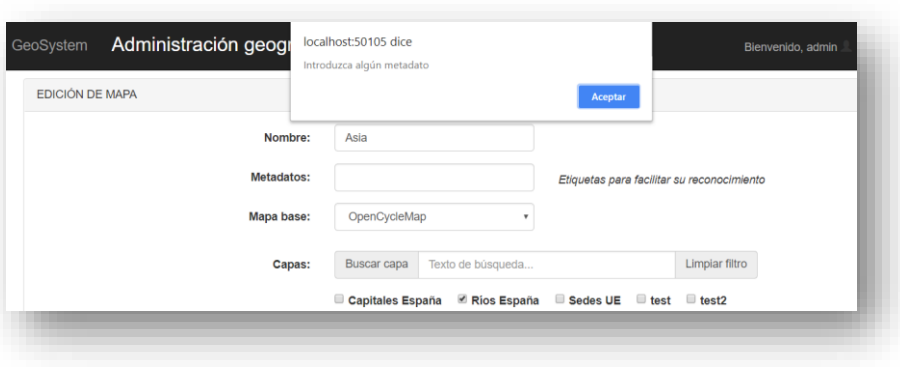
PR-ADG-005	Edición de mapa con campos sin rellenar
Descripción	Al tratar de editar un mapa sin completar todos los campos necesarios, comprobar que aparece un mensaje de error que no permite guardarlo
Comprobación	

Ilustración 44: Prueba de Edición de mapa con campos sin rellenar

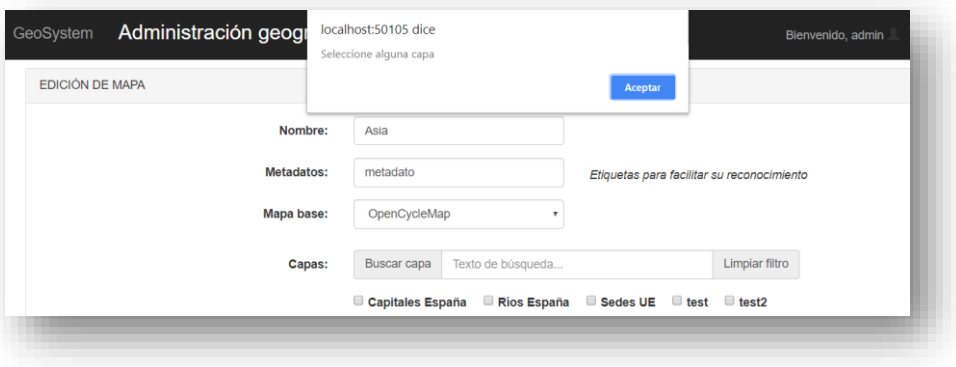
PR-ADG-006	Edición de mapa sin capas seleccionadas
Descripción	Al tratar de editar un mapa sin seleccionar ninguna capa, comprobar que aparece un mensaje de error que no permite guardarlo
Comprobación	

Ilustración 45: Prueba de Edición de mapa sin capas seleccionadas

PR-ADG-007	Edición de mapa exitosa
Descripción	Comprobar que se edita un mapa cuando todos los campos introducidos son correctos
Comprobación	

Ilustración 46: Prueba de Edición de mapa exitosa

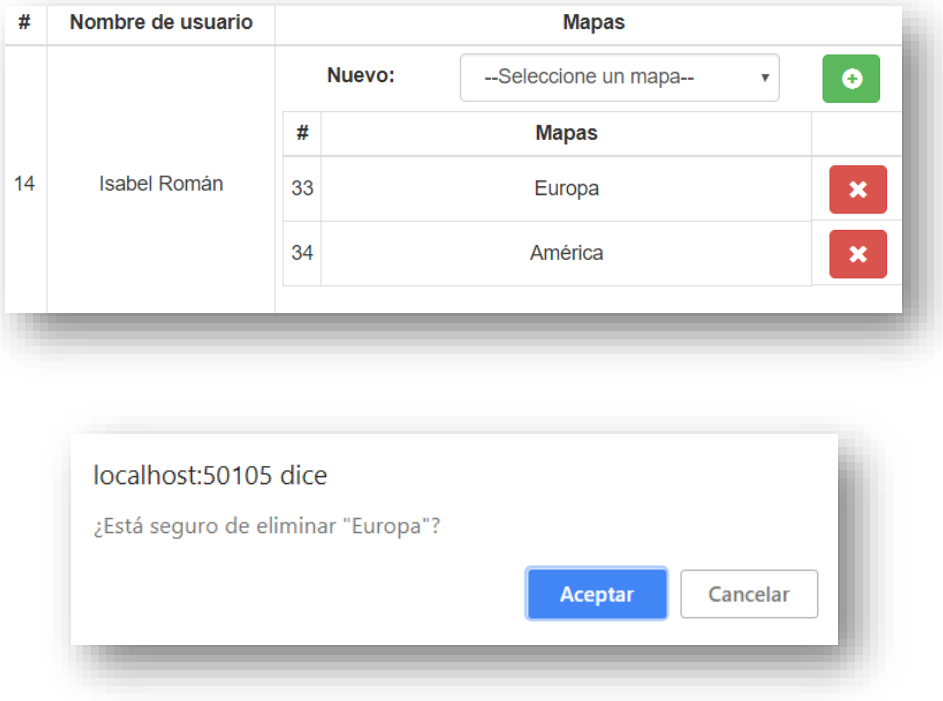
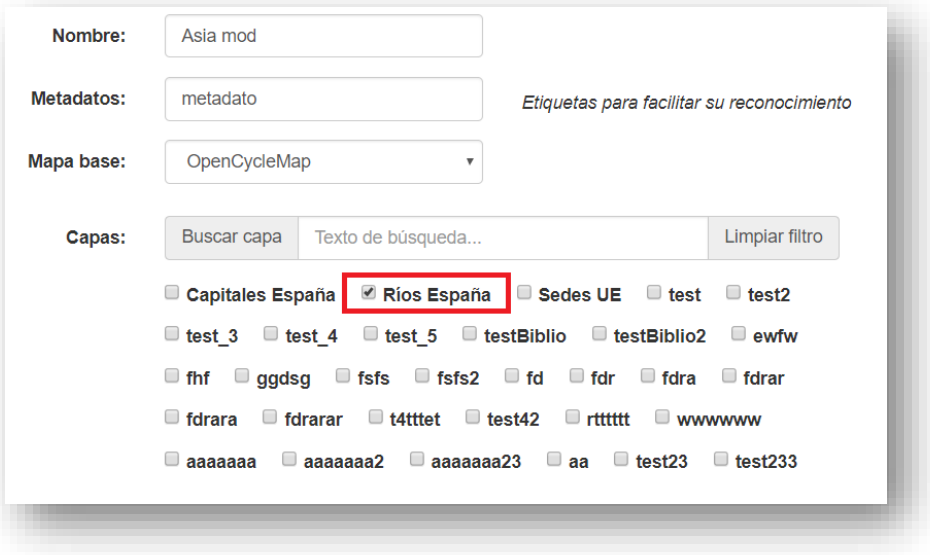
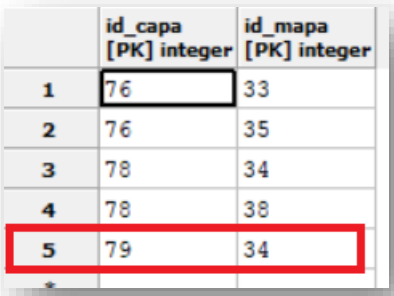
PR-ADG-008	Eliminar mapa con usuario vinculado
Descripción	Tratar de eliminar un mapa que tenga un usuario vinculado y comprobar que aparece un mensaje de error que lo impide
Comprobación	

Ilustración 47: Prueba de Eliminar mapa con usuario vinculado

PR-ADG-009	Eliminar mapa con capa vinculada
Descripción	Tratar de eliminar un mapa que tenga una capa vinculada y comprobar que se elimina tanto el mapa como la relación con la capa
Comprobación	



	id_capa [PK] integer	id_mapa [PK] integer
1	76	33
2	76	35
3	78	34
4	78	38
5	79	34

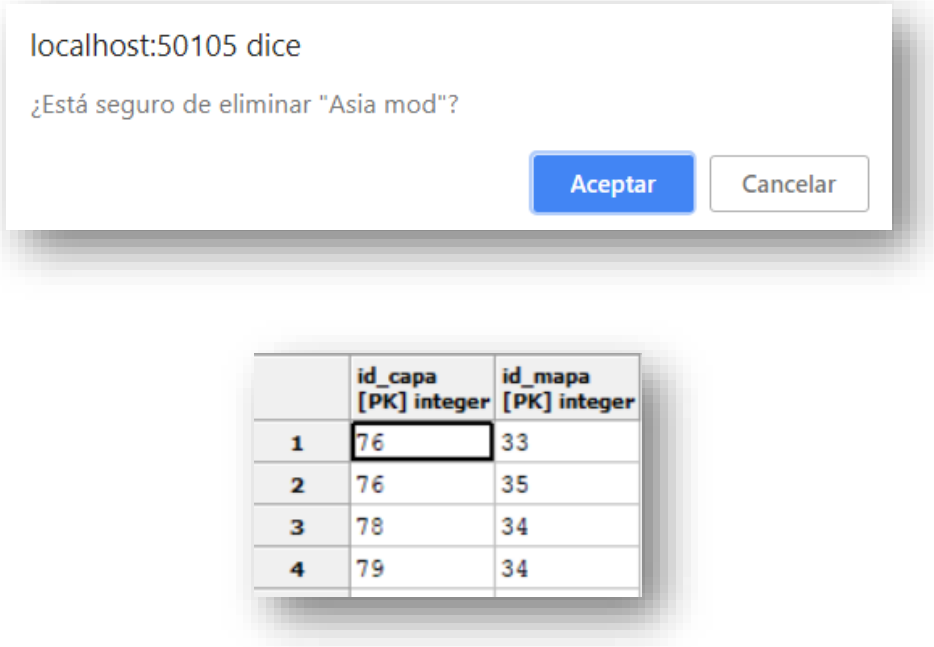


Ilustración 48: Prueba de Eliminar mapa con capa vinculada

7.3.2. Capas

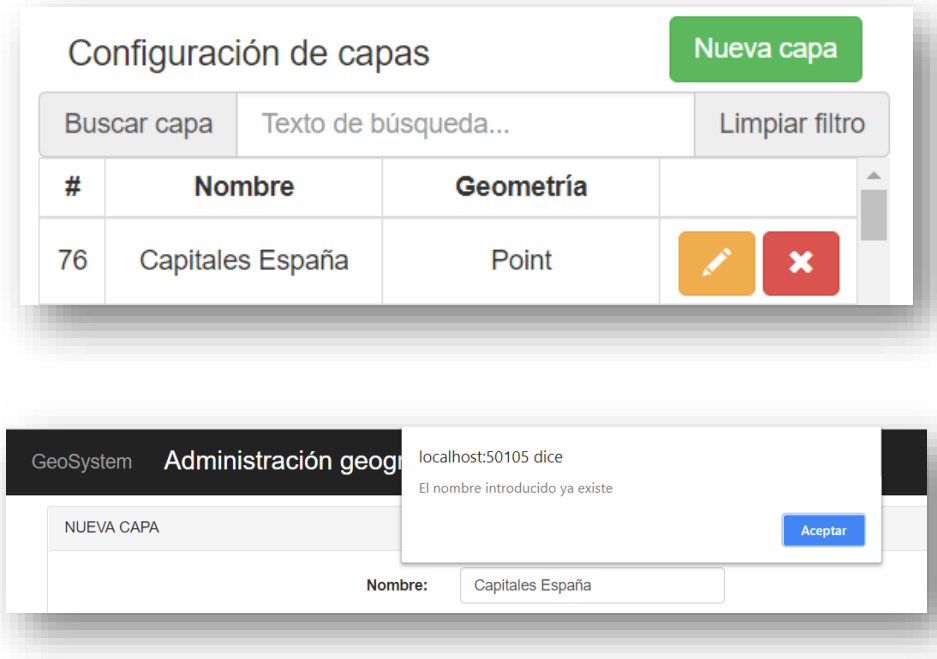
PR-ADG-010	Nombres únicos de capas
Descripción	Tratar de crear una capa con el mismo nombre de una ya existente y comprobar que se obtiene un mensaje de error
Comprobación	

