UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

Grado en Ing. Sist. de Telecom., Sonido e Imagen

“Geolocalizador de antenas móviles”

TRABAJO FINAL DE GRADO

Autor/a:
Ana Ayet Pons

Tutor/a:
Jordi Bataller i Mascarell

GANDIA, 2014

GEOLOCALIZADOR DE ANTENAS MÓVILES

ABSTRACT:

The main objective in performing this degree final project is to provide to the Internet community a web app able to locate mobile phone antennas and to implement search filters so that users can also check coverages in certain areas.

It has created a dynamic application in which the user can supply data and photos and there is not an administrator in charge of collecting all the information.

Another goal is to improve and learn new knowledges for the development of my professional life.

Keywords: Project / map / tag / coverage/ ASP.NET

RESUMEN:

El principal objetivo a la hora de realizar este trabajo final de grado es aportar a la comunidad de Internet una aplicación web capaz de posicionar antenas de telefonía móvil y realizar filtros de búsqueda de manera que los usuarios puedan además comprobar coberturas de operadores en determinadas zonas.

Se ha creado una aplicación dinámica en la que el usuario puede aportar datos y fotos, de manera que no exista un administrador encargado de recopilar y colgar toda la información.

Otro objetivo a conseguir es ampliar y aprender nuevos conocimientos útiles para el desarrollo de mi vida profesional.

Palabras clave: Proyecto / mapa / tag / cobertura / ASP.NET

ÍNDICE

Capítulo 1. Introducción.

En este primer tema se describen los motivos por los que se va a realizar este proyecto, además de los objetivos que se pretenden cumplir.

Un segundo apartado habla de las aplicaciones actuales que podemos encontrar en Internet, de las ventajas e inconvenientes de cada una de las páginas web.

Por último, se citan los contenidos estudiados a lo largo del proyecto.

1.1. Motivación y objetivos.

Una necesidad que surge al usuario de teléfonos móviles antes o después es conocer la cobertura existente en una zona determinada para cada operador de telefonía que hay en el mercado; bien para elegir una determinada compañía o simplemente al realizar un desplazamiento puntual (trabajo, vacaciones) a un determinado lugar y conocer allí las posibilidades de comunicación.

En la actualidad, esta información es bastante incompleta y está dispersa, en el sentido de que cada operador ofrece sólo su propia información, siendo complicado encontrar la información referente a todos los operadores en una misma zona y así poder comparar.

Ante esta situación hemos pensado desarrollar una aplicación web permita centralizar esta información para que cualquier usuario pueda informarse fácilmente. Permitiremos que tanto personas ordinarias como los propios instaladores de antenas puedan recopilar la información sobre las coberturas en cada zona. La idea de uso es la siguiente: Un usuario se registra en esta aplicación, tras lo cual puede introducir datos sobre la existencia e identificación de torres, qué antenas tiene cada torre, y los datos importantes como tipo de cobertura, operador, potencia, fotos, y posicionamiento.

Evidentemente, según quién proporcione la información (que quedará identificado con un rol distinto: usuario, instalador o como operador) podrá dar más o menos detalles al respecto, siendo lo mínimo obligatorio el posicionamiento y el operador y cobertura de la antena. Por otra parte, cualquier usuario esté registrado o no podrá consultar los datos y fotos de la zona y operador/es que necesite.

1.2. Trabajo relacionado.

Desde el punto de vista de un usuario que quiere conocer la cobertura de telefonía en un área determinada, podemos encontrar las siguientes páginas web:

- www.coberturamovil.es: Página web que no está actualizada. Si introduces la población o dirección para geolocalizarte no te muestra las antenas que están a tu alrededor.
- www.movistar.es/particulares/coberturas/movil/: Web de Movistar que muestra si la cobertura es buena, muy buena o sin cobertura pero no indica dónde están las estaciones móviles ni podemos comparar con otros operadores.

- http://www.vodafone.es/conocenos/es/cobertura-y-tiendas/cobertura/consulta-de-cobertura-movil/: Al igual que la web anterior tenemos información de un solo operador móvil, sin poder saber qué estación móvil y compañía tenemos más cerca de nosotros.
- Les siguen en el buscador enlaces a webs de distintas compañías: http://lared.orange.es/, http://www.simyo.es/mapa-cobertura.html, etc. en las que al igual que los enlaces anteriores sólo podemos encontrar cobertura de un operador según los colores que establecen que indican si es buena, muy buena o sin cobertura.
- Si continuamos buscando información, esta vez en páginas oficiales nos encontramos con la web: http://geoportal.mityc.es/VCTEL/vcne.do que es la más completa, con datos fiables pero poco visuales.

Como se ve tenemos aplicaciones que muestran la cobertura de un solo operador o bien páginas web que tienen muchas menos funcionalidades de las que queremos que tenga la nuestra. Nuestra página intentará reunir las ventajas de todas ellas y cubrir sus inconvenientes.

1.3. Organización de este documento

En el capítulo 1, Introducción, estamos presentando las motivaciones y objetivos de este proyecto, así como una breve descripción de la perspectiva actual del mercado.

En el capítulo 2, Tecnología, se describen las tecnologías que hemos utilizado para el desarrollo del proyecto, hablaremos de programas de desarrollo, bases de datos, lenguajes de programación y paradigmas, todo ello de manera muy breve.

En el capítulo 3, Análisis de requerimientos del sistema, analizamos las exigencias tanto funcionales como no funcionales de la aplicación. Los requerimientos serán detalles técnicos o de diseño que debe cumplir nuestra página web.

En el capítulo 4, Clases y funciones, hablamos del diseño: describiremos las tablas de la base de datos, las clases y funciones y cómo se ha organizado.

En el capítulo 5, Implementación, comentamos los problemas que se nos han planteado durante el desarrollo del proyecto y cómo los hemos resuelto. Además describimos la estructura de directorios y ficheros.

En el capítulo 6, Manual de usuario, detallamos cómo funciona la aplicación, describiremos los roles de la aplicación y qué podemos hacer en cada pantalla y cómo podemos hacerlo.

En el último capítulo, Conclusiones y Trabajo futuro, tratamos de mostrar una visión más general del proyecto, aportando las conclusiones a las que hemos llegado tras el desarrollo de este trabajo final de grado. Además, se apuntan unas posibles ampliaciones o mejoras.

Capítulo 2. Tecnología.

En este capítulo describiremos las tecnologías y herramientas que hemos utilizado. Básicamente son las siguientes: SQL como lenguaje de programación para el acceso y manipulación base de datos, Visual Studio como programa de desarrollo para sistemas operativos Windows, ASP.NET como FrameWork de desarrollo para la creación de páginas web, C# como lenguaje de programación orientado a objetos, MVC como patrón estructural para organizar y estructurar los componentes de un sistema software. Css hace referencia a un lenguaje de hojas de estilos, JavaScript es un lenguaje de programación interpretado orientado a objetos y HTML es un lenguaje de marcado para la elaboración de páginas web.

2.1. SQL Server.

El SQL (Structured Query Language) es un lenguaje estándar de programación para el acceso y manipulación de los datos de cualquier base de datos.

El lenguaje de consulta estructurado (SQL) se utiliza para acceder y manipular datos en cualquier base de datos del mercado, como por ejemplo, para las bases de datos MySQL, Oracle, DB2, SQL Server, Access. Está compuesto por comandos, cláusulas, operadores y funciones de agregado. Estos elementos se combinan en las instrucciones para crear, actualizar y manipular las bases de datos.

SQL Server Management Studio presenta una interfaz gráfica para configurar, supervisar y administrar instancias de SQL Server. También permite implementar, supervisar y actualizar los componentes de capa de datos usados por las aplicaciones, como bases de datos y almacenamientos de datos.

2.2. Visual Studio.

Visual Studio es una colección completa de herramientas y servicios que le permitirá crear una gran variedad de aplicaciones, tanto para plataformas de Microsoft como para otras plataformas.

Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

2.3. ASP.NET.

ASP.NET es un FrameWork de desarrollo para la creación de páginas web y sitios web con HTML, CSS y para la creación de páginas web y sitios web con HTML, CSS, JavaScript y servidor de secuencias de comandos. ASP.NET está construido sobre el Common Language Runtime, permitiendo a los programadores escribir código ASP.NET usando cualquier lenguaje admitido por el .NET Framework.

La programación web es una mezcla de varios lenguajes de etiquetas, un gran uso de lenguajes de script y plataformas de servidor.

ASP.NET admite tres modelos de desarrollo diferentes:

- Páginas Web
- MVC (Model View Controller)
- Web Forms

2.4. C#.

C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET. C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.

C# es un lenguaje de programación independiente diseñado para generar programas sobre la plataforma .NET.

2.5. MVC.

ASP.NET MVC ofrece una alternativa para la creación de aplicaciones sobre la plataforma .NET muy cercana al funcionamiento de la web y a sus protocolos, con un control total sobre el marcado enviado al cliente, la posibilidad de generar URLs muy amigables y que nos ayudará a construir sistemas perfectamente estructurados, testeables, muy robustos y escalables.

MVC es un patrón arquitectural, un modelo o guía que expresa cómo organizar y estructurar los componentes de un sistema software, sus responsabilidades y las relaciones existentes entre cada uno de ellos. Su nombre, MVC, parte de las iniciales de Modelo-Vista-Controlador (Model-View-Controller) que son las capas o grupos de componentes en los que organizaremos nuestras aplicaciones bajo este paradigma.

La arquitectura MVC propone, independientemente de las tecnologías o entornos en los que se base el sistema a desarrollar, la separación de los componentes de una aplicación en tres grupos (o capas) principales: el modelo, la vista y el controlador, y describe cómo se relacionarán entre ellos para mantener una estructura organizada, limpia y con un acoplamiento mínimo entre las distintas capas.

- El Modelo contiene las entidades que representan el dominio, la lógica de negocio, y los mecanismos de persistencia de nuestro sistema.
- En la Vista encontraremos los componentes responsables de generar el interfaz con el exterior, normalmente el UI de nuestra aplicación.
- En el Controlador se encuentran los componentes capaces de procesar las interacciones del usuario, consultar o actualizar el modelo, y seleccionar las vistas apropiadas en cada momento.

Relación entre Modelo, Vista y Controlador:

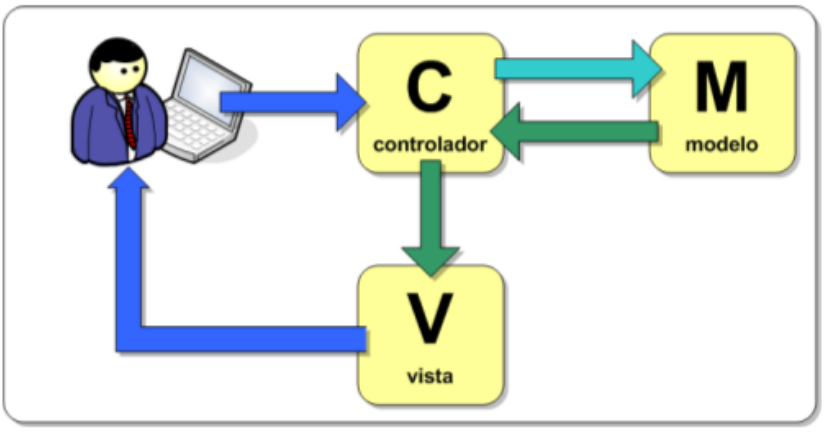


Figura 1. Diagrama de interacción entre Modelo, Vista y Controlador.

2.6. CSS.

Las hojas de estilo en cascada (Cascading Style Sheets, CSS) hacen referencia a un lenguaje de hojas de estilos usado para describir la presentación semántica (el aspecto y formato) de un documento escrito en lenguaje de marcas. Su aplicación más común es dar estilo a páginas webs escritas en lenguaje HTML y XHTML, pero también puede ser aplicado a cualquier tipo de documentos XML. La información de estilo puede ser adjuntada como en un documento separado o en el mismo documento HTML.

CSS tiene una sintaxis muy sencilla y consta de tres reglas. Cada regla consiste en uno o más selectores y un bloque de estilos con los estilos a aplicar para los elementos del documento que cumplan con el selector que les precede.

2.7. JavaScript.

JavaScript es un lenguaje de programación interpretado. Se define como orientado a objetos, basado en prototipos, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web.

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM). JavaScript se interpreta en el agente de usuario, al mismo tiempo que las sentencias van descargándose junto con el código HTML.

2.8. HTML.

HTML (Hyper Text Markup Language) hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que define una estructura básica y un código para la definición de contenido de una página web, como texto, imágenes, etc.

El lenguaje HTML basa su filosofía de desarrollo en la referenciación. De este modo, la página web contiene sólo texto mientras que recae en el navegador web (interpretador del código) la tarea de unir todos los elementos y visualizar la página final.

Capítulo 3. Análisis de requerimientos del sistema.

El objetivo de este apartado es obtener una especificación detallada del sistema que se quiere desarrollar.

Una vez leído este punto sabremos qué debe hacer la aplicación y cómo tiene que funcionar.

3.1. Requerimientos funcionales.

Los requerimientos funcionales definen las funciones del sistema de software o sus componentes, responden a la pregunta de qué debe hacer el sistema.

Queremos construir una aplicación web que realice las siguientes funciones:

- Posicionamiento de las antenas independientemente del zoom que tengamos en el mapa y fotos de la antena.

Las webs que tenemos en la actualidad y que muestran las estaciones base sólo las muestran cuando el zoom aumenta la zona notablemente, nosotros posicionaremos la estación base independientemente del zoom del mapa.

- Información al usuario de las estaciones móviles: posicionamiento, fotos y tags.

Cualquier usuario podrá conocer la situación exacta de cada antena de los distintos operadores móviles, además podrá acceder a las fotos colgadas y buscar estaciones móviles mediante tags.

- Registro de usuarios, empresas instaladoras y operadores.

Los usuarios, empresas instaladoras y operadores registrados introducirán los datos necesarios para que puedan acceder a ciertas pantallas de la página web no visibles para el resto de usuarios.

Se podrá crear expedientes y actualizarlos de manera que quede toda la información de una torre en un solo lugar.

En estos se implantará una estructura de una torre por expediente, con sus características técnicas y todas las antenas asociadas a esa torre, con sus respectivas características.

Se completará la información con fotos y con tags que asociaremos a éstas.

3.2. Requerimientos no funcionales.

Los requisitos no funcionales se refieren a características del sistema relacionadas con la plataforma, el rendimiento, etc. Se utilizan para delimitar los requisitos funcionales al imponer condiciones a los mismos.

- Disponibilidad.

La aplicación web se alojará en un hosting, de esta forma estará siempre disponible al usuario, empresa u operador que quiera acceder a ella.

- Usabilidad.

La aplicación web podrá utilizarse en todo tipo de dispositivos, desde ordenadores hasta tabletas o smartphones, puesto que tiene un diseño (Responsive Design) que se adapta al tamaño de pantalla automáticamente.

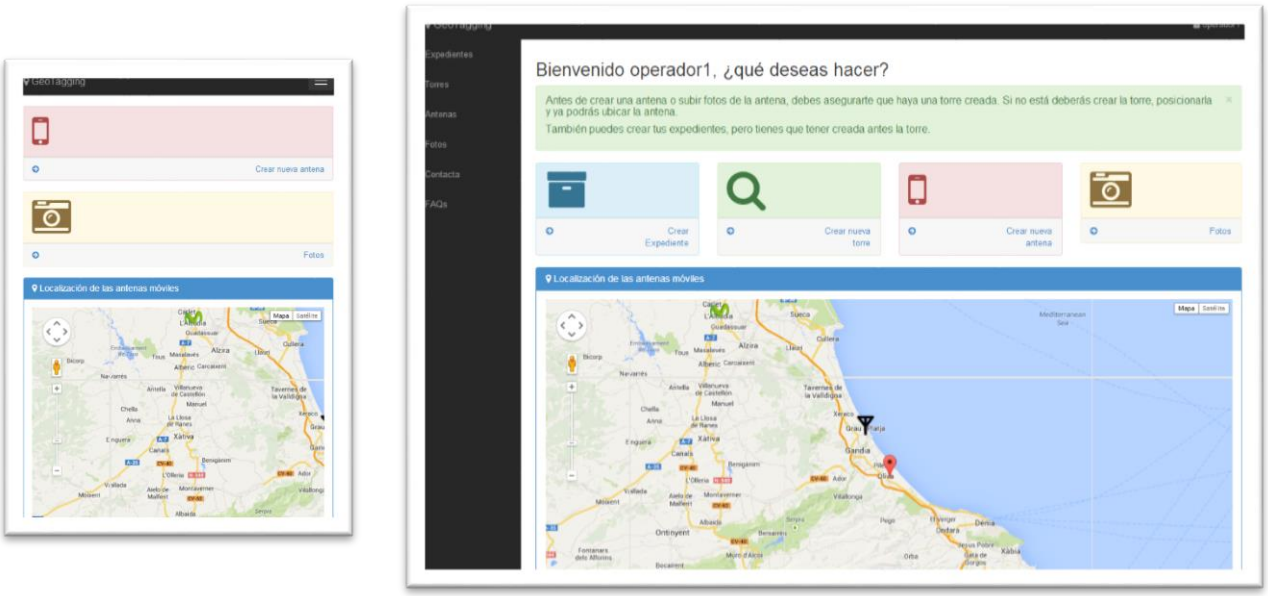


Figura 2. Diferentes vistas de la aplicación según el tipo de dispositivo o resolución de pantalla.

- Escalabilidad.

Mediante el paradigma de programación MVC hemos conseguido que la web pueda ir ampliándose conforme las necesidades de nuestros clientes.

- Mantenibilidad y costo.