Ilustración 49: Prueba de Nombres únicos de capas

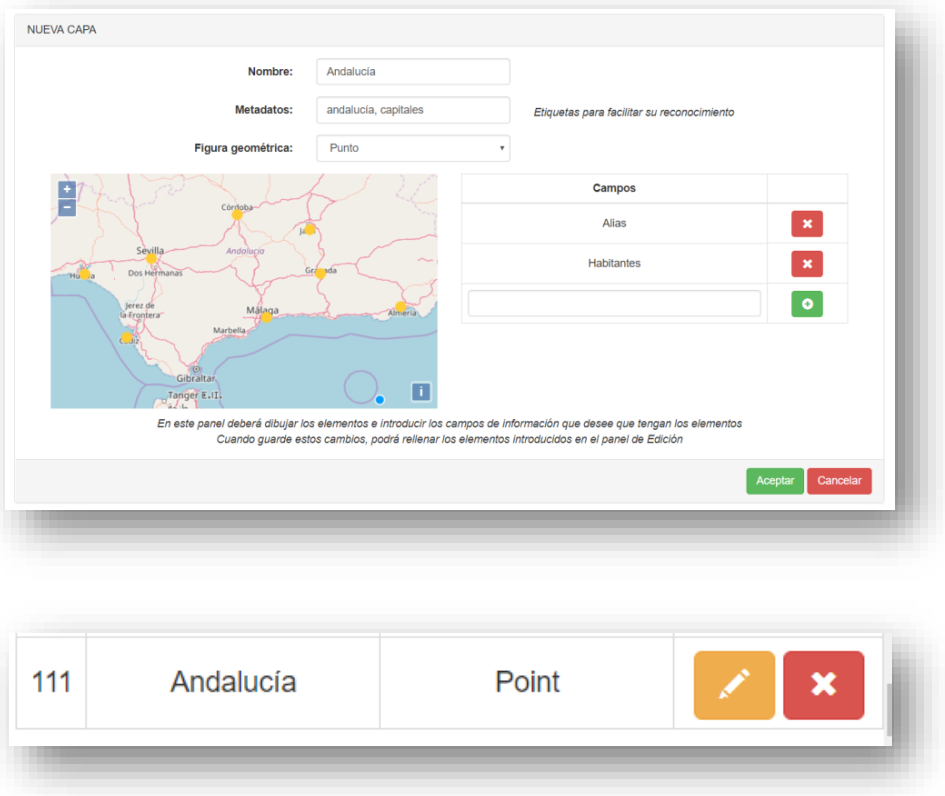
PR-ADG-011	Creación de capa exitosa
Descripción	Comprobar que se crea una capa cuando todos los campos introducidos son correctos
Comprobación	

Ilustración 50: Prueba de Creación de capa exitosa

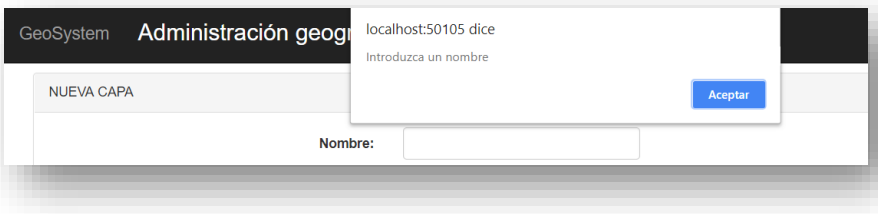
PR-ADG-012	Creación de capa con campos sin rellenar
Descripción	Al tratar de crear una capa sin completar todos los campos necesarios, comprobar que aparece un mensaje de error que no permite guardarlo
Comprobación	

Ilustración 51: Prueba de Creación de capa con campos sin rellenar

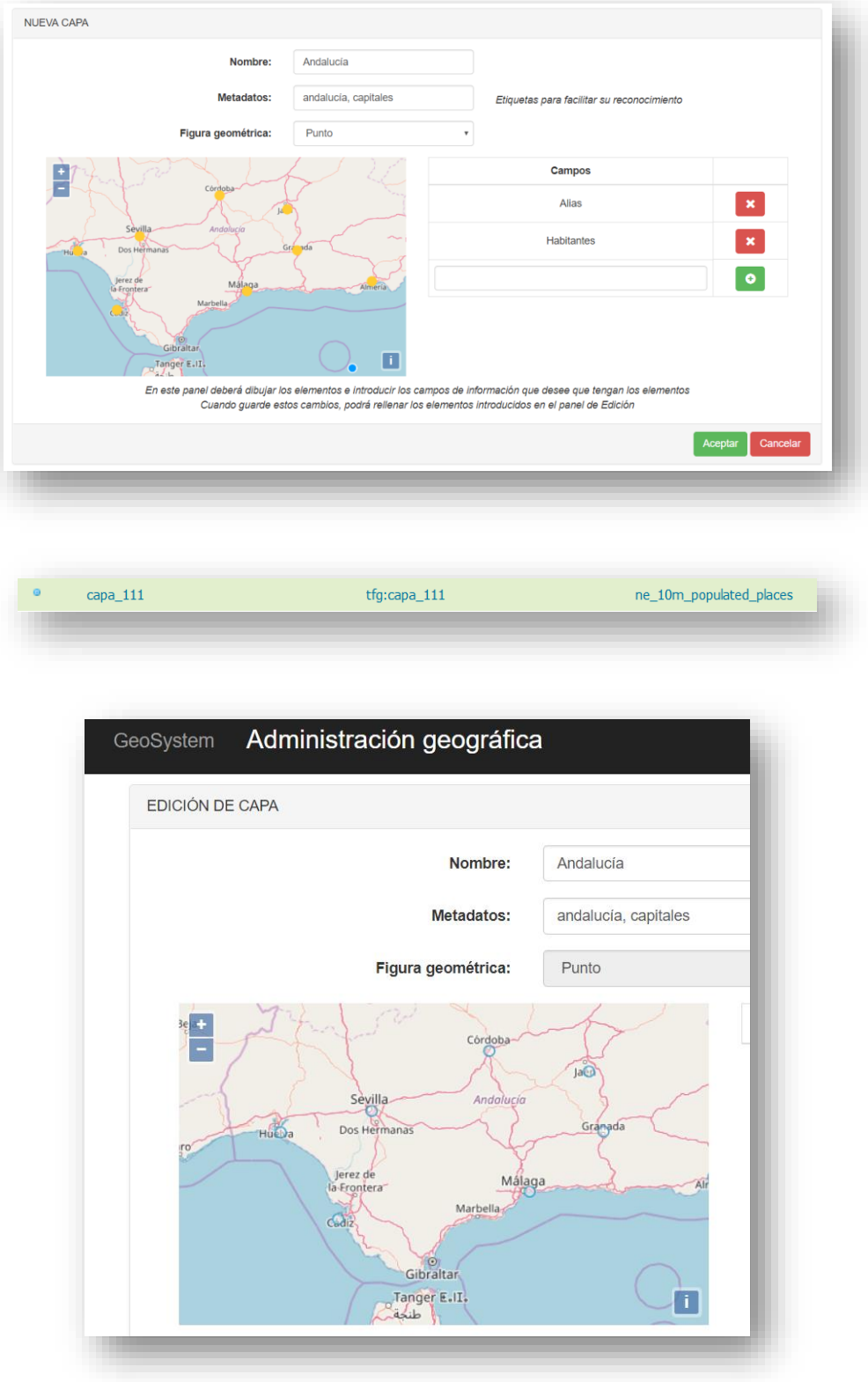
PR-ADG-013	Comprobación de la creación de los elementos de la nueva capa
Descripción	Al crear una capa exitosamente, comprobar que sus elementos geográficos se han guardado Nota: Es necesario el paso manual de subirla en GeoServer
Comprobación	

Ilustración 52: Prueba de Comprobación de la creación de los elementos de la nueva capa

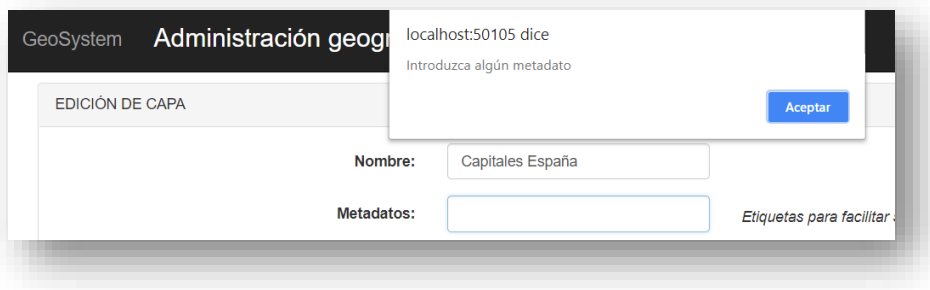
PR-ADG-014	Edición de capa con campos sin rellenar
Descripción	Al tratar de editar una capa sin completar todos los campos necesarios, comprobar que aparece un mensaje de error que no permite guardarla
Comprobación	