La aplicación se actualiza conforme van subiendo datos usuarios, empresas y operadores. Por tanto, no necesita mantenimiento alguno. El costo total de la web sería el producido por el alojamiento en un hosting, pero que se podría financiar con publicidad o bien tal como hemos decidido, buscando un hosting gratuito.

- Interoperatibilidad.

Al tratarse de una aplicación web con Responsive Design puede ser usada por cualquier tipo de usuario, independientemente del sistema operativo que utilice o el tipo de dispositivo en el que estemos.

- Exactitud.

La exactitud (Longitud y latitud) del sistema tiene que ser muy alta ya que no es permisible que se dé información errónea a los usuarios.

Capítulo 4. Diseño de la aplicación.

En este capítulo trataremos de describir los pasos que hemos seguido antes de la puesta en marcha de la aplicación.

Lo primero que se ha creado es la base de datos [4], una vez diseñada la base de datos y hechas las relaciones entre las tablas relacionaremos el código de programación con nuestra bbdd mediante un interfaz que proporciona MVC de manera que, automáticamente, se genera una estructura de las clases según las bases de datos [1].

Una vez tenemos la relación y estructura pasaremos a programar la aplicación tal como MVC estructura: modelo, vista, controlador.

A su vez, en MVC4 se ha instalado el administrador de paquetes NuGet mediante el cual podemos aprovechar APIs de terceros. En nuestro proyecto se aprovechó para instalar la utilidad desarrollada para la gestión de mapas de google, que incluye una serie de clases, por ej. GoogleMarker, que hemos utilizado para crear ubicaciones que después situamos en el mapa. Gracias al paquete de GoogleMaps se simplifica notablemente la programación en JavaScript [10].

También se ha instalado el paquete de Nuggets MVCMailer [7], el cual incluye una interfaz de vistas para poder enviar correos electrónicos desde el proyecto de MVC. El paquete consiste en la creación de una clase donde aparecerán todos los tipos de correo electrónico como métodos y cada uno de ellos llevará asociada una vista, por lo que desde cualquier controlador podremos acceder a dichos métodos y empaquetar la vista para enviarla al destinario elegido.

El código se ha programado en C# [2] con las ventajas del Framework 4 para poder obtener muchas más funcionalidades como Linq que permite realizar el acceso y gestión de datos de forma mucho más eficiente que construir querys desde el inicio.

Se ha estructurado en Modelo-Vista- Controlador separando de esta manera los datos (Modelo) de la Vista y ejecutándose a través del Controlador.

En las vistas hemos utilizado lenguaje XHTML [5] para el código, CSS para los estilos y JavaScript [6] para dinamizar la página web.

4.1. Base de datos de la aplicación.

Para crear nuestra aplicación empezaremos creando nuestra base de datos en SQL Server 2012 Management Studio [4]. En esta almacenaremos toda la información que vayamos guardando desde la web.

Tal como hemos citado, como herramienta de desarrollo se ha utilizado Visual Studio y se ha estructurado el código utilizando el paradigma MVC. Para conectar con la base de datos, se ha utilizado el interfaz que proporciona el “Entity Data Model”, el cual genera de forma automática una estructura de clases según las tablas y relaciones de la base de datos (además, esta estructura permite actualizarse si hay cambios en la base de datos).

MVC4 también facilita la programación en otros muchos aspectos, en nuestro caso podíamos elegir entre crear nosotros el código de programación para autentificar usuarios o aprovecharnos de sus ventajas que es por lo que se optó, ya que se incluye por defecto el “SimpleMembership provider”, que es un proveedor de autenticación con una estructura base de tablas, en las que se almacena

tanto los usuarios y contraseñas encriptadas así como otra información relevante como las fechas de acceso y la gestión de roles de usuarios.

4.1.1. Diseño de la Base de Datos.

Nuestra base de datos SQL se llama geotagging y está compuesta de 13 tablas creadas por nosotros más 4 tablas creadas automáticamente debido al uso de Simple Membership provider como proveedor de servicios de autenticación. Estas tablas automáticas almacenan los datos de acceso, nombre de usuario y contraseña encriptada, además proporcionan la interfaz de login utilizando proveedores de terceros como Google. En los siguientes puntos detallaremos cada una de ellas.

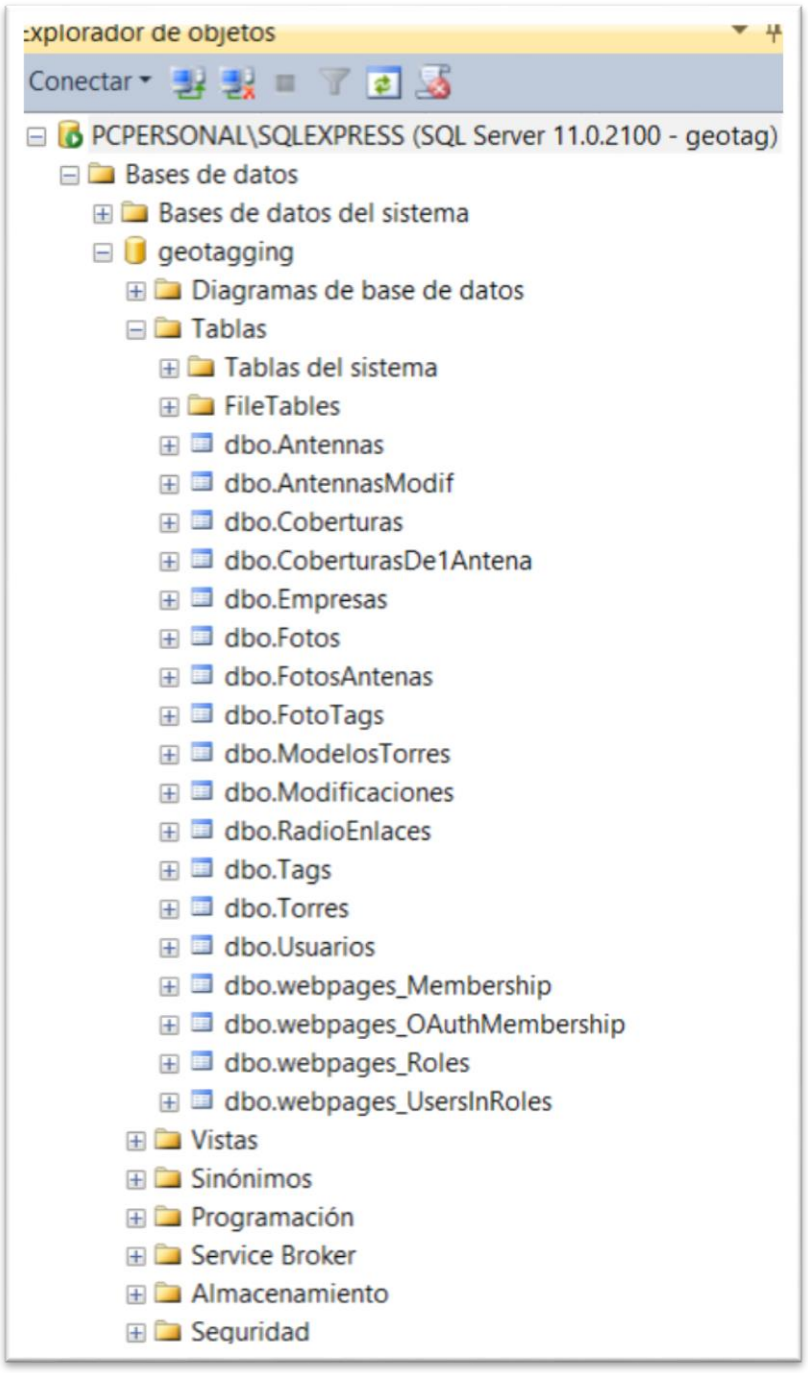


Figura 3. Detalle de la base de datos geotagging.

Una vez diseñada la base de datos la conexión con la herramienta de desarrollo se realizará utilizado el interfaz que proporciona el “Entity Data Model”, el cual automáticamente crea las tablas y las relaciones entre ellas.

4.1.2. Diagrama de la base de datos.

Una vez relacionada con nuestra base de datos con nuestro entorno de programación obtendríamos un diagrama tal como el que se muestra abajo:

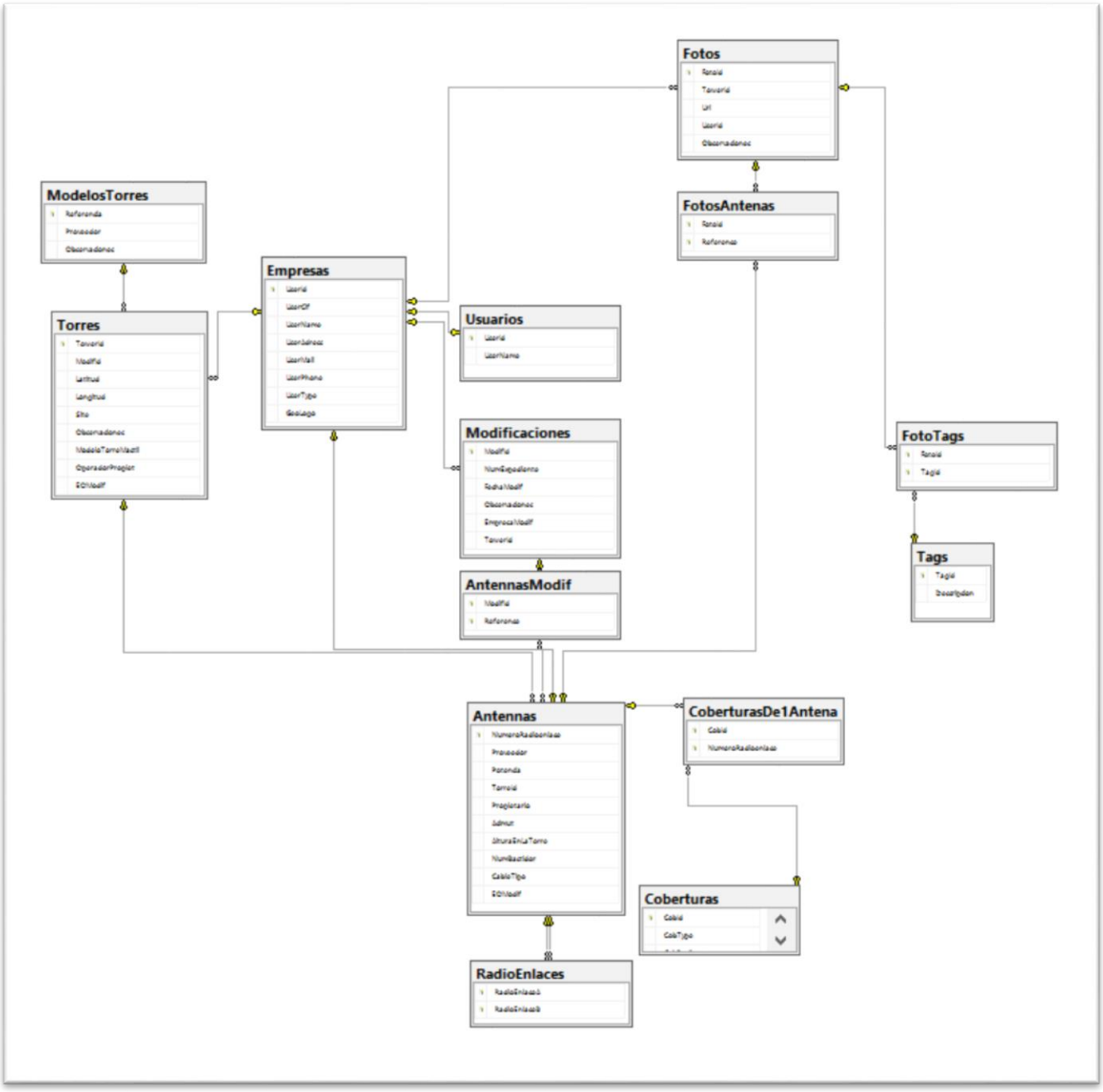


Figura 4. Diagrama de la base de datos.

4.1.3. Tablas de la base de datos.

A continuación detallaremos las tablas más importantes, explicando las variables de cada una de ellas.

Tabla 1: Antenas.

Antennas

	Nombre de columna	Tipo comprimido
🔑	NumeroRadioenlace	nvarchar(50)
	Proveedor	nvarchar(50)
	Potencia	int
	TorreId	int
	Propietario	int
	Acimut	decimal(20, 5)
	AlturaEnLaTorre	decimal(20, 5)
	NumBastidor	nvarchar(50)
	CableTipo	nvarchar(50)
	EOModif	int

-NumeroRadioenlace: Serie de caracteres que identifican la antena de manera única.
-Proveedor: Fabricante de la antena.
-Potencia: Valor en Vatios (W) de la potencia de la antena.
-TorreId: Indica la torre a la que está asociada la antena.
-Propietario: Operador de la antena.
-Acimut: Grados de la elevación de la antena.
-AlturaEnLaTorre: Altura a la que está instalada la antena en la torre.
-NumBastidor: Serie que identifica la antena.
-CableTipo: Cable que se utiliza para realizar la conexión de la antena.
-EOModif: Empresa que ha hecho la instalación de la antena.

Tabla 2: Antenas-Modificaciones.

AntennasModif

	Nombre de columna	Tipo comprimido
🔑	ModifId	int
🔑	Reference	nvarchar(50)

Tabla que enlaza las antenas con la tabla modificaciones:
-ModifId: Identificador de la modificación/ expediente que se ha realizado.
-Reference: Identificador de una antena, es único para cada antena. Se corresponde con el campo NumeroRadioenlace en la tabla Antenas.

Tabla 3: Coberturas.

Coberturas

	Nombre de columna	Tipo comprimido
🔑	CobId	int
	CobType	nvarchar(100)
	CobRadio	decimal(5, 2)

Tabla de coberturas de telefonía móvil que existen en España.
-CobId: Identificador de la tabla para las diferentes coberturas.
-CobType: Tipo de Cobertura: GSM, 3G, etc.
-CobRadio: Distancia que cubre esa antena.

Tabla 4: Coberturas de 1 antena.

CoberturasDe1Antena

🔑	CobId
🔑	NumeroRadioenlace

Relaciona la tabla de coberturas que tenemos con la antena a la que queremos asociarle las coberturas.
-CobId: Identificador de la tabla para las diferentes coberturas.
-NumeroRadioenlace: Serie de caracteres que identifican la antena de manera única.

Tabla 5: Empresas- Operadores.

Empresas

	Nombre de columna	Tipo comprimido
🔑	UserId	int
	UserCif	nvarchar(10)
	UserName	nvarchar(50)
	UserAdress	nvarchar(50)
	UserMail	nvarchar(50)
	UserPhone	nchar(9)
	UserType	int
	GeoLogo	nvarchar(100)

Tabla que recoge los datos necesarios para registrarse todos los usuarios.
-UserId: Identificador único de una empresa, operador o usuario.
-UserCif: DNI/CIF del usuario que se registra en la web.
-UserName: Nombre.
-UserAdress: Dirección fiscal de la empresa, operador o dirección del usuario.
-UserMail: E-Mail.
-UserPhone: Teléfono de contacto.
-UserType: Campo que distingue si el que se registra es una empresa (0), un operador (1) o un usuario (2).
-GeoLogo: Campo en el que se guarda el nombre del archivo del logo del operador para después utilizarlo en el mapa.

Tabla 6: Fotos.

Fotos

	Nombre de columna	Tipo comprimido
🔑	FotoId	int
	TowerId	int
	Url	nvarchar(250)
	UserId	int
	Observaciones	nvarchar(MAX)

-FotoId: Identifica de manera única una foto.
-TowerId: Identificador que relaciona la foto con la torre en la que se encuentra.
-Url: Dirección donde se encuentra la foto.
-UserId: Relaciona la foto con el usuario que la ha subido (empresa u operador).
-Observaciones: Campo opcional en el que podemos anotar datos de la foto.

Tabla 7: Fotos-Antenas.

FotosAntenas

	Nombre de columna	Tipo comprimido
🔑	FotoId	int
🔑	Reference	nvarchar(50)

Relaciona la tabla fotos con la tabla antenas para poder asignar una foto a una antena.
-FotoId: Identifica de manera única una foto.
-Reference: Identificador de una antena, es único para cada antena. Se corresponde con el campo NumeroRadioenlace en la tabla Antenas.

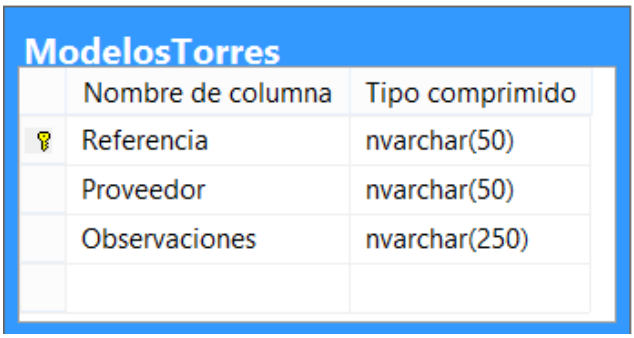
Tabla 8: Foto-Tags.

FotoTags

	Nombre de columna	Tipo comprimido
🔑	FotoId	int
🔑	TagId	int

Relaciona la tabla fotos con la tabla de tags, de manera que se puedan asignar uno o más tags a una foto.
-FotoId: Identificador único de una foto.
-TagId: Identificador único de un tag.

Tabla 9: Modelos de las torres.