Ilustración 53: Prueba de Edición de capa con campos sin rellenar

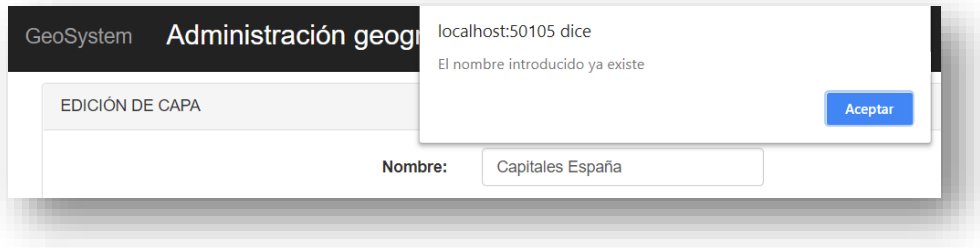
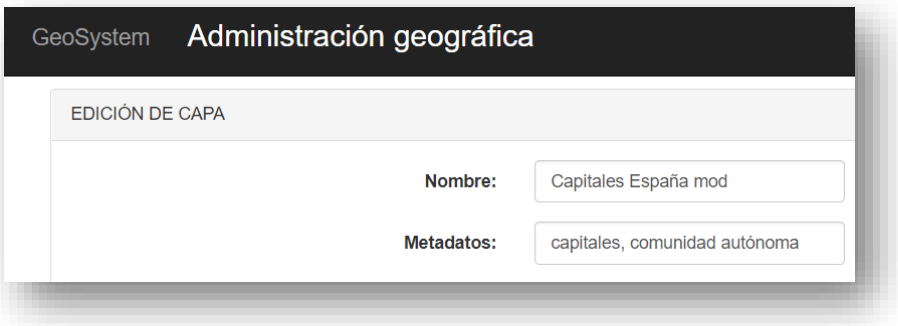
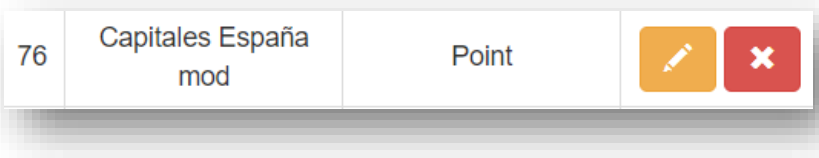
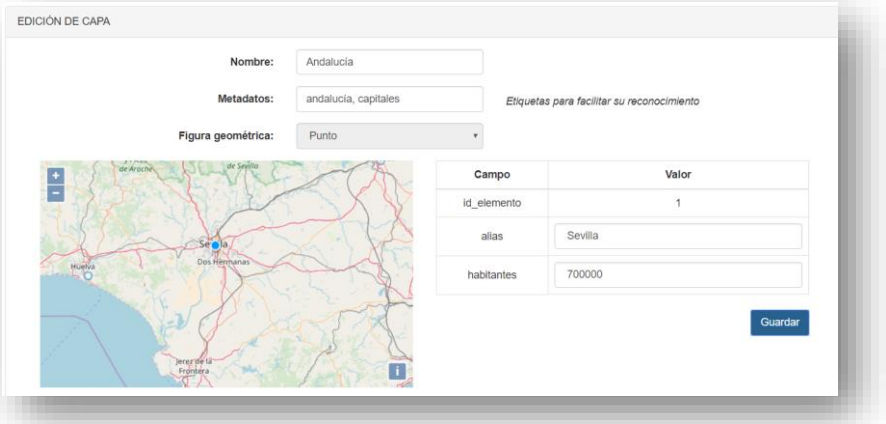
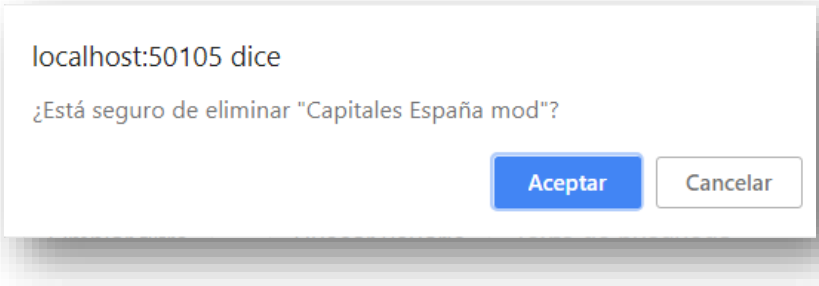
PR-ADG-015	Edición de capa con nombre repetido
Descripción	Al tratar de editar una capa e introducir un nombre ya existente, comprobar que aparece un mensaje de error que no permite guardarla
Comprobación	

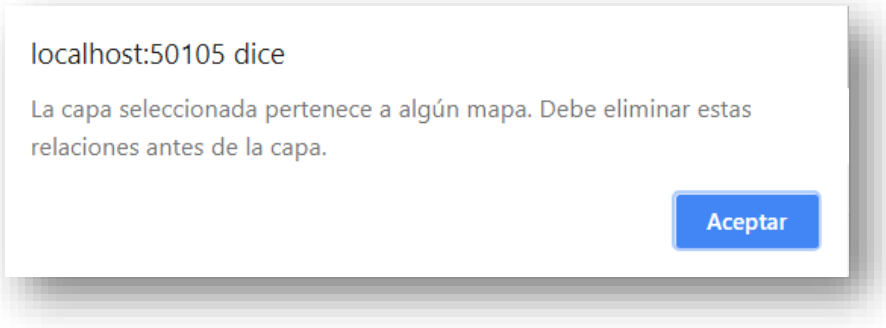
Ilustración 54: Prueba de Edición de capa con nombre repetido

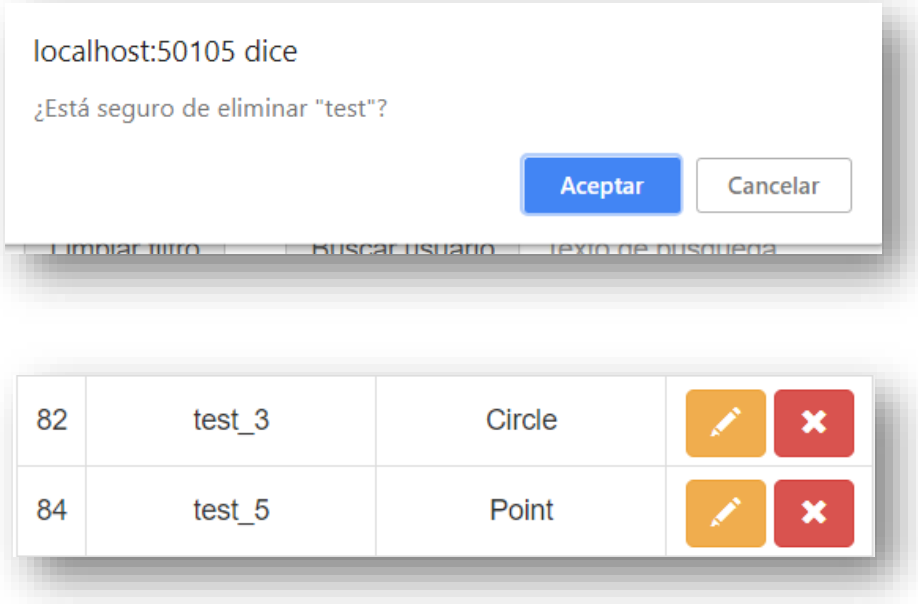
PR-ADG-016	Edición de capa exitosa
Descripción	Comprobar que se edita una capa cuando todos los campos introducidos son correctos
Comprobación	

	 *Ilustración 55: Prueba de Edición de capa exitosa*

PR-ADG-017	Edición de datos de un elemento
Descripción	Comprobar que los datos se guardan al selecionar y rellenar un elemento de la capa
Comprobación	 *Ilustración 56: Prueba de Edición de datos de un elemento*

PR-ADG-018	Eliminar una capa ligada a un mapa
Descripción	Comprobar que, al tratar de eliminar una capa ligada a un mapa, aparece un mensaje de error que impide el eliminado
Comprobación	

	 *Ilustración 57: Prueba de Eliminar una capa ligada a un mapa*

PR-ADG-019	Eliminar una capa sin relaciones
Descripción	Comprobar que se elimina una capa correctamente si ésta no tiene relación con ningún mapa.
Comprobación	 *Ilustración 58: Prueba de Eliminar una capa sin relaciones*

7.3.3. Usuarios-mapas

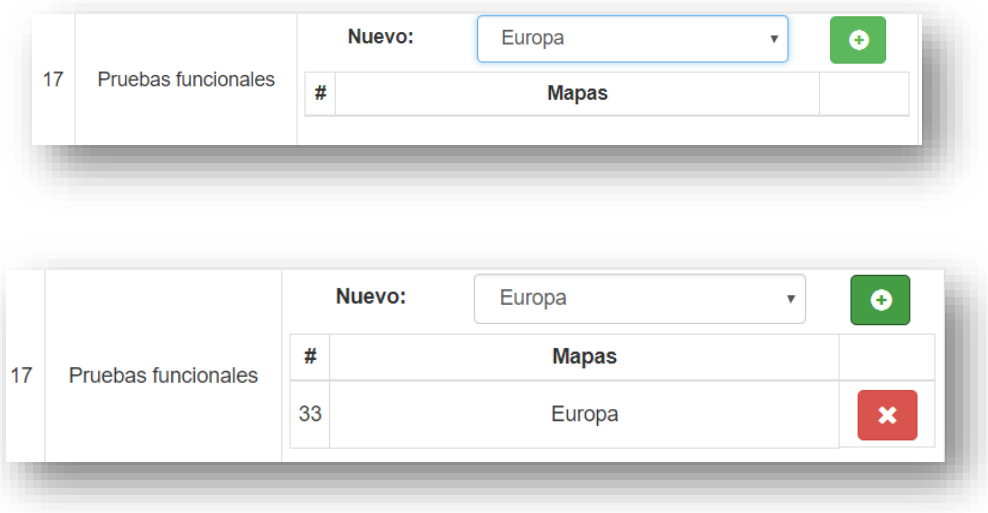
PR-ADG-020	Añadir un mapa a un usuario
Descripción	Comprobar la adición de la relación de un mapa con un usuario
Comprobación	

Ilustración 59: Prueba de Añadir un mapa a un usuario

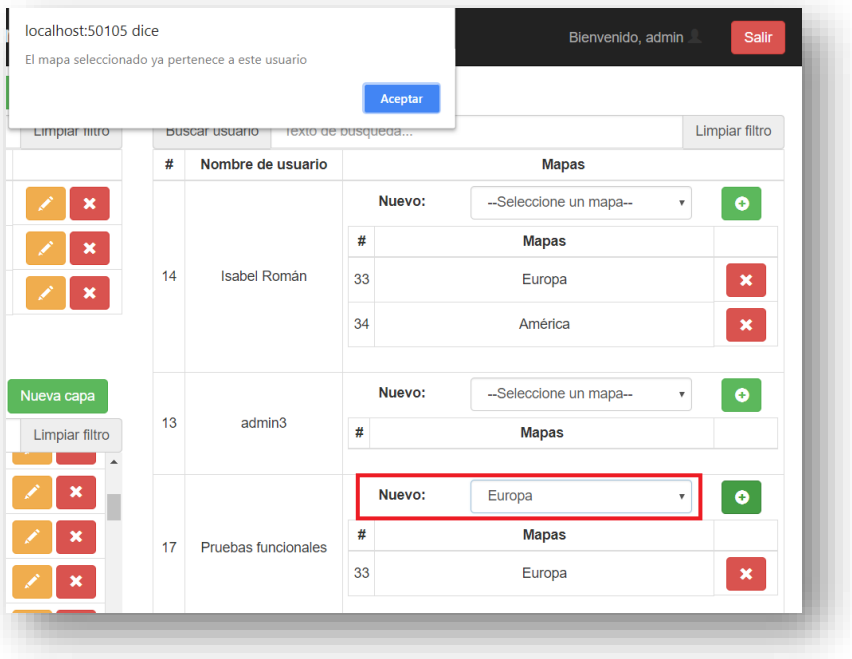
PR-ADG-021	Añadir un mapa a un usuario que ya lo tiene asignado
Descripción	Comprobar que aparece un mensaje de error cuando se trata de asignar a un usuario un mapa que ya tiene
Comprobación	