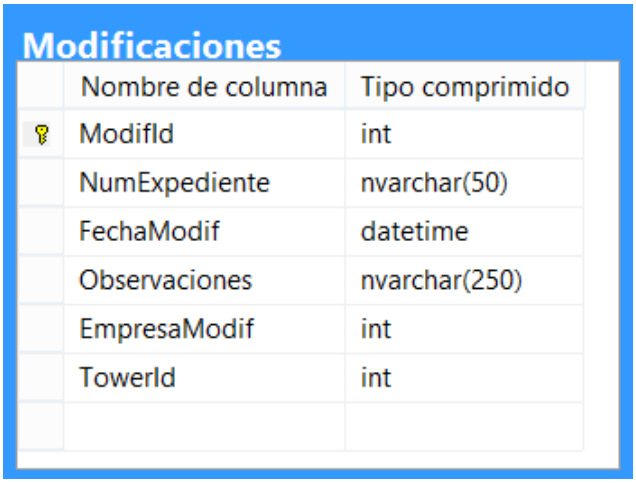


ModelosTorres

Nombre de columna	Tipo comprimido
Referencia	nvarchar(50)
Proveedor	nvarchar(50)
Observaciones	nvarchar(250)

Especifica las características de una torre.
-Referencia: Campo que identifica de manera única un modelo de torre.
-Proveedor: Fabricante de la torre.
-Observaciones: Campo opcional en el cual podemos añadir distintas características técnicas que queramos destacar.

Tabla 10: Modificaciones o expedientes.

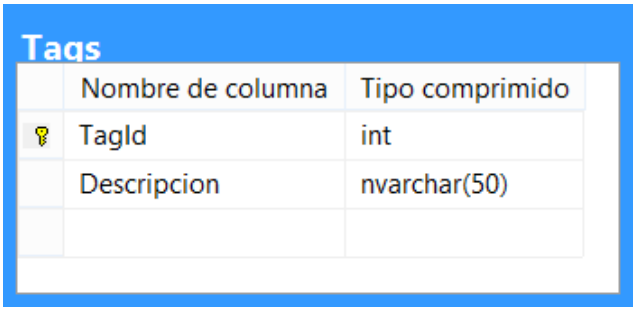


Modificaciones

Nombre de columna	Tipo comprimido
ModifId	int
NumExpediente	nvarchar(50)
FechaModif	datetime
Observaciones	nvarchar(250)
EmpresaModif	int
TowerId	int

En esta tabla se guarda los datos cada vez que queramos abrir un expediente nuevo o realizar modificaciones en uno existente (sólo es visible para una empresa o un operador).
-ModifId: Identifica de manera única un expediente o modificación realizada en él.
-NumExpediente: Caracteres que identifican ese expediente según la nomenclatura del operador.
-FechaModif: Fecha en que se ha creado o modificado el expediente.
-Observaciones: Campo opcional por si queremos destacar algo.
-EmpresaModif: Empresa que ha realizado el expediente o modificación.
-TowerId: Identificador de la torre a la que pertenece ese expediente.

Tabla 11: Tags.

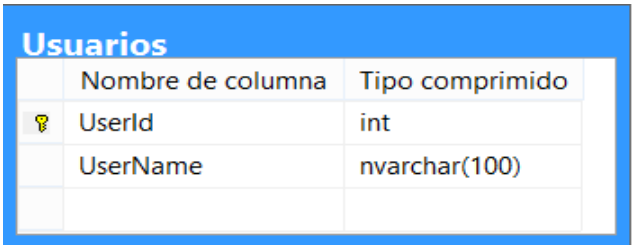


Tags

Nombre de columna	Tipo comprimido
TagId	int
Descripcion	nvarchar(50)

-TagId: Identificador del tag.
-Descripción: Campo opcional que ayuda con información adicional al nombre del tag.

Tabla 12: Usuarios.



Usuarios

Nombre de columna	Tipo comprimido
UserId	int
UserName	nvarchar(100)

Se ha creado para relacionar la tabla empresas con las que crea automáticamente la bbdd.

-UserId: Identificador del usuario.
-UserName: Nombre del usuario.

Tabla 13: Torres

Torres

	Nombre de columna	Tipo comprimido
🔑	TowerId	int
	ModifId	int
	Latitud	decimal(20, 10)
	Longitud	decimal(20, 10)
	Site	nvarchar(50)
	Observaciones	nvarchar(250)
	ModeloTorreMastil	nvarchar(50)
	OperadorPropiet	int
	EOModif	int

Tabla que recoge los parámetros de una torre.
-TowerId: Identificador único de una torre.
-ModifId: identificador del expediente.
-Latitud-Longitud: posición de la torre.
-Site: Nombre de la torre.
-Observaciones: Campo opcional por si queremos especificar más características de la torre.
-ModeloTorreMastil: Referencia del mástil de la torre.
-OperadorPropiet: Operador al que le pertenece la torre.
-EOModif: Empresa que ha instalado o modificado la torre.

Las siguientes tablas se han creado automáticamente.

Tabla 14: Usuarios registrados.

webpages_Membership

	Nombre de columna	Tipo comprimido
🔑	UserId	int
	CreateDate	datetime
	ConfirmationToken	nvarchar(128)
	IsConfirmed	bit
	LastPasswordFailur...	datetime
	PasswordFailuresSi...	int
	Password	nvarchar(128)
	PasswordChanged...	datetime
	PasswordSalt	nvarchar(128)
	PasswordVerificatio...	nvarchar(128)
	PasswordVerificatio...	datetime

Tabla en la que se recogen el usuario y la contraseña.

Esta tabla es fundamental para el registro de un usuario nuevo en la web. Guarda el userId, la contraseña encriptada, y otros datos como la fecha de creación de la contraseña y la última fecha en que se modificó la contraseña.

Tabla 15: Autentificación de usuarios registrados.

webpages_OAuthMembership

	Nombre de columna	Tipo comprimido
🔑	Provider	nvarchar(30)
🔑	ProviderUserId	nvarchar(100)
	UserId	int

Tabla de proveedores externos, por ejemplo Google.

4.2. Clases y funciones.

Se ha organizado la información según el patrón modelo-vista-controlador. Para ello empezaremos detallando los métodos del modelo, seguiremos con el controlador y acabaremos explicando las vistas.

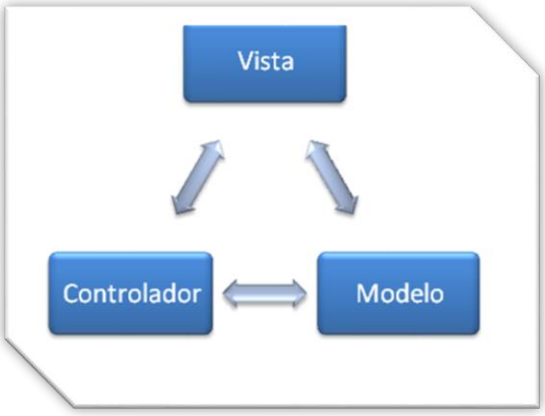


Figura 5: Esquema Modelo-Vista-Controlador.

4.2.1. Modelo.

Como se ha comentado más arriba nuestro modelo se compone de diferentes clases. Podemos distinguir las que hemos creado nosotros y aquellas creadas por el Entity Data Model a partir de la bbdd, que serán incluidas en el Anexo.

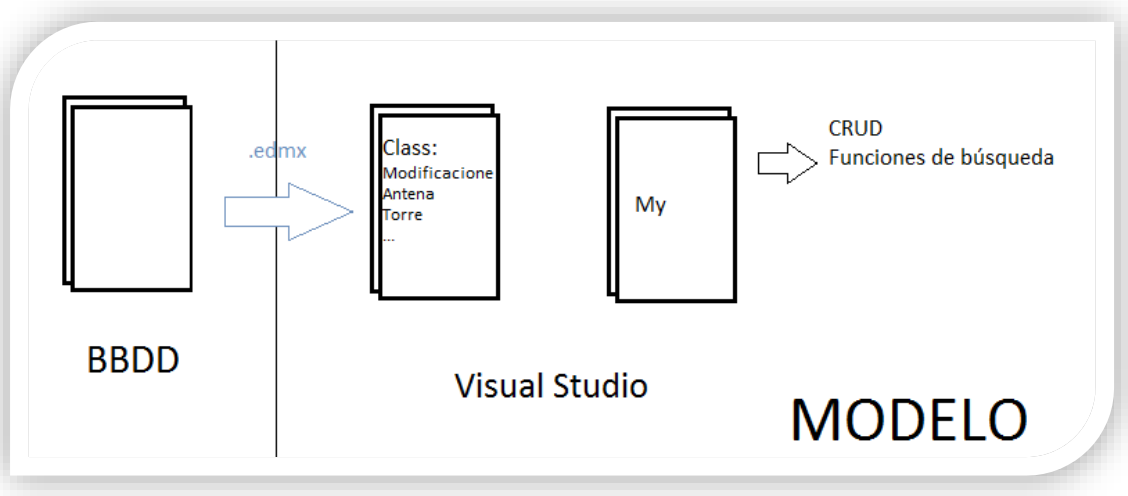


Figura 6. Relación de las diferentes clases del modelo.

Una vez establecidas las diferentes clases automáticas se ha pasado a crear diferentes clases que hemos necesitado para el desarrollo de la aplicación.

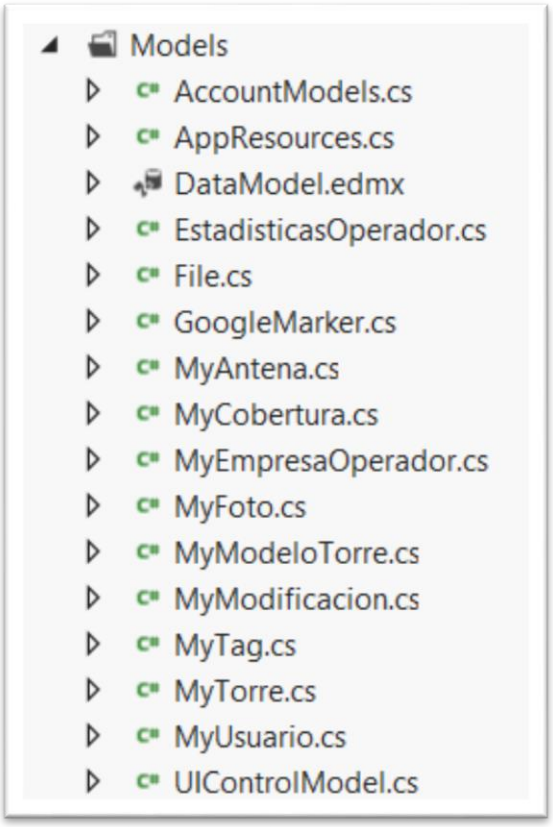
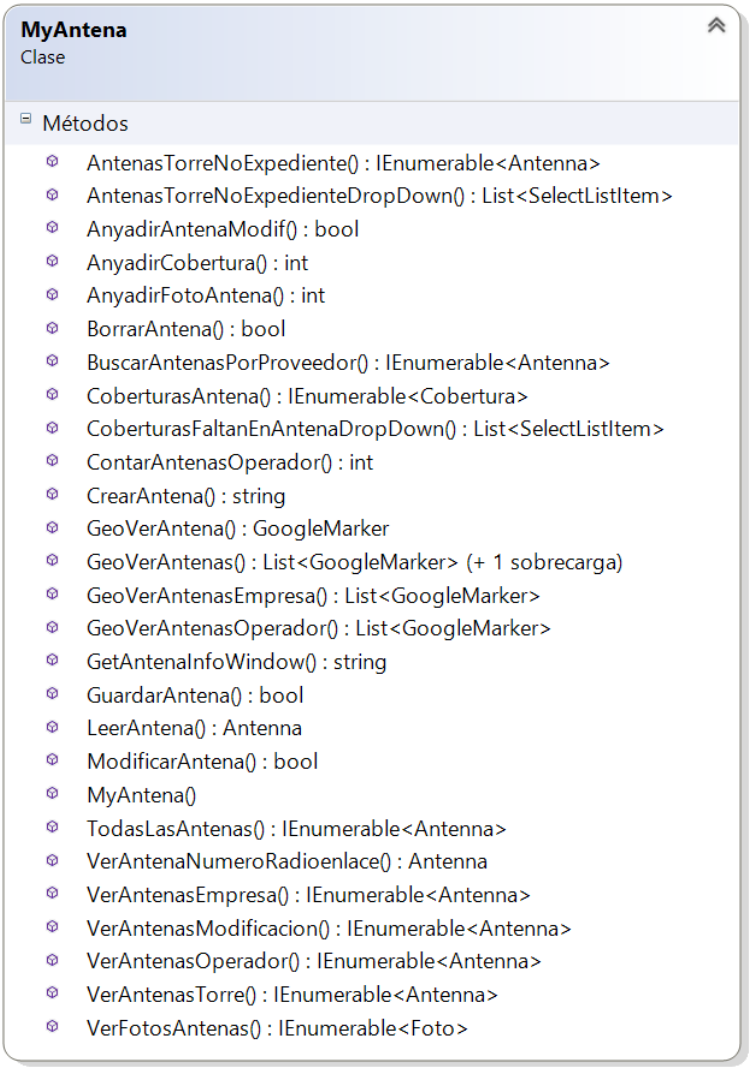
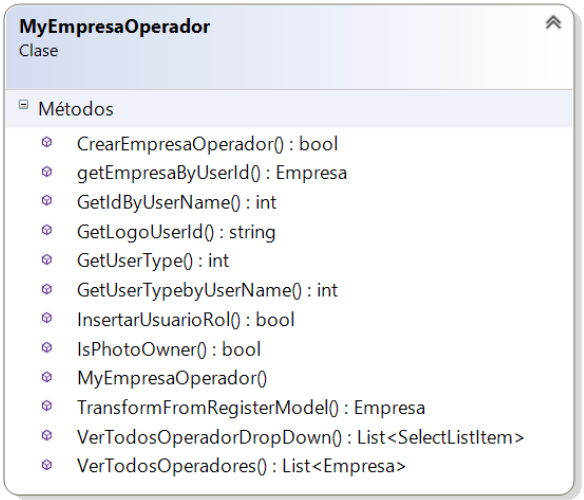


Figura 7. Clases creadas.

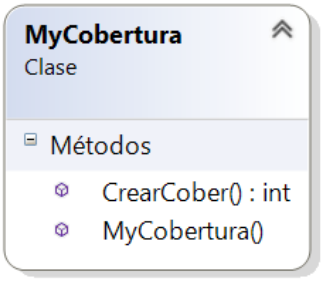
Seguidamente, explicaremos algunas de las clases más importantes:



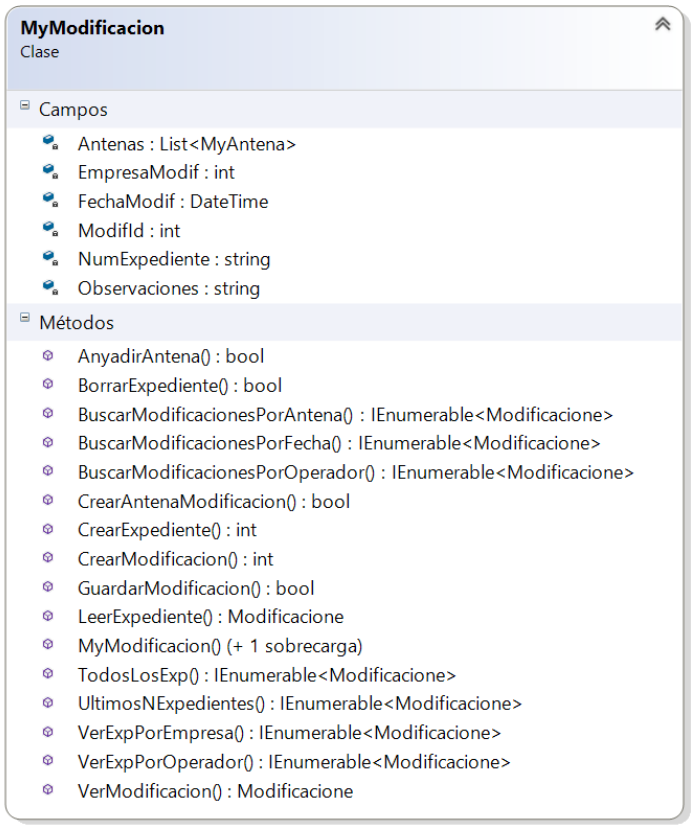
La clase MyAntena ha sido creada por nosotros y contiene todos los métodos referentes a las antenas.



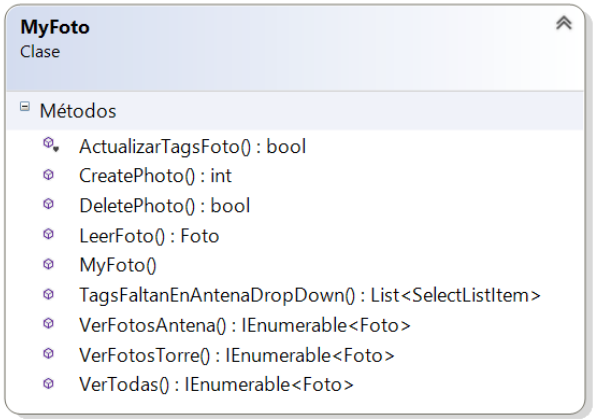
Esta clase recoge los métodos que necesitamos crear para trabajar con los usuarios, operadores o empresas instaladoras.