Ilustración 60: Prueba de Añadir un mapa a un usuario que ya lo tiene asignado

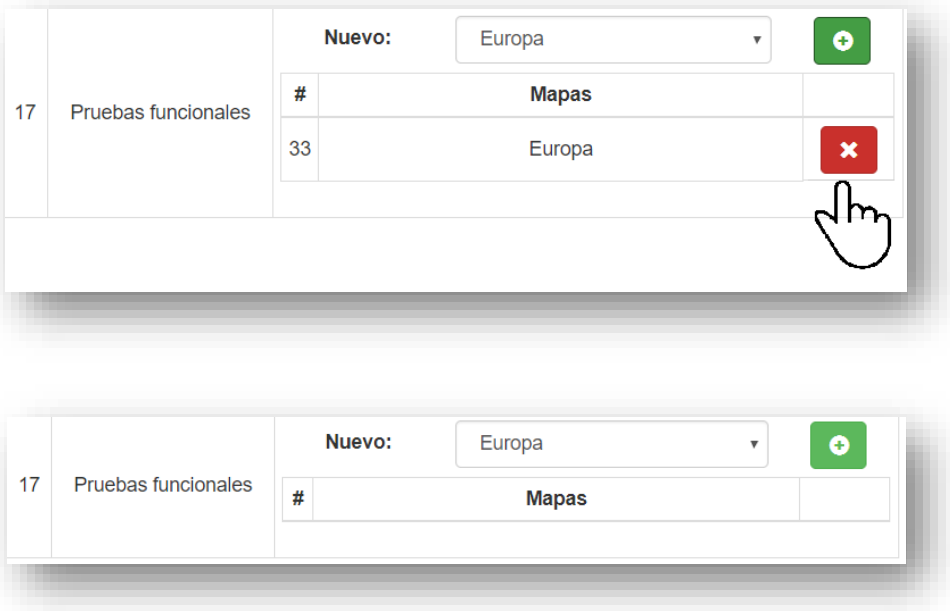
PR-ADG-022	Eliminar un mapa de un usuario
Descripción	Comprobar que se efectúa el eliminado de un mapa para un usuario
Comprobación	

Ilustración 61: Prueba de Eliminar un mapa de un usuario

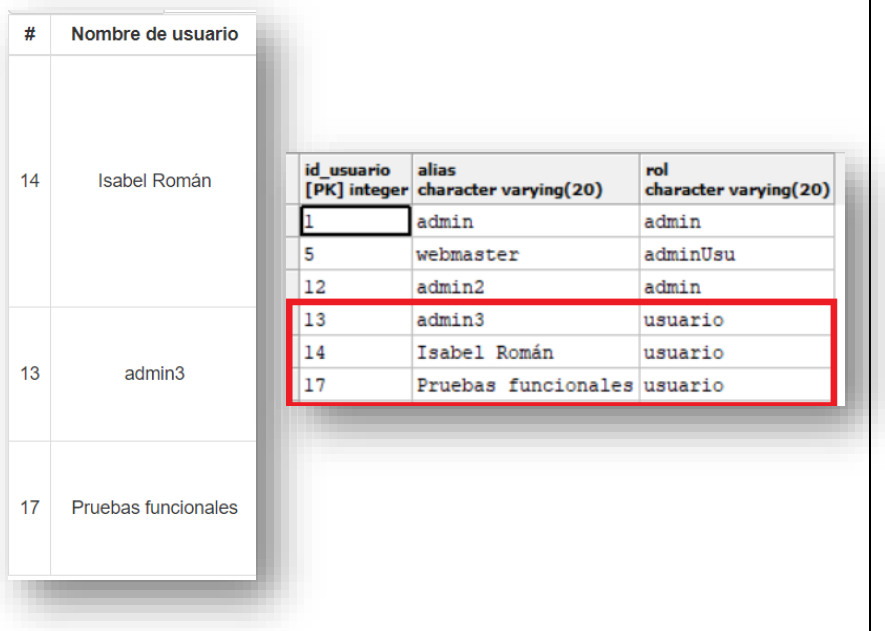
PR-ADG-023	Listado de usuarios
Descripción	Comprobar que solo aparecen usuarios del tipo Usuario, y no Administradores
Comprobación	

Ilustración 62: Prueba de Listado de usuarios

7.4. Visor

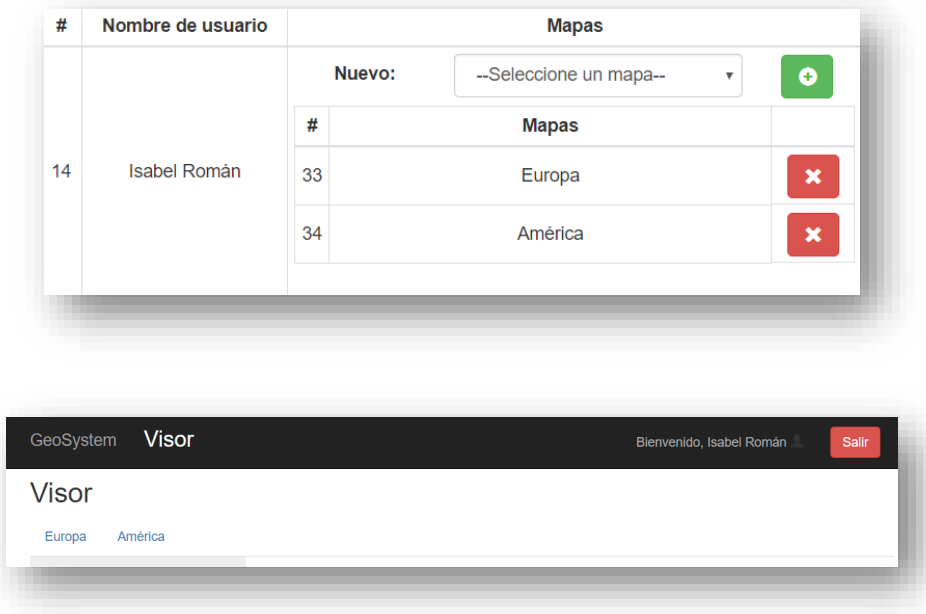
PR-VIS-001	Listado de mapas
Descripción	Comprobar que el listado de mapas que se despliega para un usuario es el que le corresponde
Comprobación	

Ilustración 63: Prueba de Listado de mapas

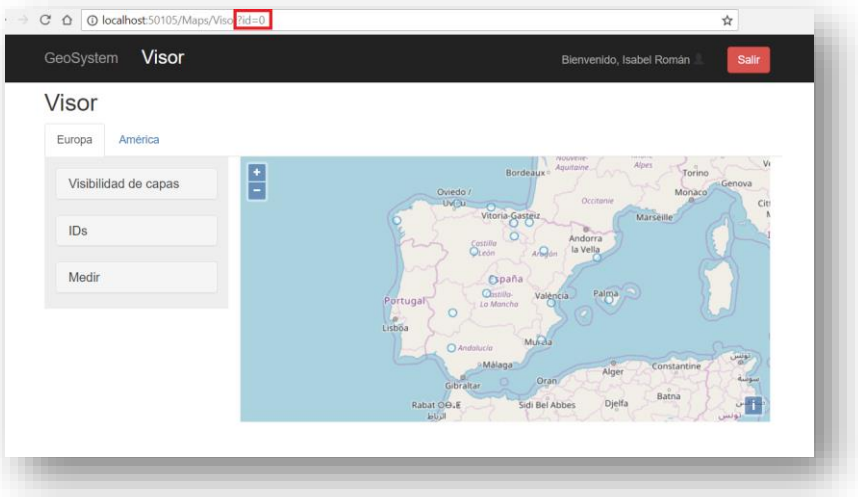
PR-VIS-002	Apertura del mapa correspondiente
Descripción	Seleccionar un mapa del menú y comprobar que se despliega
Comprobación	

Ilustración 64: Prueba de Apertura del mapa correspondiente

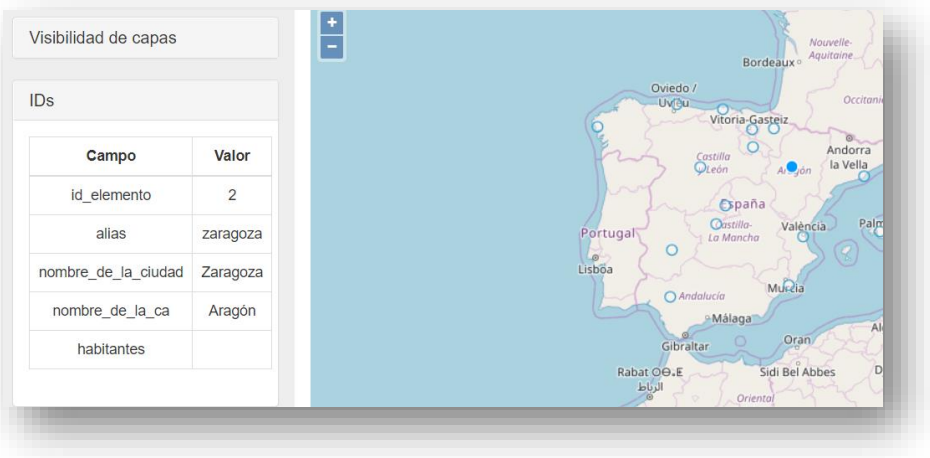
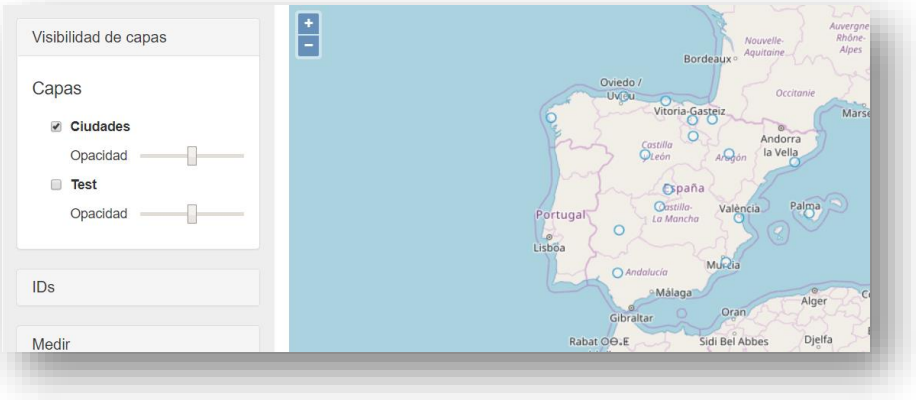
PR-VIS-003	Identificación del mapa
Descripción	Seleccionar un elemento del mapa y comprobar que se cargan sus datos en la herramienta de identificación
Comprobación	