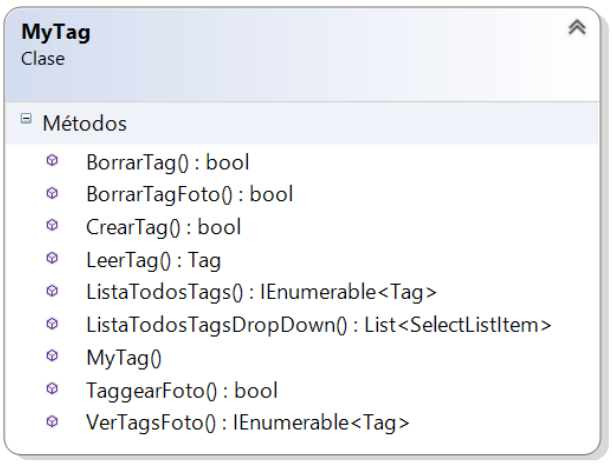
Clase que nos muestra los distintos tipos de cobertura móvil que puede tener una antena.



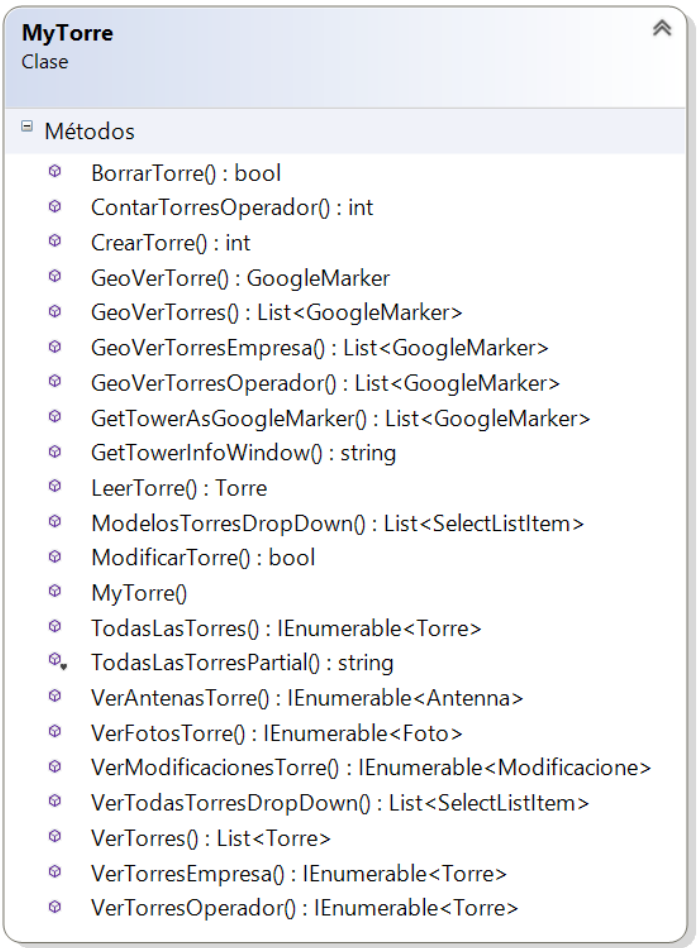
Clase que se utiliza en los perfiles de operador y empresa instaladora para poder realizar operaciones CRUD en los expedientes.



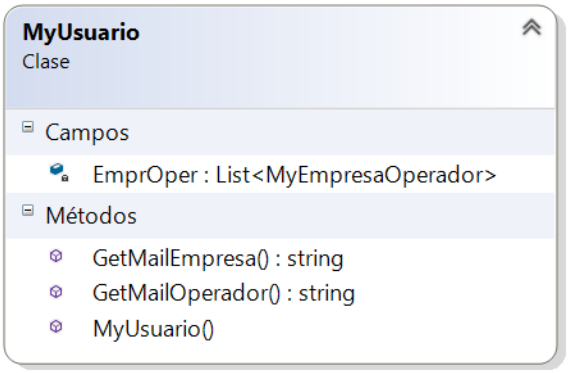
Clase que incluye los métodos relacionados con las fotos.



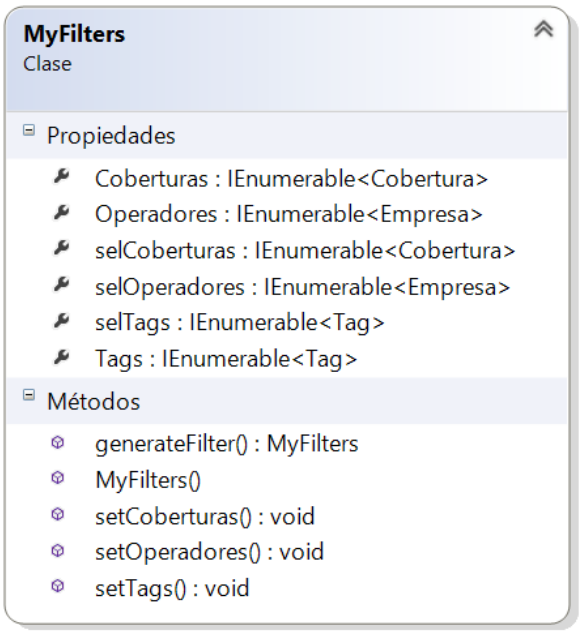
Clase que recoge los métodos creados para etiquetar fotos y cambiarlas las etiquetas si se desea.



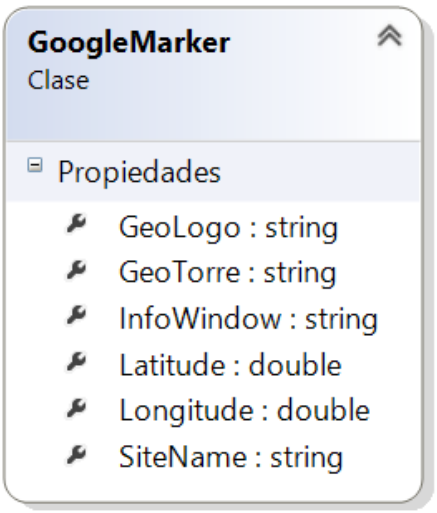
Clase que recoge todos los métodos creados para una torre: operaciones CRUD y mapas.



Incluye los métodos creados para poder recoger el mail de un usuario.



Con esta clase conseguimos hacer filtros en la página principal en función de lo que demande el usuario.



También, en nuestra aplicación se ha creado una clase para la gestión de mapas de google. En ella se guardarán las variables necesarias a la hora de posicionar una torre.

4.2.2. Controlador.

Una vez estructuradas las clases el siguiente paso es crear los controladores necesarios. El controlador describe cómo se relacionan entre ellos para mantener una estructura organizada, limpia y con un acoplamiento entre las distintas capas.

Su misión es actuar como intermediario entre el usuario y el sistema. Así pues hemos creado los siguientes controladores.

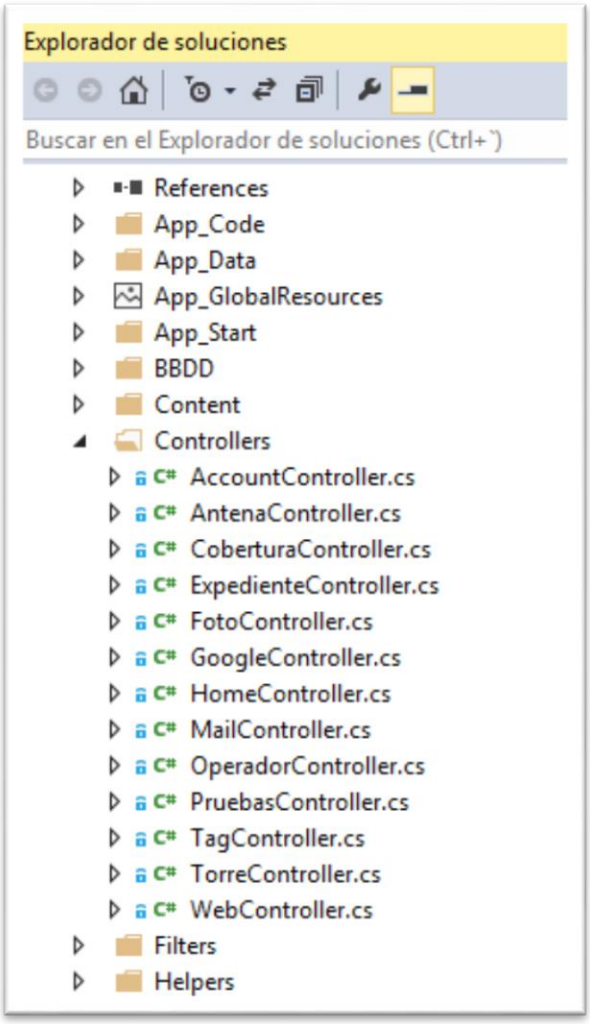


Figura 8. Controladores de la aplicación.

En el controlador se encuentran los componentes capaces de procesar las interacciones del usuario, consultar o actualizar el modelo, y seleccionar las vistas apropiadas en cada momento.

```
public ActionResult BorrarTorre(int towerId)
{
    MyTorre to6 = new MyTorre();
    return View(to6.LeerTorre(towerId));

}

[HttpPost]
public ActionResult BorrarTorre (Torre torre)
{
 //   ViewData["elementos"] = n;
    int c= torre.OperadorPropiet;
    MyTorre to6 = new MyTorre();
    bool a = to6.BorrarTorre(torre);
    if (a)
    {
        return RedirectToAction("VerTorresOperador", "Torre", new { userName=HttpContext.User.Identity.Name});
    }
    else
    {
        return View(torre);
    }
}
```

Figura 9. Vista de un método del controlador TorreController.

4.2.3. Vistas.

Los componentes de la vista son los responsable de generar el interfaz de nuestra aplicación, es decir, de componer las pantallas que van a utilizar los usuarios del sistema.

Es aquí donde encontraremos los componentes capaces de generar el lenguaje de marcado de la página que será enviado al usuario. Y además, por realizarse la programación en ASP.NET MVC, utilizaremos archivos .cshtml de Razor y emplearemos como lenguaje de marcado HTML y Javascript.

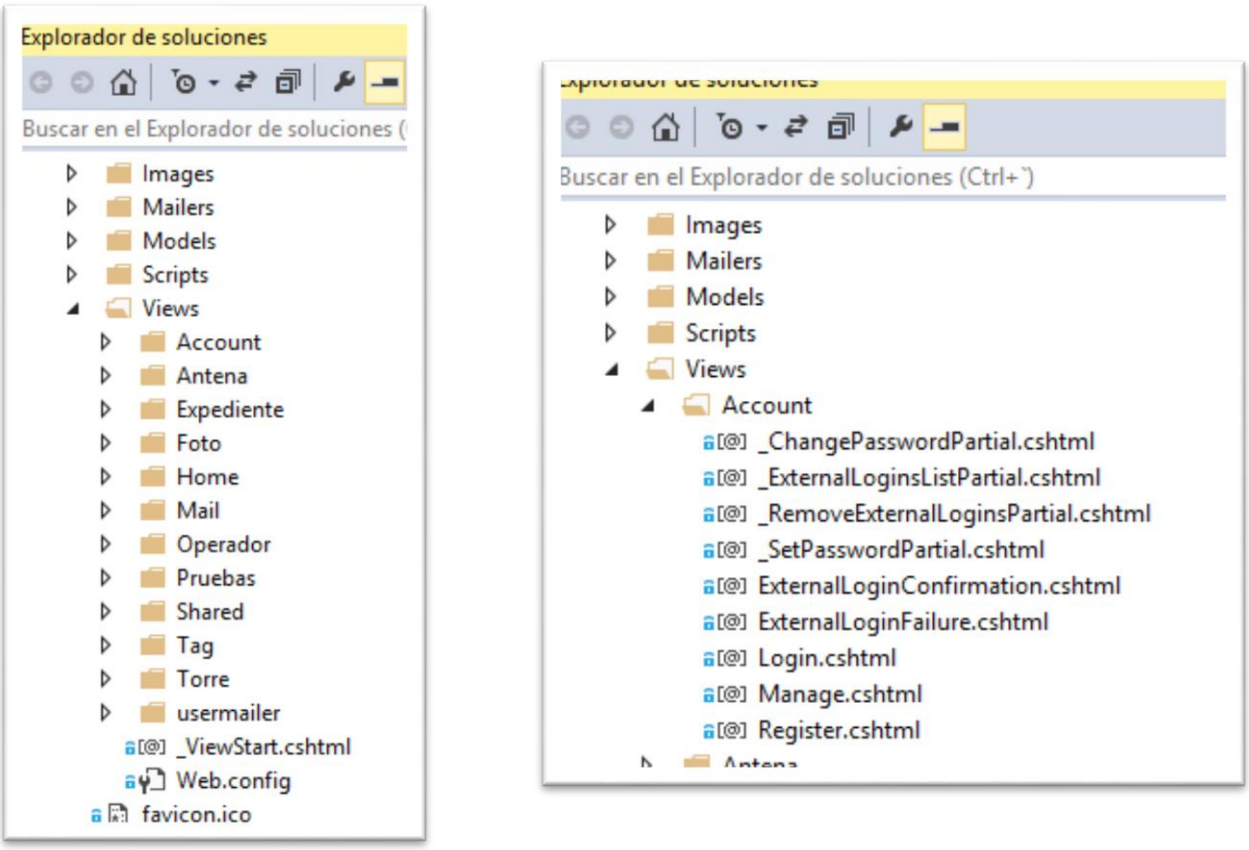


Figura 10. Vistas creadas en el proyecto. A la derecha vistas de la carpeta “Account”.

En el proyecto desarrollado distinguiremos dos tipos de vistas, aquellas que ocupan toda la pantalla y aquellas que ocupan una porción de página.

En una vista parcial el tipo de contenido es muy similar al de las vistas de contenido, aunque normalmente será más simple debido a que su objetivo se reduce a generar una porción de página.

4.3. Marcado Html y css.

4.3.1. HTML.

Html (Hyper Text Markup Language) hace referencia al lenguaje de marcado que hemos utilizado para la elaboración de páginas web.

Este lenguaje se ha utilizado a la hora de crear una vista. Tiene la estructura de una página web y está programada mediante código Html y js.

4.3.2. Responsive Design.

Además, en el código de la aplicación se ha utilizado la filosofía de diseño “Responsive Design”. Con esta técnica creamos un diseño web adaptable con código HTML y CSS.

Mediante “Responsive Design” no es necesario crear aplicaciones “ad-hoc” y evita tener que crear aplicaciones para por ejemplo aplicaciones iPhone o Android.

Se trata de marcar como bloques la estructura de la página web de manera que esos bloques vayan ajustándose conforme la resolución de la web.

```
@model GeoTagging.Models.LoginModel

<section id="loginForm" class="contact">
    @using (Html.BeginForm(new { ReturnUrl = ViewBag.ReturnUrl })) {
    @Html.AntiForgeryToken()
        @Html.ValidationSummary(true)

        <div class="row">
            <div class="col-lg-12">
                <ol class="breadcrumb">
                    <h2><i class="fa fa-user"> Iniciar sesión</i></h2>
                </ol>
            </div>
        </div><!-- /.row -->

        <div class="row">
            <div class="col-lg-12">
                <div class="well">
                    <fieldset>
                        <legend>Formulario de inicio de sesión</legend>
```

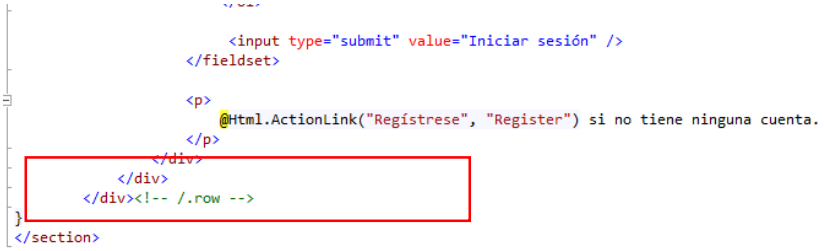


Figura 11. Vista de parte del código HTML en una vista. Se ha marcado el código “Responsive Design”.

4.3.3. Código CSS.

Son los archivos responsables de los estilos de la aplicación. En nuestro caso estos archivos se guardan en una carpeta llamada css.

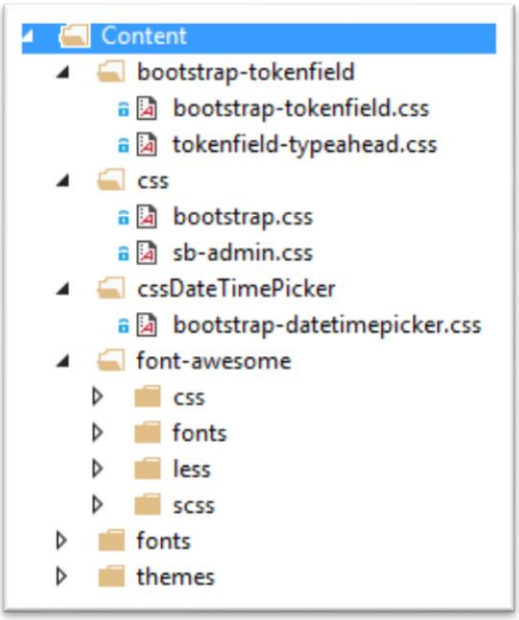


Figura 12. Carpeta que contiene los archivos css.

Para nuestra página web hemos utilizado más de un archivo css:

- Bootstrap-tokenfield: incluye los estilos necesarios para crear los tags.
- Bootstrap: se encuentran los estilos de la página web.
- Sb-admin: en este archivo se incluyen los estilos de las tablas.
- cssDateTimePicker: recoge el código necesario para crear el calendario con las fechas y horas.
- Font awesome: contiene una fuente de iconos mediante la que se pueden insertan iconos y símbolos en lugar de imágenes, con lo que la página es más ligera.

4.4. JavaScript.

Durante el desarrollo del trabajo hemos utilizado código JavaScript para crear páginas web dinámicas.

Los archivos de JavaScript se guardan en la carpeta Scripts.