Ilustración 65: Prueba de Identificación del mapa

PR-VIS-004	Visibilidad de capas
Descripción	Deseleccionar una capa y comprobar que deja de mostrarse en el mapa
Comprobación	

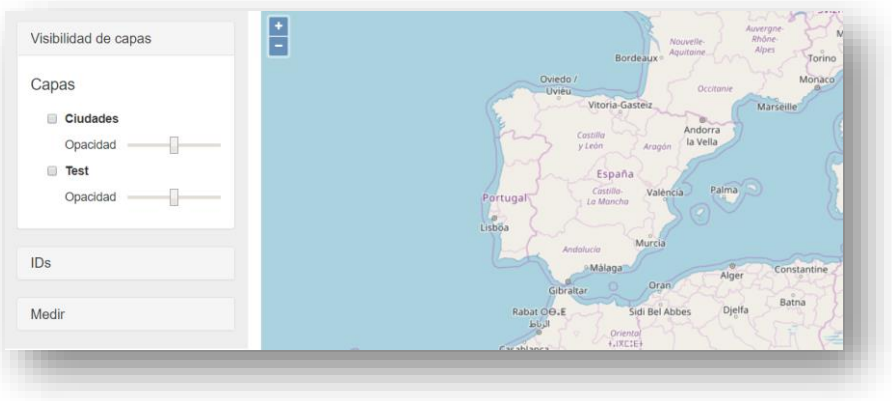
	

Ilustración 66: Prueba de visibilidad de capas

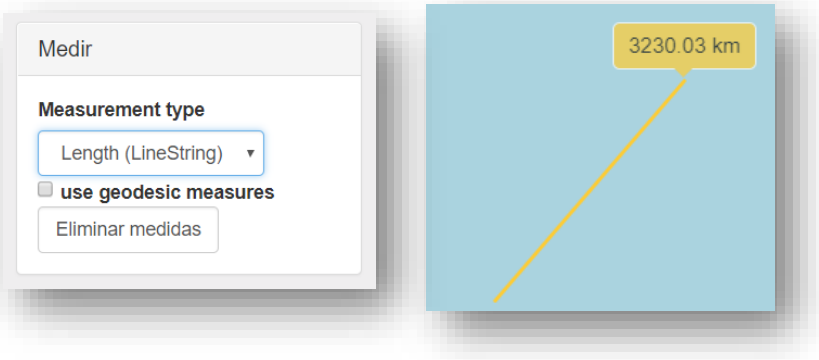
PR-VIS-005	Medición de distancias
Descripción	Comprobar la medición de distancias en el mapa
Comprobación	

Ilustración 67: Prueba de Medición de distancias

8 DESPLIEGUE

"¡Eureka!"

- Arquímedes -

Es necesario seguir una serie de procedimientos para poder migrar, portar y desplegar el sistema creado a otros entornos. Este despliegue requiere no solamente la aplicación principal, si no todos aquellos sistemas externos que utiliza.

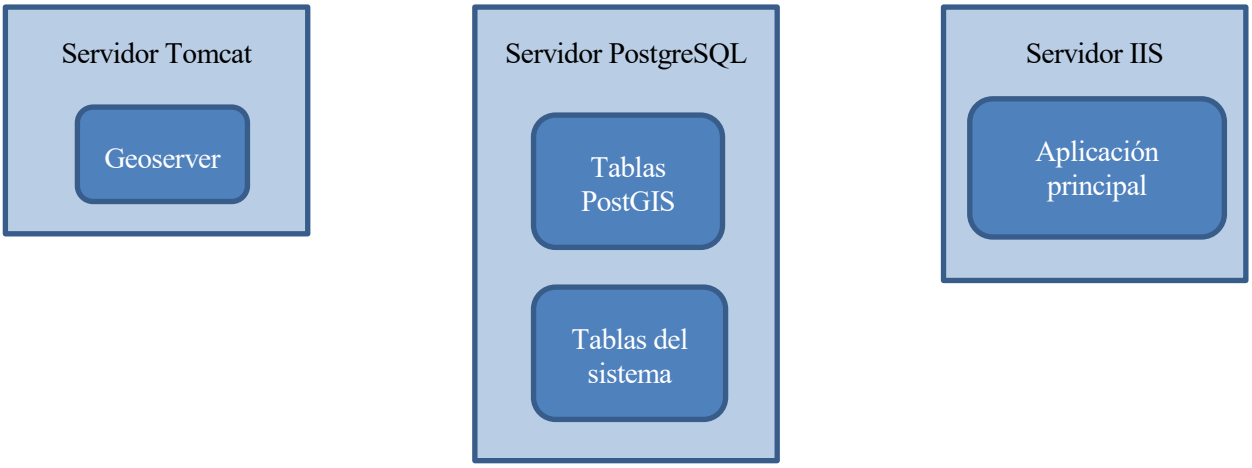


Ilustración 68: Despliegue

8.1. Proyecto

El principal producto del proyecto ha sido desarrollado en la plataforma Visual Studio. Ésta facilita el uso del servidor web Microsoft IIS de Microsoft para su publicación.

8.1.1. Publicación

El menú de Visual Studio, se selecciona la opción 'Publicar #Nombre del proyecto#':

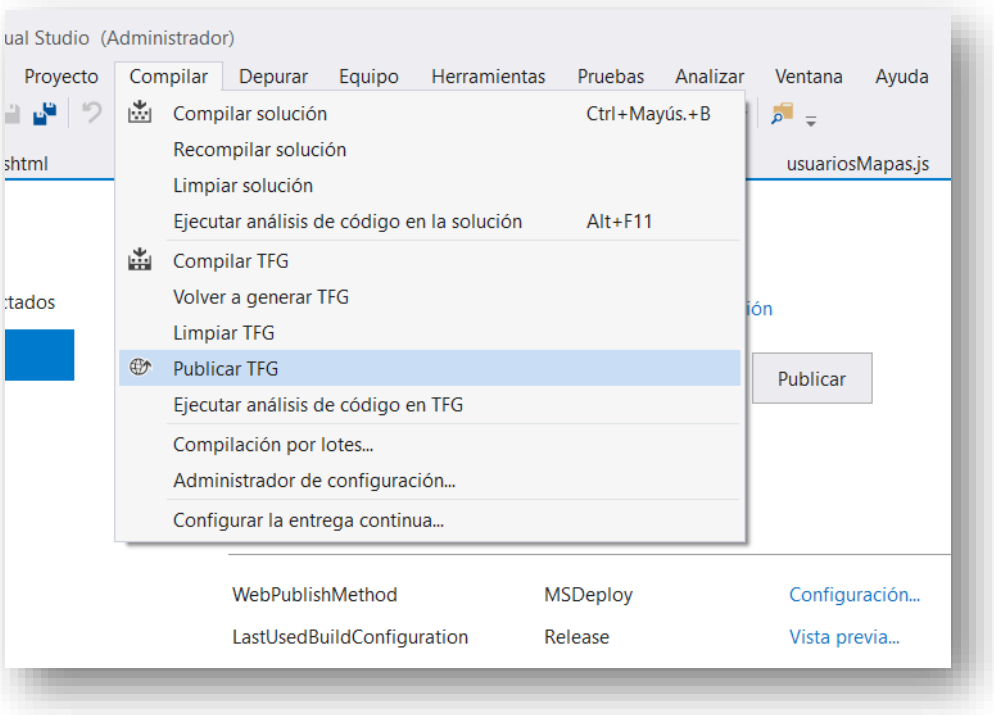


Ilustración 69: Publicar proyecto en Visual Studio

Se selecciona crear un nuevo perfil de carpeta:

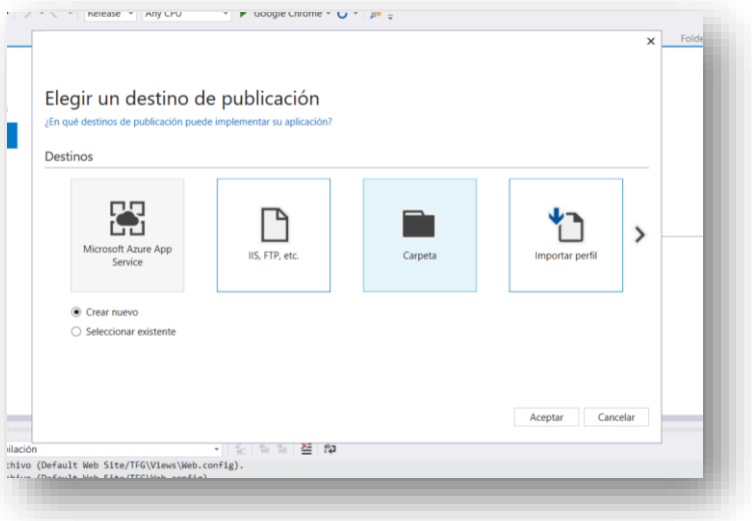


Ilustración 70: Perfil de carpeta para publicar proyecto

El proyecto se publicará en la ruta que se indique.

8.1.2. Despliegue

En cuando al servidor ISS, éste puede instalarse como Característica de Windows:

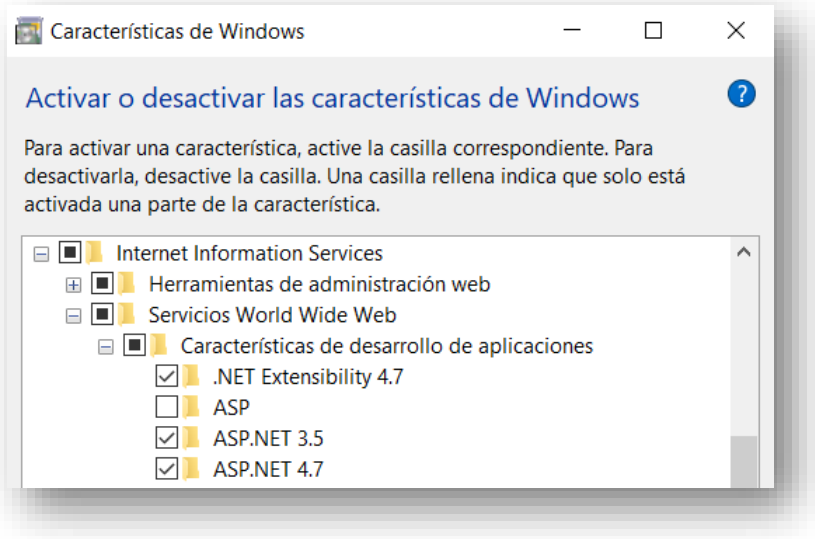


Ilustración 71: Instalación de servidor ISS

Pueden consultarse los grupos de aplicaciones de IIS desde su administrador, accesible con el comando Windows 'inetmgr':