Al igual que las hojas de estilos css los archivos de js se instancian al principio de la vista _Layout.cshtml que corresponde a la página principal.

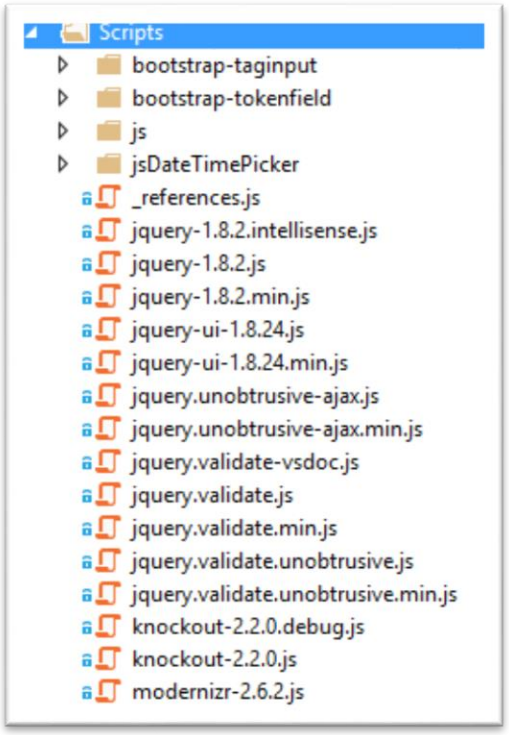


Figura 13. Carpeta con los Scripts.

Durante el desarrollo de la aplicación hemos utilizado varias veces código JavaScript para dinamizar la página web.

Tags

Para la creación de los tags se ha utilizado JavaScript. Se escriben las palabras clave que se pretenden guardar, y una vez introducidas se pulsa el botón “Actualizar Tags”

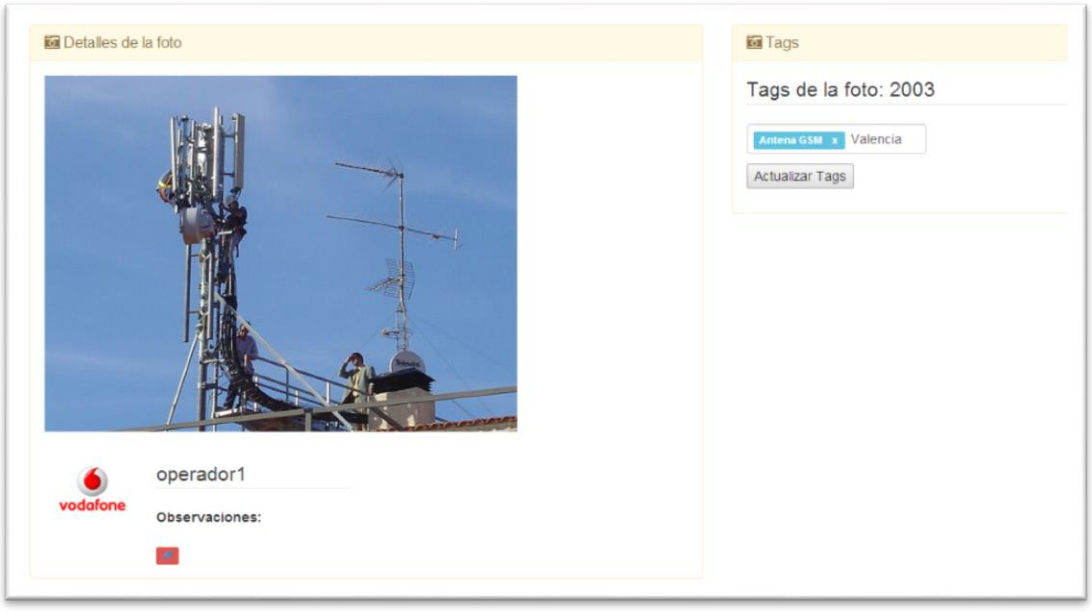


Figura 14. Creación de tags en las fotos.

A su vez estos tags nos servirán de filtro en la página principal:

Figura 15. Filtros de la vista principal del programa, en la que están los tags.

Calendario fecha-hora.

Se ha utilizado código js para crear un calendario que guardará la fecha y hora de la creación/modificación del expediente (rol empresa y operador).

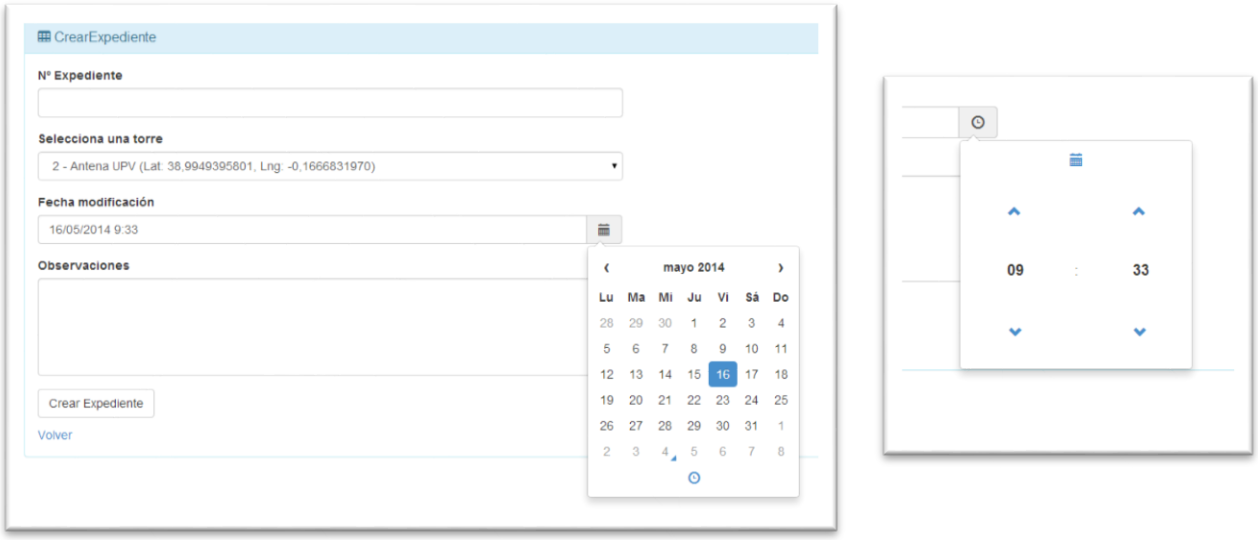


Figura 16. Detalle del calendario de la aplicación, fecha y hora.

Sliders con Responsive Design.

Para que las fotos vayan pasando consecutivamente sin necesidad de que el usuario pase una a una se ha utilizado el correspondiente código js para crear un slider. Sin embargo, además el slider se ha codificado de manera que cumpla Responsive Design y se adapte al tamaño de la pantalla.

Figura 17. Vista de las fotos en un slider a pantalla completa y desde un dispositivo móvil.

4.5. Mailing.

Cuando un usuario se registra en la aplicación se envían automáticamente dos mails a diferentes usuarios.

El primer e-mail se envía al administrador de la aplicación web a modo de aviso. De esta manera éste puede comprobar si el usuario realmente ha utilizado la aplicación para introducir contenido correcto y adecuado.

Figura 18. E-mail que recibe el administrador de la aplicación cuando se registra un nuevo usuario.

El segundo e-mail lo recibe el usuario que se ha registrado, dándole la bienvenida a la aplicación.

Figura 19. E-mail de bienvenida que recibe el usuario nuevo.

Por último, también recibiremos un e-mail cuando un usuario quiera enviarnos algún comentario desde la pestaña de contacta.

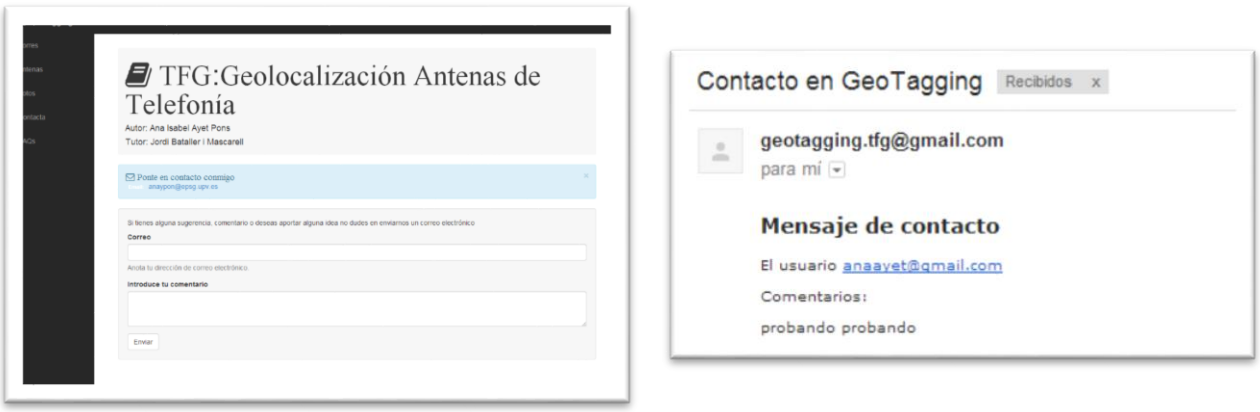


Figura 20. E-mail que recibe el administrador del formulario de contacta.

4.6. Fotos/ Gestión de archivos.

La aplicación creada pretende ser dinámica y visual, es por esto que se ha dado importancia a que el usuario pudiera subir fotos y visualizarlas en cualquier dispositivo.

Además, se ha logrado poder subir varias fotos a la vez para que el proceso sea mucho más rápido.

En la aplicación pincharemos el botón elegir archivos y seleccionaremos los archivos que queramos todos a la vez. En ese caso todas las fotos llevarán en el campo “Observaciones” la misma descripción. Como hemos comentado, este proceso se puede realizar igualmente desde un móvil o desde un ordenador.

Una vez seleccionado los archivos pincharemos subir fotos:

Figura 21. Vista en la que podemos ver que hemos seleccionado 9 archivos a subir.

Una vez subidas las fotos ver que durante el desarrollo de la aplicación se ha tenido en cuenta que puede haber distintos tamaños de fotos y que para ello necesitamos adaptar las fotos.

Por último, comentar que las fotos se guardan en el servidor con un guid para que no se dupliquen.

5. Implementación.

En este capítulo se trata de recoger los problemas que han surgido durante la realización del trabajo final de grado así como se han solucionado.

A su vez, también describiremos los pasos que hay que seguir una vez terminada la programación para poder subir la aplicación a Internet y que sea visible para los usuarios.

5.1. Problemas durante la realización de la aplicación.

A continuación comentaremos algunos de los problemas que han surgido durante la realización de la página web y cómo los hemos solucionado:

- Implementación de los mapas de Google.

En un principio se eligió el proyecto "Googlemap control for Asp.Net MVC" de la página web www.jmelosegui.com/map como referencia para crear los mapas de la aplicación. En esta página explicaba el código que debíamos colocar en cada parte del proyecto, vista, controlador, etc…

Sin embargo, los mapas no se cargaban bien (había que clicar dos veces sobre el mapa), eran lentos y las opciones de zoom tampoco funcionaban.

Se optó por ir directamente al código de google [10], y se eligió la versión 3 del API de JavaScript de Google Maps.

El código utilizado se puede encontrar en la vista que empiezan como Geo…, tanto en la carpeta Torre como en la carpeta Antena.

- Las "infowindow" no funcionaban.

Cuando se desplegaba más de una ventana en el mapa, una después de otra, siempre se tenía la misma información, la del último marcador, independientemente de la ventana que desplegásemos.

Utilizando puntos de interrupción se descubrió que la información estaba dentro de un bucle for, así pues, se guardaba únicamente la última información y el resto al pasar por el bucle for se perdía.

```
var marker = new google.maps.Marker({
        position: coords,
        map: map,
        title: "Your current location!"
});

@foreach (var marker in Model)
{
<text>
    var point = new google.maps.LatLng(@marker.Latitude, @marker.Longitude);
    var logo = "/Images/Torre.png";
    var marker = new google.maps.Marker({
        position: point,
        map: map,
        icon: logo,
        title: '@marker.SiteName',
        html: '@marker.InfoWindow'
    });

    google.maps.event.addListener(marker, "click", function () {
        infoWindow.setContent(this.html);
        infoWindow.open(googleMap, this);
    });
</text>
}
```

Figura 22. Código de la función "marker".

En la solución se ha creado una función por separado que tiene como parámetro un “marker” de esa manera cada infowindow tiene su marker y no el último como anteriormente.

- Solapamiento de las antenas que están en una misma torre.

Cuando una torre tiene más de una antena tenemos el inconveniente de que se solapan los logos. Ante esto se ha programado mediante JavaScript y según la API de JavaScript de Google Maps y [8], de manera que al pinchar en el logo de la torre se desplegasen todas las antenas de una misma torre.

5.2. “How to?”

Una vez se ha terminada la programación el siguiente paso es “subir” la aplicación a un hosting de manera que esté visible en Internet.

5.2.1. Alojamiento de la aplicación web.

Para ello lo primero que se ha hecho es buscar un “hosting” que cumpla las características que necesitamos:

1. Se ha buscado un servidor Windows.
2. Además se necesitaba un hosting ASP.NET 4.5.
3. El hosting debe tener base de datos SQL.

En nuestro caso nos decantamos por una página web que ofrecía alojamiento gratuito: www.somee.com que incluye, además, dominio gratuito siempre que cumplamos unas características determinadas.

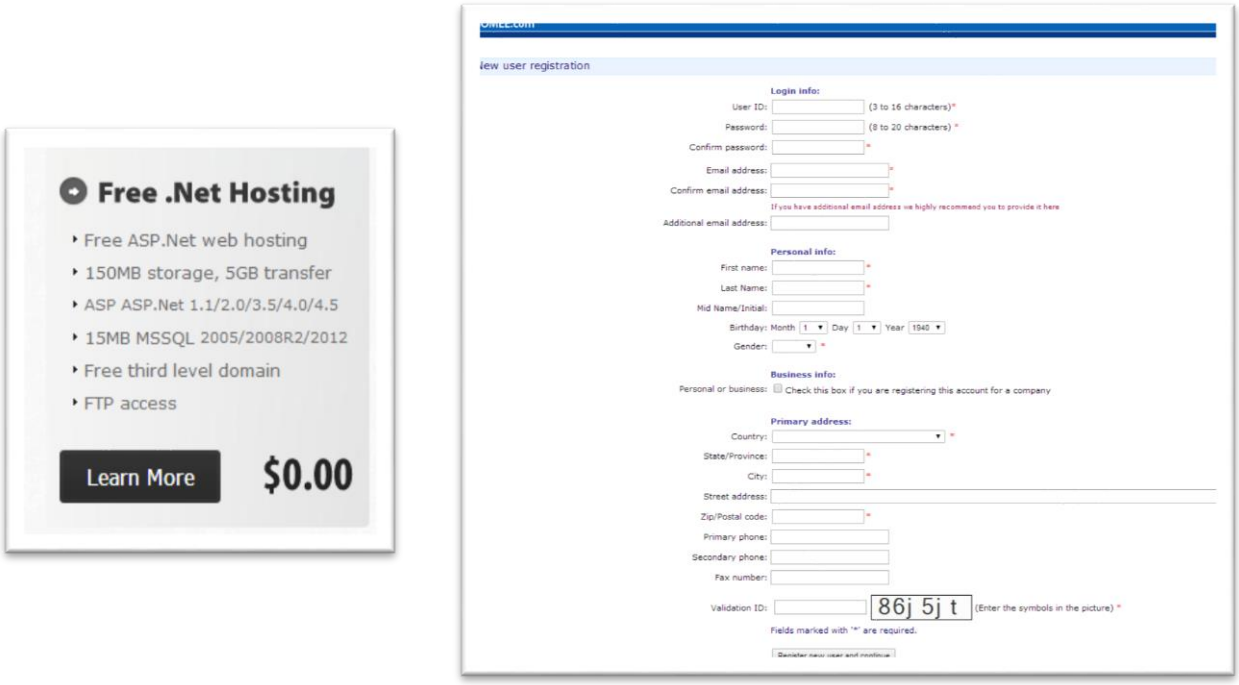


Figura 23. Registro en el hosting.

Tras darnos de alta como usuario debemos buscar una aplicación FTP para la transferencia de los archivos. A la hora de elegir la herramienta de transferencia de archivos hemos optado por WinSCP.

5.2.2. Aplicación para la transferencia de archivos.

WinSCP es una aplicación de Software Libre. Es un cliente SFTP gráfico para Windows que emplea SSH. Su función principal es facilitar la transferencia segura de archivos entre dos sistemas informáticos, el local y uno remoto que ofrezca servicios SSH.

Algunas características:

- Interfaz gráfica (GUI).
- Está disponible en castellano.
- Se integra con Windows.
- Editor de texto integrado.
- Soporte de autenticación mediante contraseñas SSH, método keyboard-interactive, clave pública y Kerberos (GSS).

WinSCP permite efectuar las operaciones básicas con archivos, tales como descargas y subidas. También es posible renombrar archivos y directorios, crear nuevos directorios, modificar las propiedades de archivos y carpetas, y crear enlaces simbólicos y accesos directos.

Es fácil de usar para aquellos usuarios que no usan normalmente la consola de cualquier terminal Linux, desde Windows. Al punto de copiar, mover o eliminar archivos sin problema y rapidez.

WinSCP permite conectarse a un servidor SSH (Secure Shell) empleando el protocolo SFTP (SSH File Transfer Protocol) o el servicio SCP (Secure Copy Protocol).

5.2.3. Conexión de la base de datos.

El hosting ya te proporciona los datos de conexión, de manera que, desde nuestro equipo con esos datos utilizando el Management Studio podemos acceder al servidor de la base de datos y crear la nuestra propia.

En la aplicación en el fichero “web.config” debemos cambiar la cadena de conexión para que apunte a este servidor.

```
<connectionStrings>
  <!--PRODUCCION     -->
  <!--<add name="DefaultConnection" connectionString="Server=geotagging.mssql.somee.com;Database=geotagging;User Id
  <add name="geotaggingEntities" connectionString="metadata=res://*/Models.DataModel.csdl|res://*/Models.DataModel.

  <!--DESARROLLO-->
  <add name="DefaultConnection" connectionString="Server=localhost\sqlexpress;Database=geotagging;User Id=geotag;Pa
  <add name="geotaggingEntities" connectionString="metadata=res://*/Models.DataModel.csdl|res://*/Models.DataModel.

</connectionStrings>
```

Figura 24. Cadena de conexión a cambiar en el archivo “web.config”

6. Manual de usuario

Todo proyecto debe tener una guía técnica para poder dar asistencia a los usuarios.

A continuación, pasaremos a explicar cómo funciona nuestra aplicación.

6.1. Requisitos previos.

En este apartado detallaremos los requisitos previos necesarios en caso de que un usuario quiera utilizar la página web.

¿Qué necesitamos saber antes de comenzar?

Esta aplicación está pensada para ser utilizada por todo tipo de personas, no necesariamente se debe tener conocimientos técnicos sobre telefonía móvil.

Se necesita un dispositivo electrónico con conexión a Internet y que además que tenga activada la opción compartir ubicación en el menú ajustes para que se puedan visualizar los mapas correctamente.

6.1.1. Perfiles.

Nuestra aplicación tiene definidos tres perfiles, independientemente de la vista de cualquier usuario no registrado a la página web:

1. Usuario.
2. Operador de telefonía móvil.
3. Empresa instaladora.

Usuario: Cuando un visitante decide aportar datos a la web se registra como usuario y puede crear torres, posicionar antenas y subir fotos de estas. Es el rol más importante ya que la aplicación prácticamente se alimentará de sus entradas.

Operador de telefonía móvil: Se considera operador de telefonía móvil cualquier compañía con red propia que modifica o amplía su red.

Empresa instaladora: Se considera todas aquellas empresas que instalan las torres y antenas en una estación base. Pueden trabajar para un solo operador o para varios.

6.1.2. Web de consulta.

El principal objetivo al crear la aplicación era conseguir una web de ayuda al usuario en la que pudiera obtener información de las coberturas móviles. Así pues, se trata de una aplicación web de consulta en su mayor parte.

(Para saber cómo utilizar la web de consulta ver apartado 6.2)

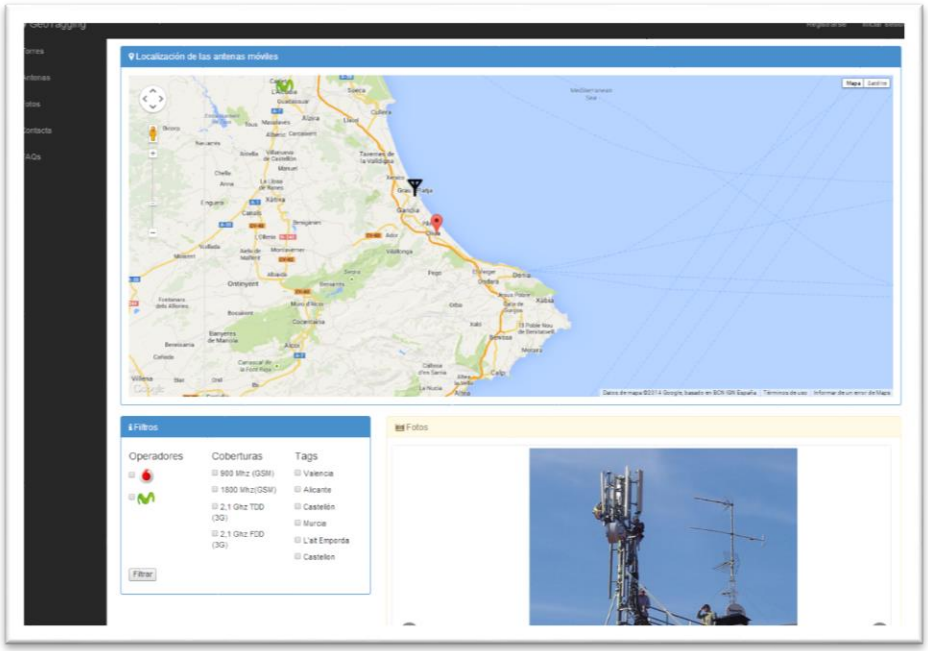


Figura 25. Vista principal de la aplicación web.

6.1.3. Registro de un usuario, empresa u operador.

En caso de que el usuario quiera subir datos de una antena, fotos, etc… y no sólo realizar consultas deberá registrarse en la página web, seleccionando *"Registrarse"* del menú de la izquierda de la pantalla. Una vez dentro rellenamos los campos teniendo en cuenta el tipo de usuario y pulsamos el botón *"Registrarse"*.

Figura 26. Registro de un usuario nuevo.

Cuando un usuario se da de alta en la aplicación automáticamente recibe un mail de bienvenida.

Figura 27. E-mail que recibe el usuario después de registrarse.

6.1.4. Iniciar sesión.

En caso que no sea la primera vez que se accede a la web y puesto que ya estamos registrados se accederá tal como se muestra en la siguiente pantalla.

Figura 28. Formulario de registro.

¿Cómo se guarda la contraseña en la bbdd?

La contraseña que el usuario registrado introduce se encripta y queda guardada en la tabla webpages_Membership de nuestra bbdd geotagging de la siguiente manera:

Resultados | Mensajes

	CreateDate	Confir...	IsC...	Last...	P...	Password	PasswordChangedDate	PasswordS
1	2013-06-30 19:31:24.140	NULL	1	NULL	0	AGapFU/bBVrBM8PnyXbWNzhsQhCcNOSpx8ct4tM9obfEbRlNrFhjHFd4qS...	2013-06-30 19:31:24.140	
2	2013-07-18 21:35:51.040	NULL	1	NULL	0	AlvY9lLLNL4puCl9/jvQpYb/olLtC9O4fAqQnJPx3Zmz0BR0+/3xX50lyPrV+...	2013-07-18 21:35:51.040	
3	2013-07-18 21:41:36.093	NULL	1	NULL	0	AHy9OSCMeG5LjO9KrWskfgKkcTof3gLPmaHozDpDBYMhOMlpVo2LNhz...	2013-07-18 21:41:36.093	
4	2013-07-18 21:42:03.340	NULL	1	NULL	0	AJlbG4MvOcWvuGLhhODMtzpNZ6alh2HXvUhSY0L86NoXaWbugkEw2U...	2013-07-18 21:42:03.340	
5	2013-07-18 21:42:33.007	NULL	1	NULL	0	ABVeSlB8cC+n7GlrBEfAgAOEzZe4TifinjMPRjZ4zT6TCnXnhQPn3ntURA...	2013-07-18 21:42:33.007	
6	2013-07-18 21:43:12.920	NULL	1	NULL	0	APFw1uS6Bv7fhWLbAwPGZr4jZ+2ZLSC1oo0PO1mr2+4kAZ58PekfWQp...	2013-07-18 21:43:12.920	
7	2013-10-01 21:18:42.123	NULL	1	NULL	0	AMwFBqLe18xNFMJKrqeuYUvfT9wFHofpBUnkDwzzyjS0lKwYu+zeyaRgRj...	2013-10-01 21:18:42.123	
8	2013-10-27 15:46:04.123	NULL	1	NULL	0	AEHw8xbBCHdWeJKGFMbre4PikYlem/Vkq3tqTL5okxNZ4hihwNWaBpN6...	2013-10-27 15:46:04.123	
9	2013-10-27 16:13:04.617	NULL	1	NULL	0	APa/86fleuVkYEOyPpK6qaMuj3v0ehKx+gG5aoN8vGmyZV9ujm1KvMqZ8...	2013-10-27 16:13:04.617	
10	2013-10-27 16:15:59.747	NULL	1	NULL	0	AHBRn8kuzQ6CylrPJLD1ksVzEXWUsakDnxUI74OXNs+MsjD70Nj86n24R...	2013-10-27 16:15:59.747	
11	2013-10-29 12:46:56.093	NULL	1	NULL	0	AAWKwu5DHR6cezQx0C+FaaozpYxw6Pn7swouo8pfBMQqaFo7LNfXg2jH...	2013-10-29 12:46:56.093	

Figura 29. Vista de la tabla de la bbdd donde se guardan las contraseñas de los usuarios.

De esta manera se guarda la confidencialidad del usuario.

6.1.5. Cerrar sesión.

Para cerrar sesión clicamos sobre el usuario, se abrirá la pestaña de cerrar sesión y la pinchamos. De esta forma el usuario estará fuera de la sesión y podrá ver de nuevo las pantallas de los usuarios no registrados.

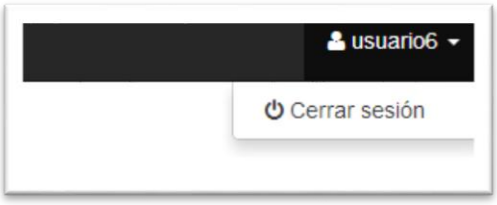


Figura 30: Cerrar sesión.

6.1.6. Contactar con el administrador.

Si se desea contactar con el administrador de la página web pincharemos “Contacta”:

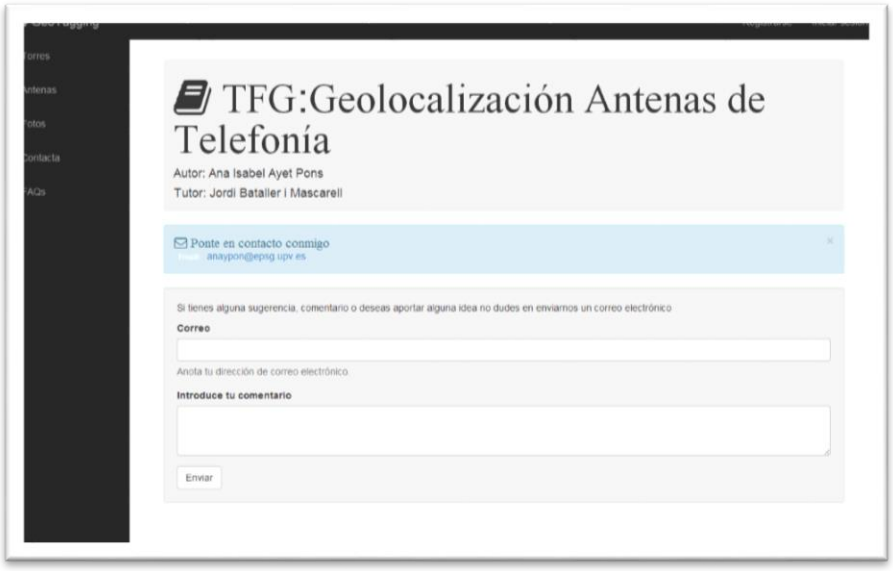


Figura 31. Formulario de contacto.

Anotamos nuestra duda o sugerencia y automáticamente la aplicación envía al administrador un e-mail:

Figura 32. E-mail recibido cuando se rellena algún comentario.

6.1.7. FAQs (Frequently Asked Questions).

FAQ (Frequently Asked Questions) se refiere a una lista de preguntas y respuestas que surgen frecuentemente dentro de un determinado contexto y para un tema en particular, en nuestro caso el funcionamiento de la aplicación.

Hemos creado una pequeña lista de las preguntas y respuestas que creemos que puede ayudar más a un usuario.

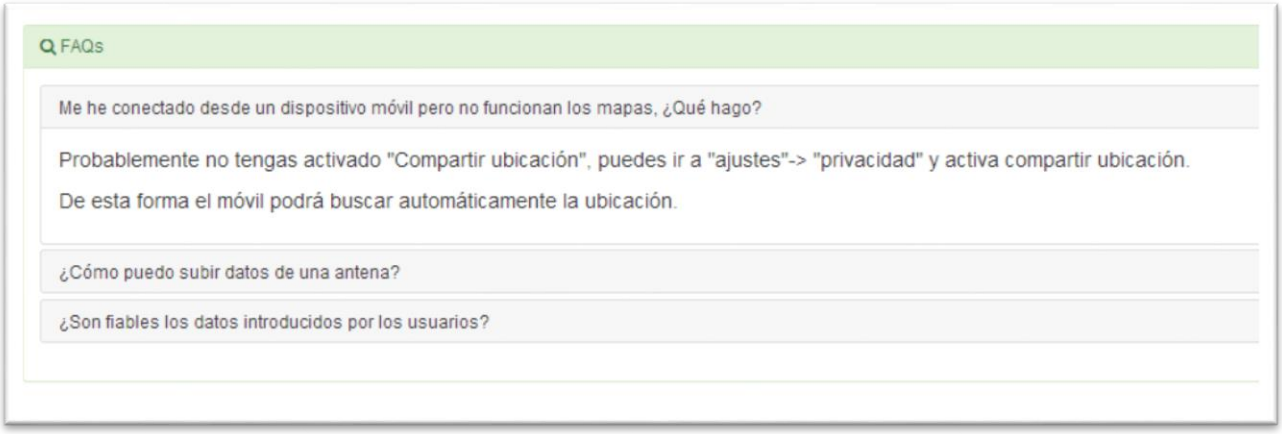


Figura 33. FAQs.

6.2. Usuario no registrado.

Consiste en una serie de pantallas que serán exclusivamente de consulta. En ellas podremos conocer dónde están ubicadas las estaciones móviles, de qué operador son, qué tipo de cobertura tienen, hacer filtros y ver fotos de las antenas.

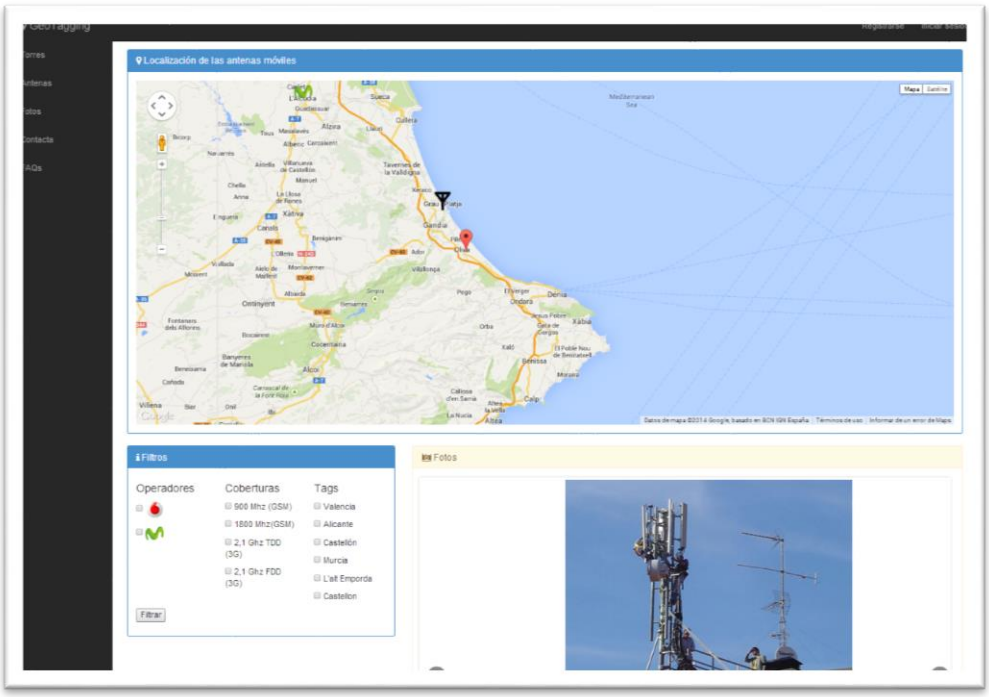


Figura 34. Vista de la página web de un usuario de consulta.

La página web se estructura de la siguiente manera:

- En la vista principal tenemos un mapa donde se han ido ubicando las torres y antenas dadas de alta. Si en una torre hay más de una antena el logotipo que se mostrará es el de la torre, pudiéndose seleccionar y se ampliarán todas las antenas de esa torre. Si la torre sólo tiene una antena se mostrará el logotipo del operador.

Figura 35. Detalle de una torre con varias antenas.

En el panel filtros podemos seleccionar determinadas antenas según nos convenga. Para ello marcamos la selección y apretamos el botón "*Filtrar*".

En la vista principal tenemos también fotos de las antenas o torres.

- Tal como hemos utilizado anteriormente arriba a la derecha tenemos los enlaces para registrarnos e iniciar/cerrar sesión.

En la parte lateral izquierda puedes ampliar información pinchando en alguno de los enlaces que vamos a ver a continuación.

6.2.1. Pestaña "Geotagging".

Es el enlace principal de la página. Siempre que queramos volver a la vista principal podemos pulsarlo.