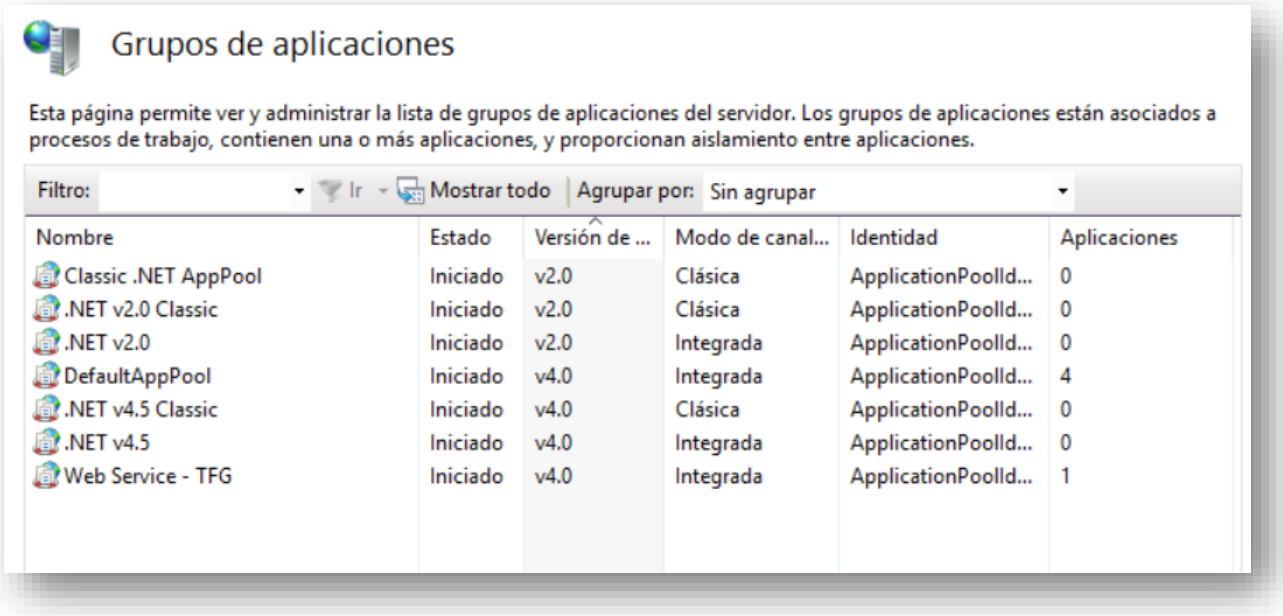


Ilustración 72: Grupos de aplicaciones IIS

Éste se sitúa en la ruta C:\inetpub\wwwroot. Basta con situar la carpeta extraída de Visual Studio en este directorio para que el proyecto quede desplegado:

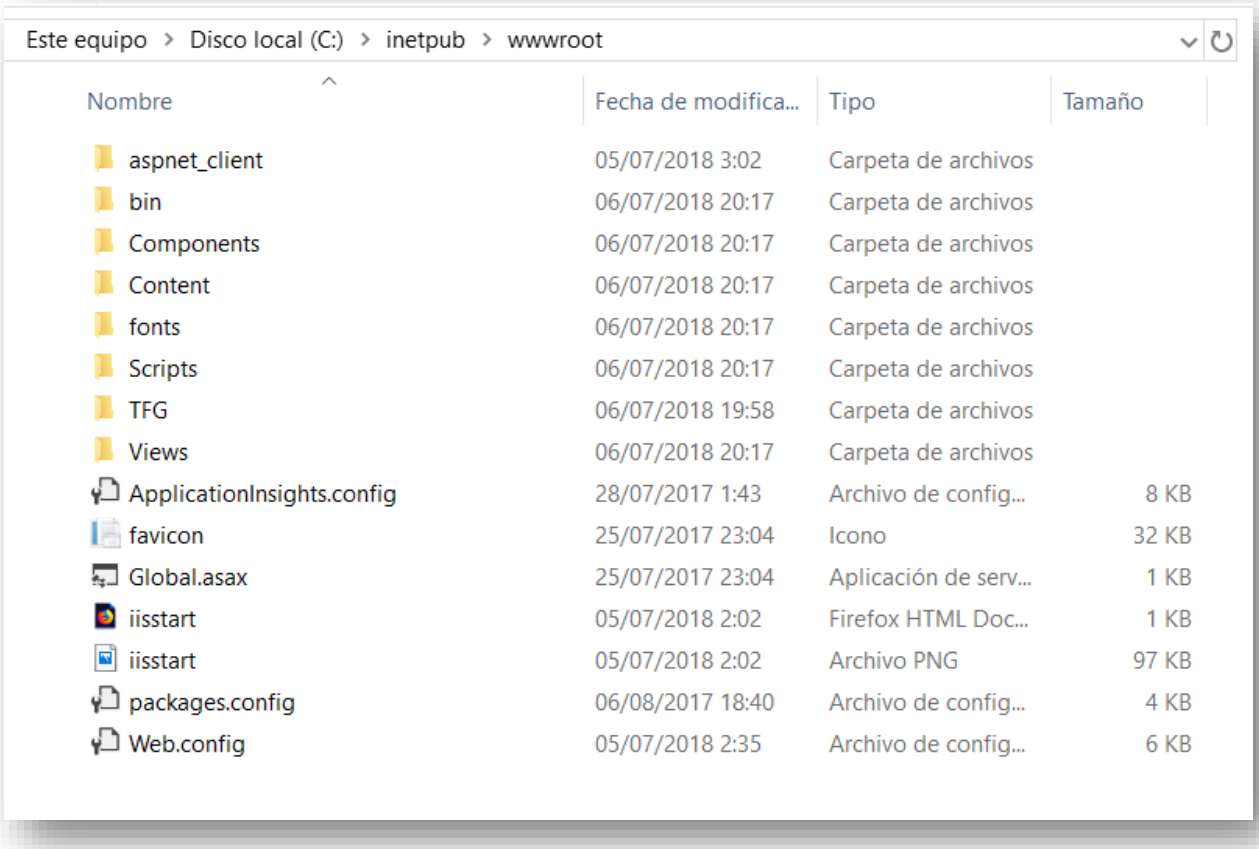


Ilustración 73: Directorio para desplegar el proyecto en el servidor

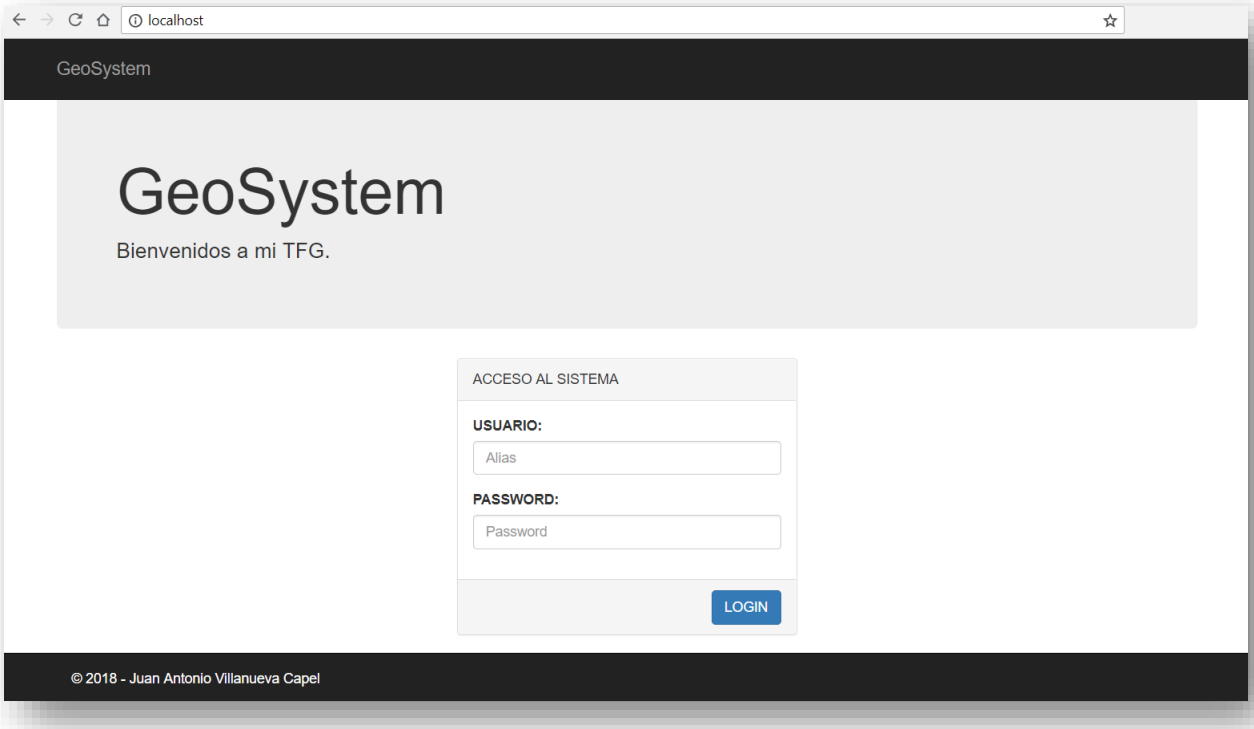


Ilustración 74: Proyecto desplegado

8.2. Tomcat y Geoserver

El servidor Tomcat es usado para desplegar Geoserver.

Al situar el archivo WAR que Geoserver proporciona en el directorio 'webapps' de Geoserver, éste queda desplegado y se genera la carpeta en la que se almacenan sus datos,

Para que Tomcat arranque correctamente, es necesario configurar las variables de entorno para tomar la versión adecuada de Java:

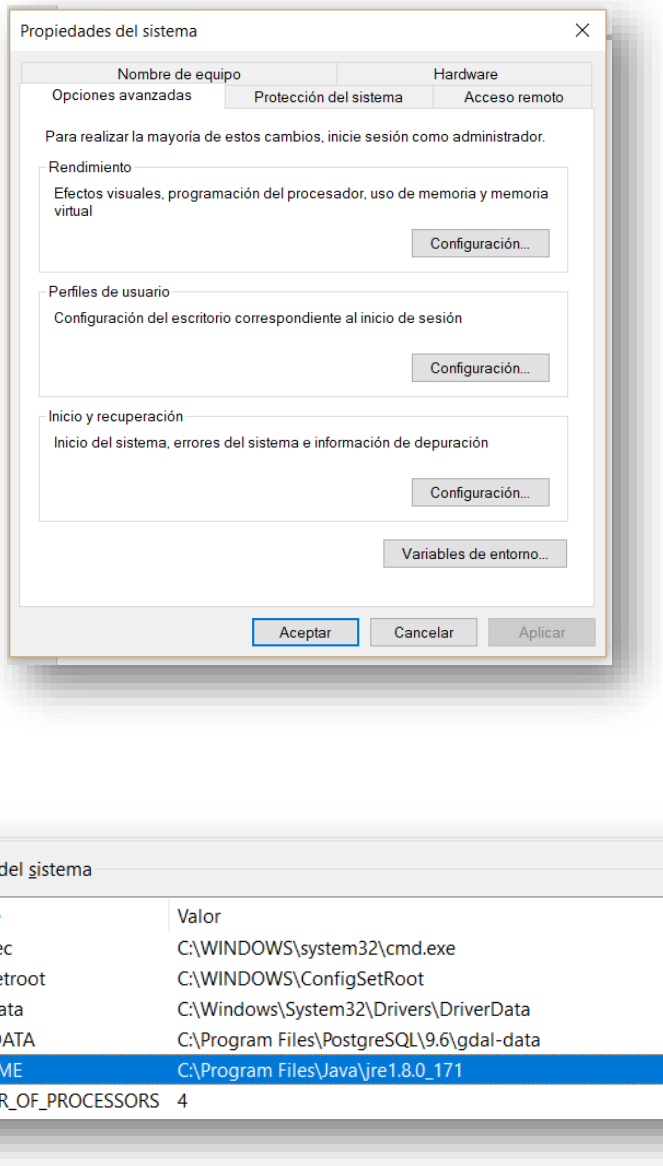


Ilustración 75: Variables de entorno Java para utilizar Tomcar

Así, Geoserver queda accesible en la ruta http://localhost:8080/geoserver/web/:

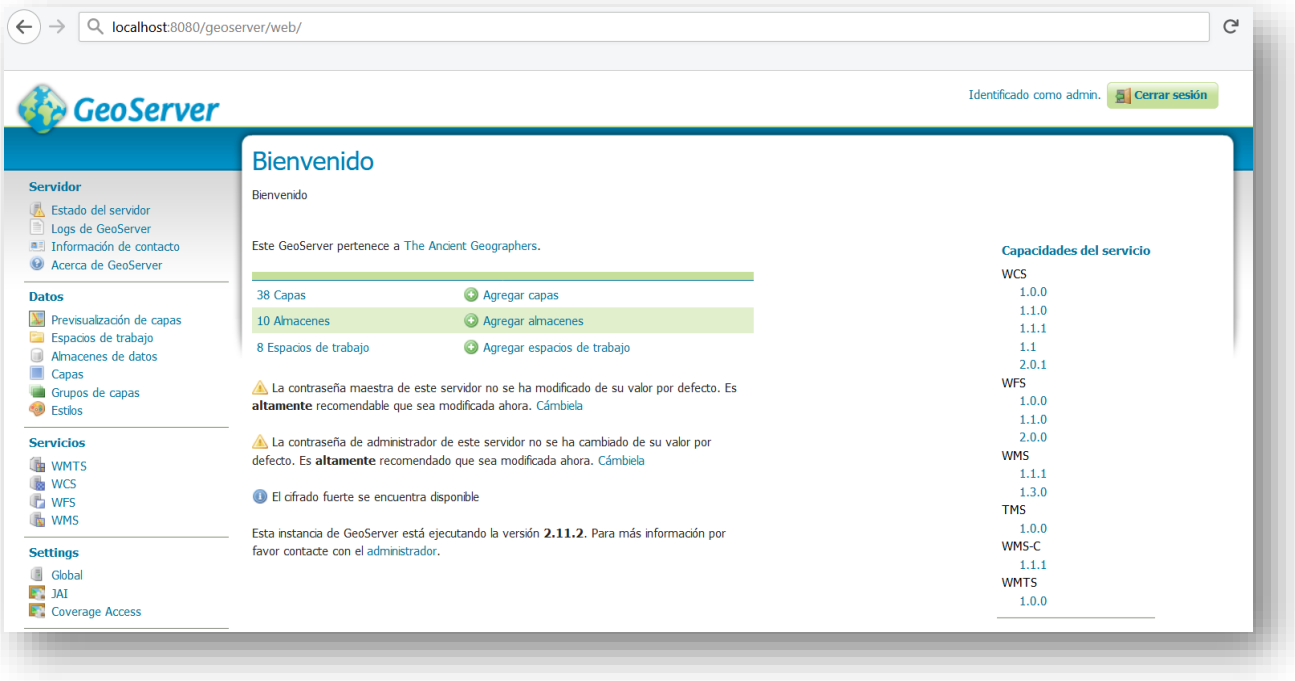


Ilustración 76: GeoServer desplegado

En este panel de administración es posible crear y gestionar las capas de las tablas existentes de la base de datos PostGIS.

8.3. PostgreSQL y PostGIS

El servidor PostgreSQL es fácilmente instalable mediante un ejecutable. Una vez instalado, Windows hace que quede corriendo.

Para gestionarlo de una forma más sencilla, es posible utilizar el software pgAdmin, en el que se introducen los parámetros del servidor.

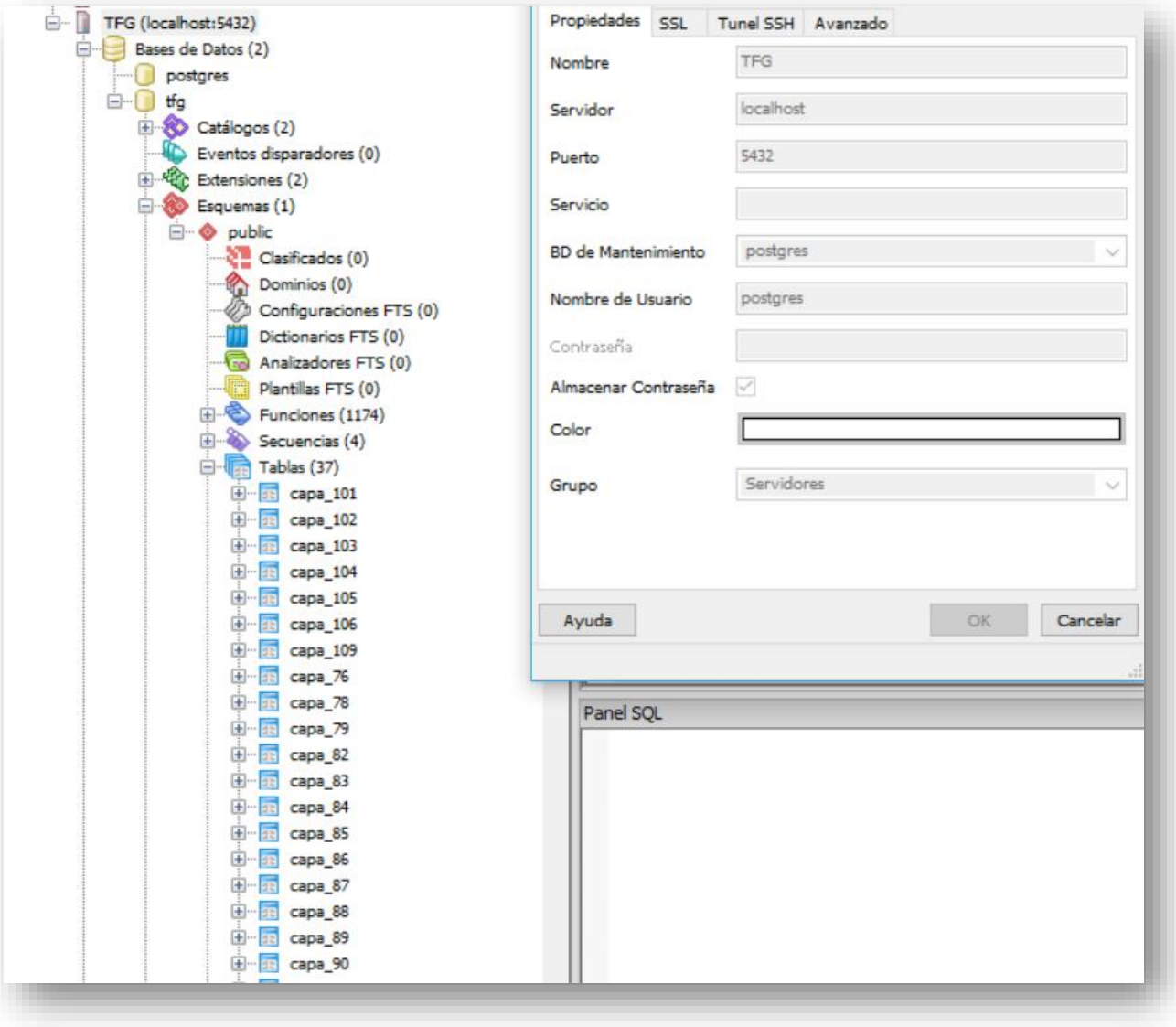


Ilustración 77: Base de datos 'TFG' en PGAdmin

PostgreSQL cuenta con las utilidades pg_dump y pg_restore [Despliegue_3] que, respectivamente, exportan e importan la base de datos al completo. Con ellas, no es posible desplegar la base de datos en cualquier sistema.

En cuanto a PostGIS, para que nuestro servidor PostgreSQL tenga soporte para tablas con contenido geográfico, también basta con descargar su ejecutable para que éste de despliegue sobre PostgreSQL.

9 CONCLUSIONES

"Que por poco no acabas conmigo, pero soy difícil de matar".

- Los Planetas -

Una vez desarrollada la aplicación objeto del proyecto, es posible analizar lo obtenido, lo aprendido, lo aplicado y las posibles mejoras que éste podría tener.

Se ha desarrollado un sistema que permite ser la base de cualquier proyecto SIG de mayor calado, resultando adaptable al ámbito en que se quiera utilizar y proporcionando al usuario una curva de aprendizaje más rápida que la del resto de soluciones del mercado.

Asimismo, la licencia de los SIG privados suele ser de gran coste. El sistema desarrollado en este proyecto está basado en software libre, con lo que podrá ser utilizado en proyectos sin una gran capacidad económica

Para ello, ha sido necesario apoyarse en multitud de tecnologías, siendo necesario lograr una interconexión entre ellas para el correcto funcionamiento del sistema.

Además de estas conclusiones del producto desarrollado; tras haberlo finalizado, veo necesario hacer una reflexión más personal.

Aunque durante el Grado hayamos realizado multitud de proyectos; enfrentarse a uno individual, sin una temática marcada desde fuera y sin el seguimiento habitual de una asignatura, ha resultado un gran reto.

También he aprendido sobre la importancia de la constancia en el trabajo, ya que éste se me ha alargado en demasía por la falta de ella. En cambio, cuando he podido dedicarle tiempo y esfuerzo, los resultados han sido palpables.

Por último, este trabajo ha sido la oportunidad de plasmar todas las enseñanzas no específicas que he obtenido durante el Grado. Las habilidades de ingeniar nuevas soluciones ante los problemas que iban surgiendo, planificar el desarrollo de un software, elaborar un documento técnico... Por ello, pienso que ha sido un buen colofón a los años en la Escuela.

9.1. Líneas de mejora

Al haber enfocado el proyecto como un sistema de administración, aún no aplicado a un SIG específico, existen multitud de posibilidades para ampliar el proyecto por este camino. Cada posible aplicación, de una gran diversidad de ámbitos, tendría distintas herramientas que facilitarían el ánalisis de los datos del visor.

Por otro lado, también habría sido posible lograr una mayor automatización de la aplicación. Especialmente, destaca el hecho del paso manual necesario para subir una capa al servidor GeoServer. Una versión en producción y a gran escala necesitaría que esta subida se realizara automáticamente en tiempo real.