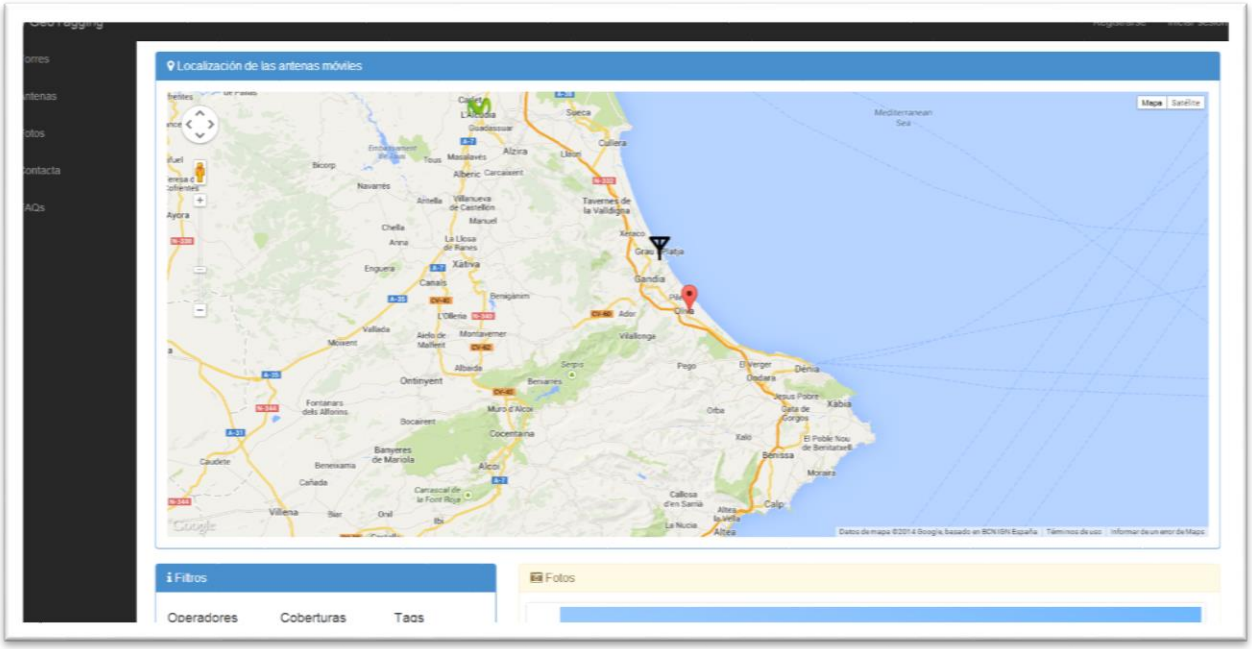


Figura 36. Vista principal de la aplicación.

6.2.2. Pestaña “Torres”.

Si pinchamos Torres tendremos una vista con todas las torres que existen posicionándolas también en el mapa.

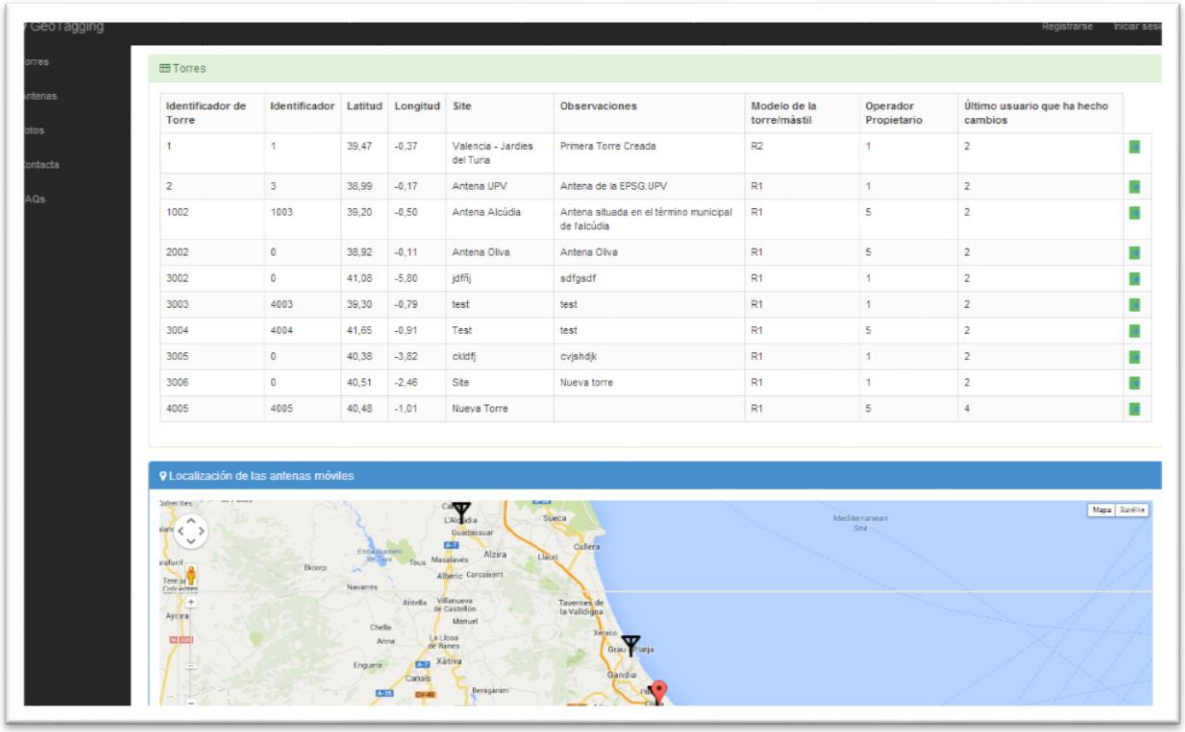


Figura 37. Vista de la pestaña Torres.

A su vez, podemos obtener detalles de una torre en concreto pulsando el botón verde del final de la línea: características, antenas que pertenecen a la torre, tags de la torre y mapa de posición.

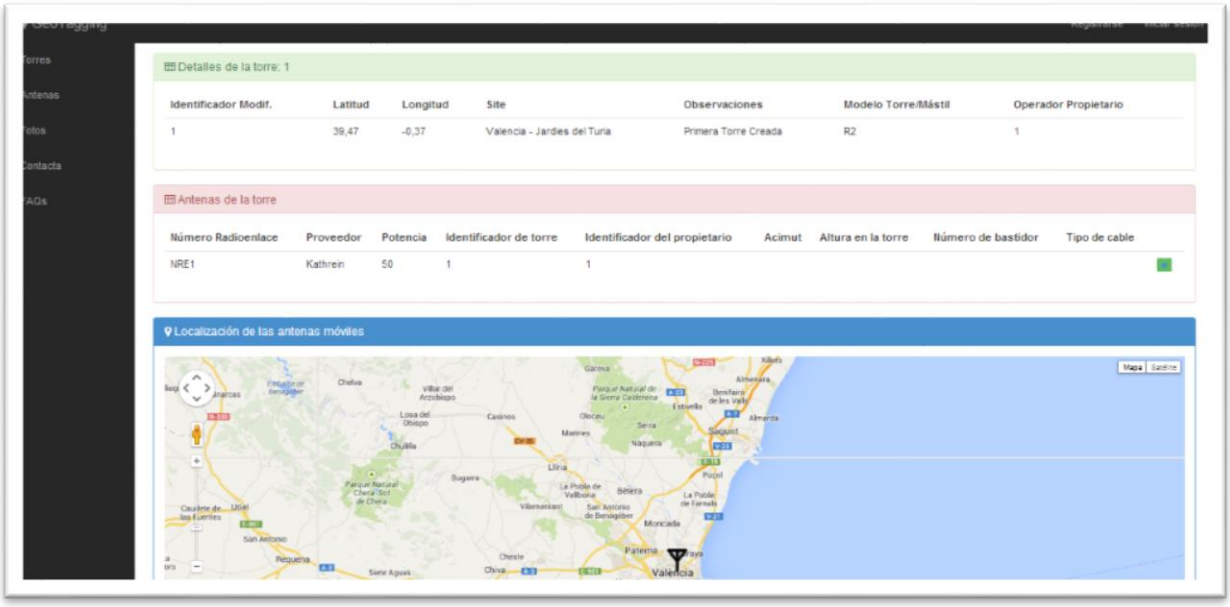


Figura 38. Vista si pulsamos en una torre.

6.2.3. Pestaña “Antenas”.

Si pinchamos la pestaña de Antenas tendremos la vista de todas las antenas con más detalle:

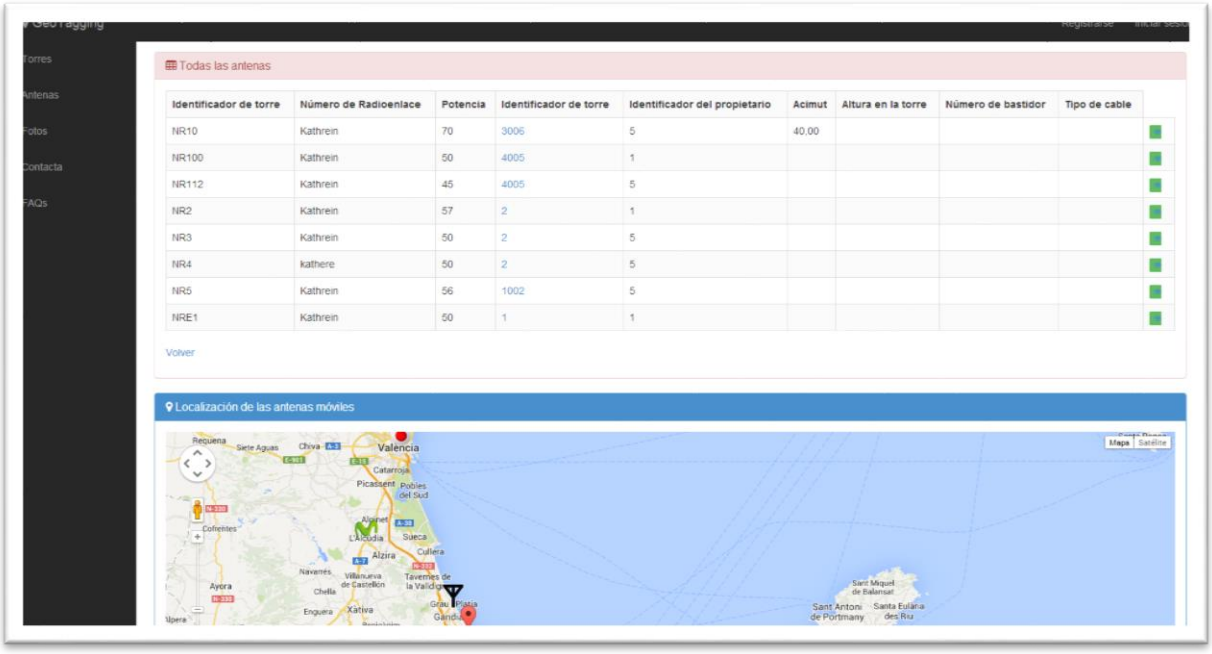


Figura 39. Vista de la pestaña Antenas.

Podemos ampliar la información de una antena pulsando en el botón verde. Obtendremos los parámetros de la antena, la localización de la antena en el mapa, las fotos de esa antena, coberturas y tags.

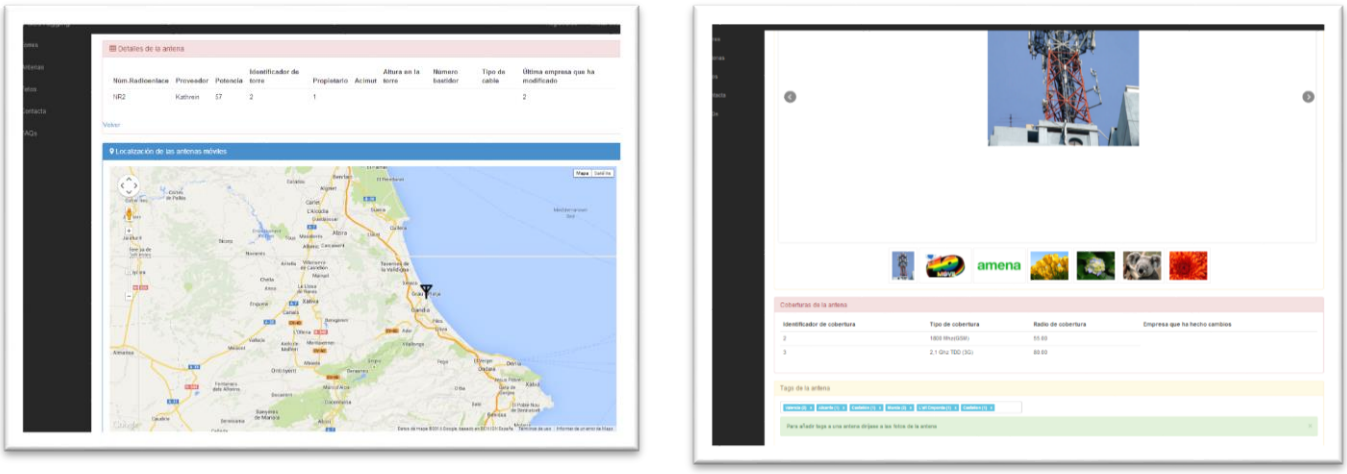


Figura 39. Vistas que obtenemos sobre una antena en concreto.

6.2.4. Pestaña “Fotos”.

En la pestaña fotos se pueden ver las fotos de cada antena o torre que los usuarios registrados, operadores o empresas han subido. Si pinchamos en cada una de estas fotos obtendremos toda la información.

Figura 40. Vista de la pestaña Fotos.

6.3. Usuario registrado.

En caso de que un usuario quiera aportar datos a la aplicación deberá registrarse. Una vez registrado accederá a la página principal desde la que puede crear/borrar torres, antenas, subir fotos, taggearlas, etc…

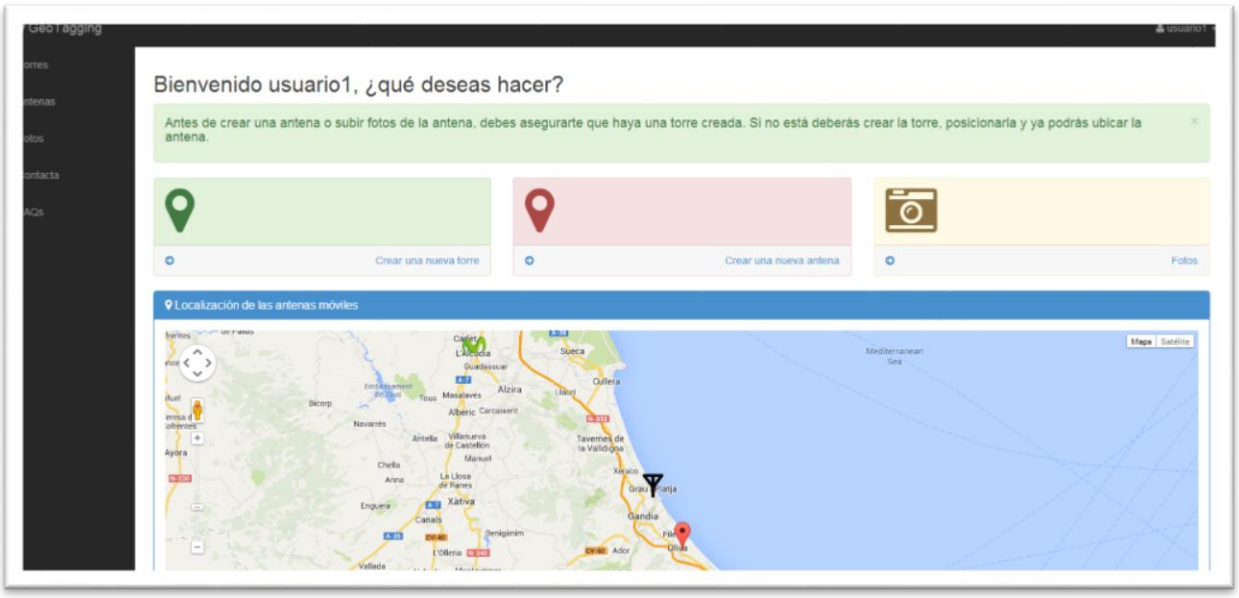


Figura 41. Vista principal de un usuario.

En la parte izquierda, como antes, tenemos el menú lateral de “torres, antenas, fotos” pero en este caso cuando pinchemos serán las torres, antenas o fotos de ese usuario en concreto.

6.3.1. Pestaña “Torres”.

Si pinchamos “Torres” podemos ver en la pantalla las torres de ese usuario ubicadas, una tabla con todas las torres, en las que se podrá ver más datos pero sólo de lectura y las torres creadas por el usuario en las que podrá ver más datos pinchando en el botón verde o bien podrá borrar la torre con el botón rojo.

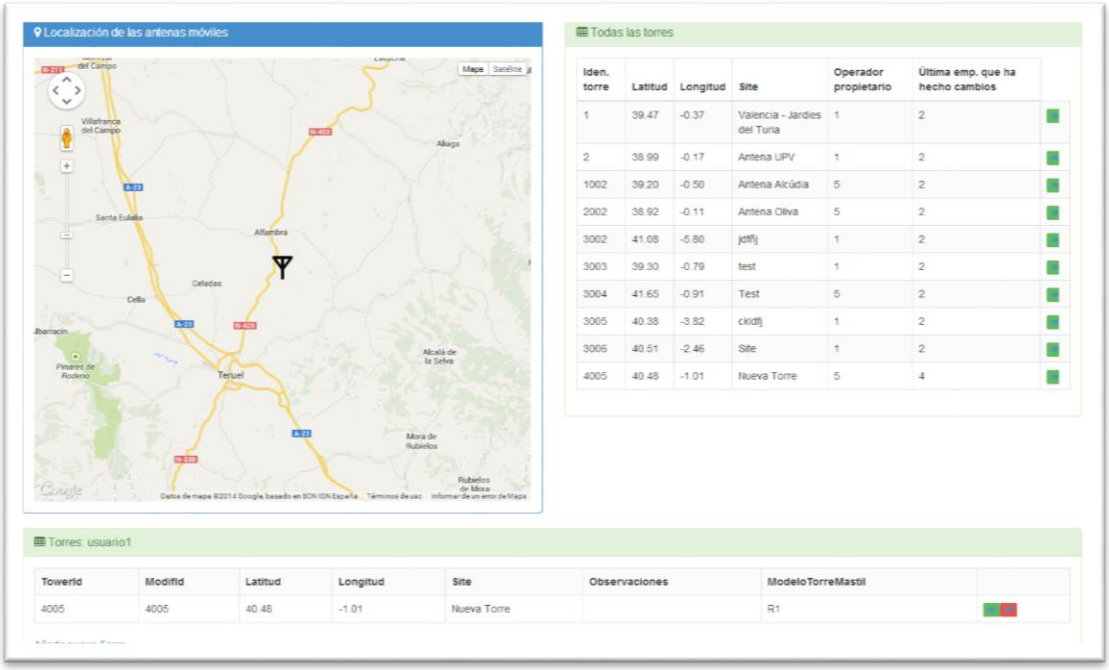


Figura 42. Vista de la pestaña Torres.