Por último, de nuevo pensando en un uso en un entorno real, sería necesario mejorar la seguridad de la herramienta, aspecto al que, al no ser el fin del proyecto, no se ha dado una importancia excesiva. Por ejemplo, las contraseñas de los usuarios deberian estar codificadas y no deberían poder ser consultadas en la base da datos.

REFERENCIAS

[Introducción_1] Cita de Peña (2006) en PÉREZ NAVARRO, A. *Introducción a los sistemas de información geográfica y geotelemática* [en línea]. Editorial UOC, 2011. ISBN: 8497889339.

[Introducción_2] Cita de Dueker y Kjerne (1989) en PÉREZ NAVARRO, A. *Introducción a los sistemas de información geográfica y geotelemática* [en línea]. Editorial UOC, 2011. ISBN: 8497889339.

[Objetivos_1] ROMÁN, I. " Introducción a la Ingeniería de Software". En: Apuntes de Ingeniería de Software, del GITT de la Universidad de Sevilla [en línea]. Marzo 2016 [consulta: marzo 2016].

[Herramientas_1] *Apache Tomcat.* The Apache Software Foundation, ©1999-2018 [consulta: 22 de febrero 2018]. Disponible en: http://tomcat.apache.org/

[Herramientas_2] Interface Servlet. Apache Tomcat [consulta: 22 febrero 2018]. Disponible en: https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/Servlet.html

[Herramientas_3] WAR (archivo). Wikipedia, 17 febrero 2009 [consulta: 22 febrero 2018]. Disponible en: https://es.wikipedia.org/wiki/WAR_(archivo)

[Herramientas_4] *Geoserver.* Open Source Geospatial Foundation, ©2014 [consulta: 22 de febrero 2018]. Disponible en: http://geoserver.org/

[Herramientas_5] *PostgreSQL.* The PostgreSQL Global Development Group, ©1996-2018 [consulta: 22 de febrero 2018]. Disponible en: https://www.postgresql.org/

[Herramientas_6] *PostGIS.* Refractions Research [consulta: 22 de febrero 2018]. Disponible en: https://postgis.net/

[Herramientas_7] ¿Qué es el .NET? EMAGISTER Servicios de formación, 17 septiembre 2015 [consulta: 22 febrero 2018]. Disponible en: https://www.emagister.com/blog/que-es-el-net-para-que-sirve/

[Herramientas_8] Introducción al lenguaje C# y .NET Framework. Microsoft Docs, 20 julio 2015 [consulta: 23 febrero 2018]. Disponible en: https://docs.microsoft.com/es-es/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework

[Herramientas_9] *Bootstrap.* Autores de Bootstrap y Twitter, Inc., ©2011-2018 [consulta: 22 febrero 2018]. Disponible en: https://getbootstrap.com/

[Herramientas_10] BRANAS, R., CHANDERMANI A., FRISBIE, M. y HAVIV, A.Q. *AngularJS: Maintaining Web Applications* [en línea]. Birmingham, UK: Packt Publishing, 2016. ISBN: 9781786467362.

[Herramientas_11] OpenLayers. Wikipedia, 3 agosto 2010 [consulta: 22 febrero 2018]. Disponible en: https://es.wikipedia.org/wiki/OpenLayers

[ERS_1] ROMÁN, I. "Ingeniería de Requisitos". En: Apuntes de Ingeniería de Software, del GITT de la Universidad de Sevilla [en línea]. Marzo 2016 [consulta: marzo 2016].

[ERS_2] Qué es un shapefile. Documentación ArcMap [consulta: 1 marzo 2018]. Disponible en: http://desktop.arcgis.com/es/arcmap/10.3/manage-data/shapefiles/what-is-a-shapefile.htm

[ERS_3] *ArcGIS.* ESRI, ©2018 [consulta: 24 de febrero 2018]. Disponible en: http://www.esri.es/arcgis/

[ERS_4] Prólogo. Documentación de QGIS [consulta: 24 febrero 2018]. Disponible en: https://docs.qgis.org/2.18/es/docs/user_manual/preamble/foreword.html

[ERS_5] 27 Differences Between ArcGIS and QGIS. GISGeography, 26 junio 2018 [consulta: 1 marzo 2018]. Disponible en: http://gisgeography.com/qgis-arcgis-differences/

[ERS_6] ArcGIS vs QGIS. Monde Geospatial, 7 febrero 2017 [consulta: 24 febrero 2018]. Disponible en: http://monde-geospatial.com/arcgis-vs-qgis-10-most-important-differences-between-arcgis-and-qgis/

[ERS_7] Taxonomía de requisitos. Marco de Desarrollo de la Junta de Andalucía [consulta: 28 febrero 2018]. Disponible en: http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/408

[Implementación_1] SQL Create Table. W3Schools [consulta: 1 julio 2018]. Disponible en: https://www.w3schools.com/sql/sql_create_table.asp

[Implementación_2] Método String.Concat. Biblioteca de clases de .NET Framework [consulta: 1 julio 2018]. Disponible en: https://msdn.microsoft.com/es-es/library/a6d350wd(v=vs.110).aspx

[Implementación_3] SQL Insert. PostgreSQL [consulta: 1 julio 2018]. Disponible en: https://www.postgresql.org/docs/8.2/static/sql-insert.html

[Implementación_4] Método String.Replace. Biblioteca de clases de .NET Framework [consulta: 1 julio 2018]. Disponible en: https://msdn.microsoft.com/es-es/library/czx8s9ts(v=vs.110).aspx

[Implementación_5] Método ST_GeomFromGeoJSON. PostGIS [consulta: 1 julio 2018]. Disponible en: http://www.postgis.org/docs/ST_GeomFromGeoJSON.html

[Implementación_6] Método ST_GeomFromText. PostGIS [consulta: 1 julio 2018]. Disponible en: http://www.postgis.org/docs/ST_GeomFromText.html

[Implementación_7] Well Known Text. Wikipedia [consulta: 1 julio 2018]. Disponible en: https://es.wikipedia.org/wiki/Well_Known_Text

[Implementación_8] Método ST_Buffer. PostGIS [consulta: 1 julio 2018]. Disponible en: http://www.postgis.org/docs/ST_Buffer.html

[Implementación_9] Deserialize JSON to a simple Dictionary<string,string> in ASP.NET. StackOverflow [consulta: 1 julio 2018]. Disponible en: https://stackoverflow.com/questions/1207731/how-can-i-deserialize-json-to-a-simple-dictionarystring-string-in-asp-net

[Implementación_10] JavaScript Window Location. W3Schools [consulta: 1 julio 2018]. Disponible en: https://www.w3schools.com/js/js_window_location.asp

[Implementación_11] Draw and Modify Features. OpenLayers Examples [consulta: 1 julio 2018]. Disponible en: https://openlayers.org/en/latest/examples/draw-and-modify-features.html

[Implementación_12] Clear Vector. OpenLayers API [consulta: 1 julio 2018]. Disponible en: https://openlayers.org/en/latest/apidoc/ol.source.Vector.html#clear

[Implementación_13] OpenLayers 3 draw, edit and save data. CodePen [consulta: 1 julio 2018]. Disponible en: https://codepen.io/barbalex/pen/fBpyb

[Implementación_14] Feature. OpenLayers API [consulta: 1 julio 2018]. Disponible en: https://openlayers.org/en/latest/apidoc/ol.Feature.html

[Implementación_15] What projection is OpenLayers using. OpenLayers FAQ [consulta: 1 julio 2018]. Disponible en: https://openlayers.org/en/latest/doc/faq.html#what-projection-is-openlayers-using-

[Implementación_16] EPSG:3857 projection. OpenStreetMap Wiki [consulta: 1 julio 2018]. Disponible en: https://wiki.openstreetmap.org/wiki/EPSG:3857

[Implementación_17] Transform coordinates. EPSG.io: Coordinate systems worldwide [consulta: 1 julio 2018]. Disponible en: https://epsg.io/transform#s_srs=3857&t_srs=4326&x=-553118.6601627&y=4620365.4846032

[Implementación_18] Projection. OpenLayers API [consulta: 1 julio 2018]. Disponible en: https://openlayers.org/en/latest/apidoc/ol.proj.html

[Implementación_19] JavaScript Array splice() Method. W3Schools [consulta: 1 julio 2018]. Disponible en: https://www.w3schools.com/jsref/jsref_splice.asp

[Implementación_20] Publishing a PostGIS table. GeoServer User Manual [consulta: 1 julio 2018]. Disponible en: http://docs.geoserver.org/stable/en/user/gettingstarted/postgis-quickstart/index.html

[Implementación_21] How to get index value in ng-repeat? StackOverflow [consulta: 6 junio 2018]. Disponible en: https://stackoverflow.com/questions/41157855/angularjs-how-to-get-index-value-in-ng-repeat

[Implementación_22] AngularJS ngClass conditional. StackOverflow [consulta: 6 junio 2018]. Disponible en: https://stackoverflow.com/questions/16529825/angularjs-ngclass-conditional

[Implementación_23] How to use $index to change class in ng-repeat? StackOverflow [consulta: 6 junio 2018]. Disponible en: https://stackoverflow.com/questions/29827312/how-to-use-index-to-change-class-in-ng-repeat

[Implementación_24] Interpolation and data-binding. AngularJS Guide [consulta: 6 junio 2018]. Disponible en: https://docs.angularjs.org/guide/interpolation

[Implementación_25] How to set the id attribute of a HTML element dynamically with angularjs? StackOverflow [consulta: 6 junio 2018]. Disponible en: https://stackoverflow.com/questions/23655009/how-to-set-the-id-attribute-of-a-html-element-dynamically-with-angularjs-1-x

[Implementación_26] Elegant way to bring back column names in a Postgres query. StackOverflow [consulta: 6 junio 2018]. Disponible en: https://stackoverflow.com/questions/8499560/postgres-elegant-way-to-bring-back-column-names-in-a-query

[Implementación_27] Método fromLonLat. OpenLayers API [consulta: 9 junio 2018]. Disponible en: http://openlayers.org/en/v3.14.2/apidoc/ol.proj.html

[Implementación_28] $rootScope. AngularJS Docs [consulta: 9 junio 2018]. Disponible en: https://docs.angularjs.org/api/ng/service/$rootScope

[Despliegue_1] Como configurar y probar en el IIS un proyecto ASP.NET. Anexsoft [consulta: 9 junio 2018]. Disponible en: http://anexsoft.com/p/123/como-configurar-y-publicar-en-el-iis-un-proyecto-asp-net

[Despliegue_1] Overview of deployment in Visual Studio. Documentos de Visual Studio [consulta: 9 junio 2018]. Disponible en: http://anexsoft.com/p/123/como-configurar-y-publicar-en-el-iis-un-proyecto-asp-net

[Despliegue_3] Cómo exportar toda la BD con pgAdmin4. StackOverflow [consulta: 9 junio 2018]. Disponible en: https://es.stackoverflow.com/questions/124737/c%C3%B3mo-exportar-toda-la-bd-con-pgadmin4