6.3.2. Pestaña “Antenas”.

Si pinchamos en “Antenas” obtenemos un listado de todas las antenas ubicadas. A su vez, si queremos ampliar información pincharemos el botón verde.

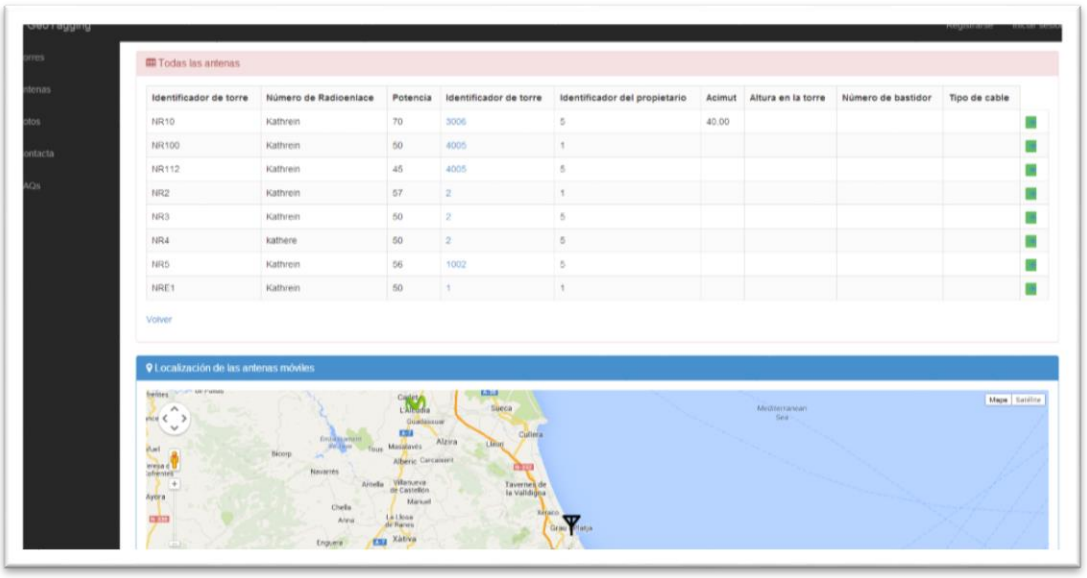


Figura 43. Vista de la pestaña Antenas.

El enlace “Contacta” y “FAQs” son iguales en todos los perfiles (ver apartados 6.1.6 y 6.1.7).

6.3.3. Pantalla principal.

En la pantalla principal tenemos los enlaces para poder crear torres, antenas o ver las fotos.

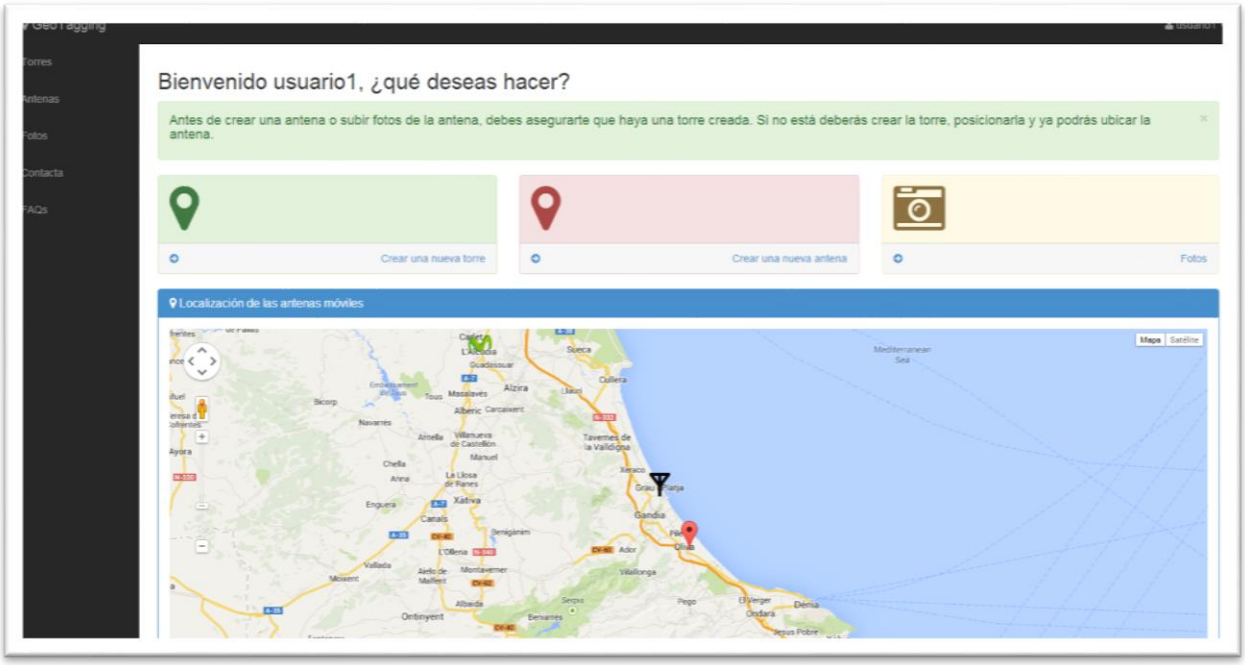


Figura 44. Vista principal cuando un usuario inicia sesión.

El procedimiento es el siguiente:

1. Primero comprobar si ya tenemos una torre en el sitio que queremos introducir datos. Si no se tiene el primer paso es crear la torre.
2. En caso que ya tengamos una torre ubicada y solo sea crear más antenas iríamos al segundo enlace “Crear una nueva antena”.
3. Si queremos ver las fotos que hemos subido pincharemos en el enlace de “Fotos”.

Crear una nueva torre

Lo primero que debemos hacer cuando creamos una torre es ubicarla. Nos situaremos en la localización adecuada mediante el zoom y arrastrando con el ratón y una vez en el sitio adecuado clicamos el botón izquierdo para geolocalizar la torre.

Rellenamos los campos que nos piden y pinchamos en el botón “Crear”.

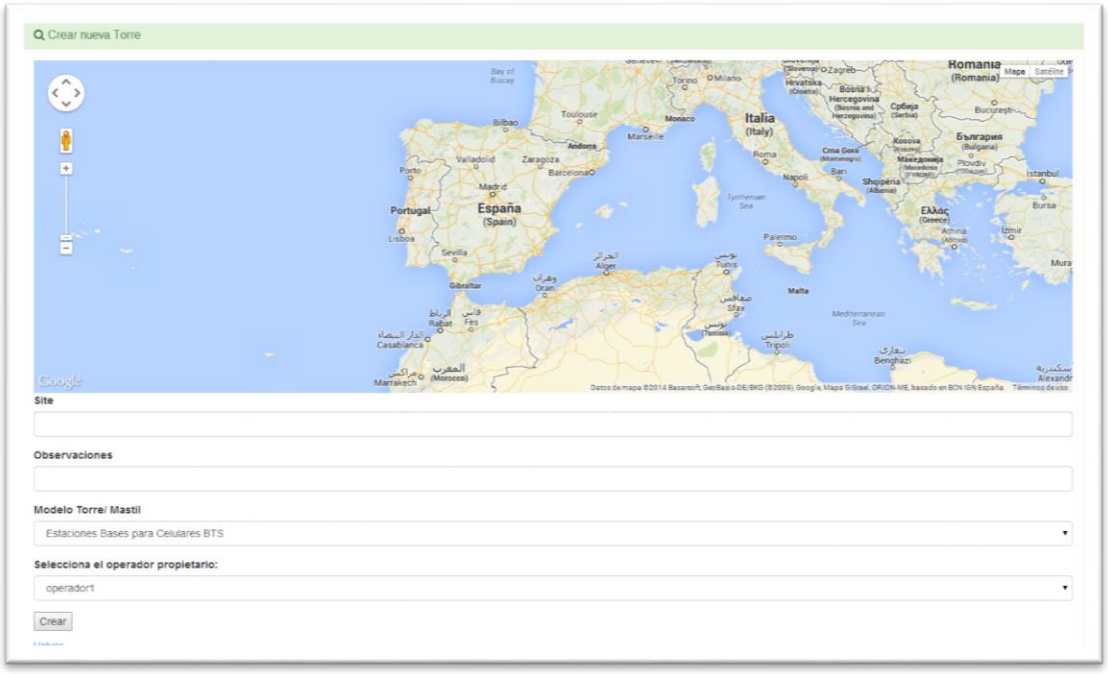


Figura 45. Pantalla de crear una torre.

Crear una nueva antena

Una vez tengamos la torre ubicada pasaremos a dar de alta antenas:

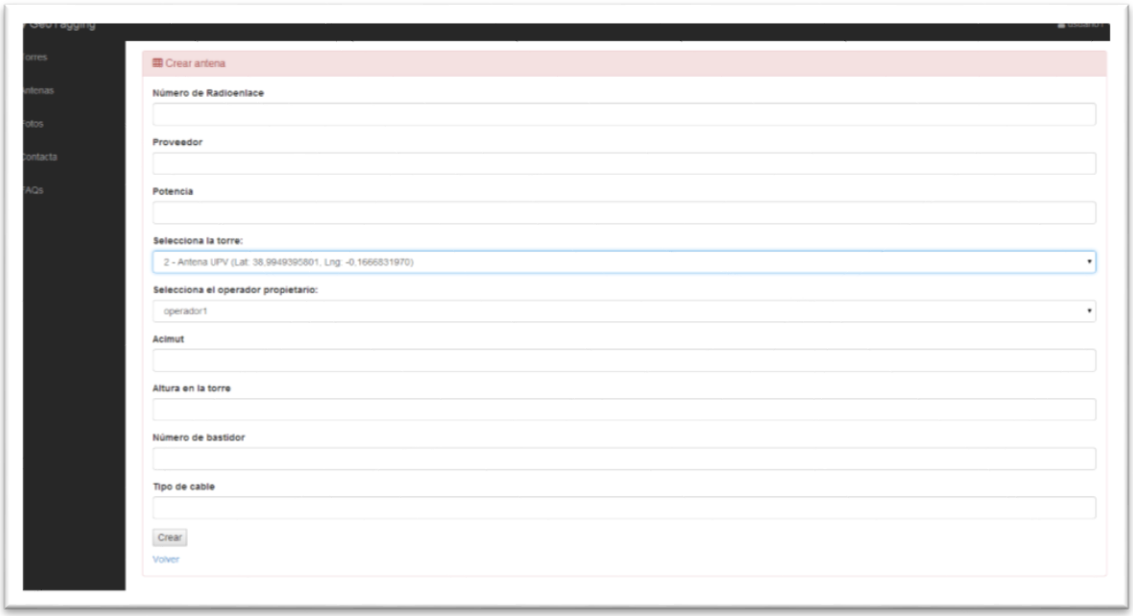


Figura 46. Pantalla para crear una antena.

Tanto en la torre como en el operador tendremos un desplegable para seleccionar la ubicación y operador de la antena.

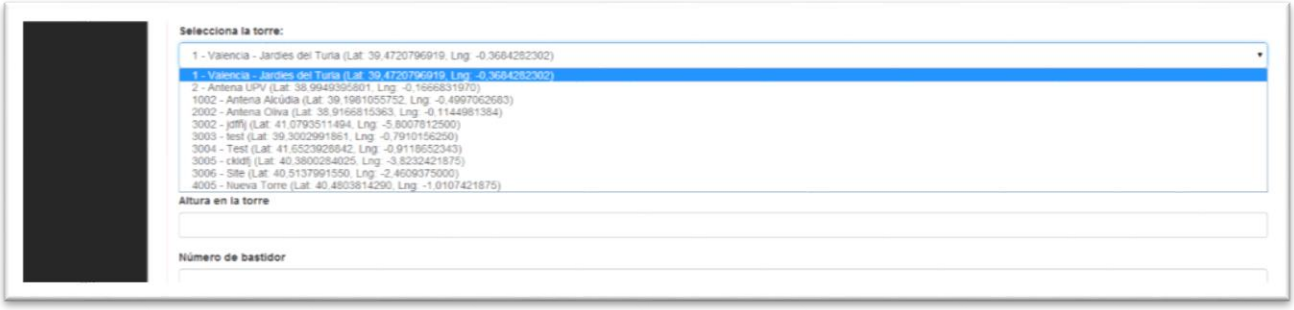


Figura 47. Desplegable para elegir torre donde está la antena.

Fotos

Si clicamos el enlace de “Fotos” tendremos todas las fotos que se han subido a la aplicación, sin embargo cuando clicamos sobre una foto sólo se podrán borrar aquellas que ha subido el usuario.

Figura 48. Foto que se puede borrar por el usuario que la ha subido.

El resto de fotos serán sólo de consulta.

6.4. Operador de telefonía móvil o empresa instaladora

En los perfiles de operador y de una empresa aparece el concepto expediente. En un expediente podemos guardar más información respecto a la torre y las antenas y, sobre todo, guardar información de fechas de modificaciones en esa torre y cuándo las han realizado.

Así pues el primer cambio visible será al abrir la sesión con estos perfiles.

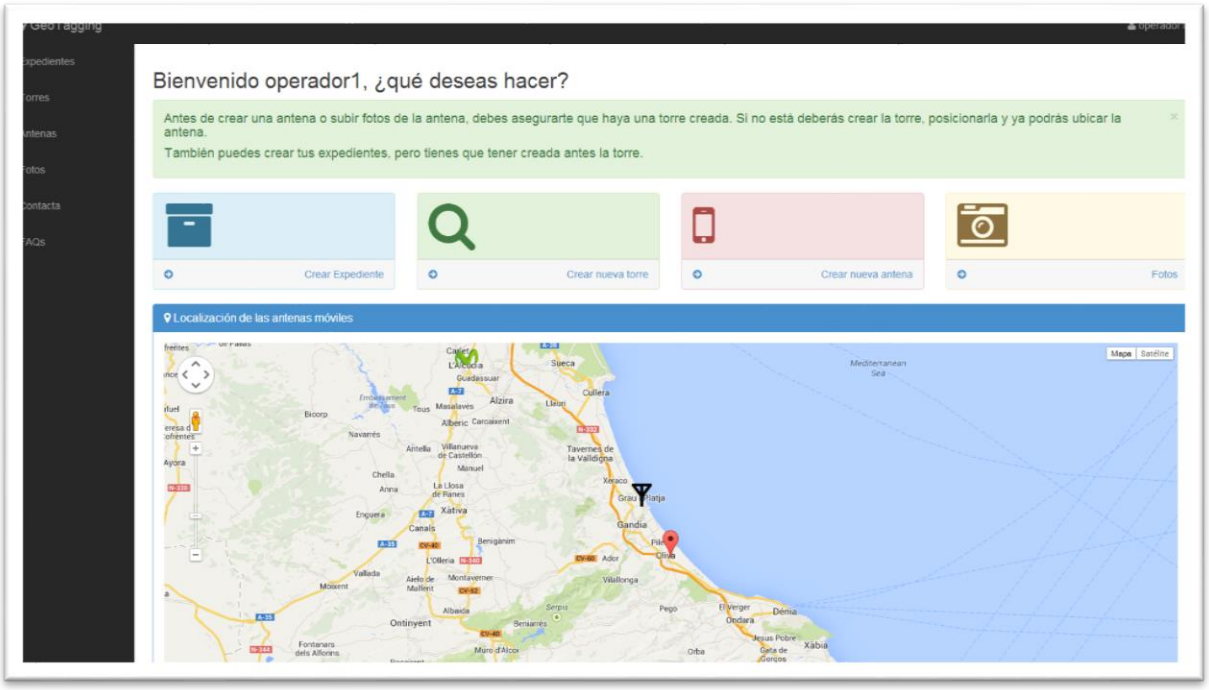


Figura 49. Vista de la pantalla al iniciar sesión un operador o una empresa instaladora.

Una vez iniciada la sesión tenemos una primera vista total posicionando las antenas

6.4.1. Pestaña “Expedientes”.

La primera diferencia que podemos encontrar a la hora de abrir la aplicación como una empresa u operador es el enlace a “Expedientes”. Mediante el menú “Expedientes” estos pueden llevar un seguimiento de las modificaciones o cambios que se realizan en una determinada torre.

Para crear un nuevo expediente debemos tener con anterioridad una torre para poder asociarla cuando se cree el nuevo expediente.

Así pues el procedimiento a seguir para introducir nuevos datos sería:

- Creamos la torre: definimos sus parámetros y la asociamos.
- Creamos las antenas que estén en esa torre y subimos fotos, si las tenemos.
- Damos de alta un nuevo expediente y asociamos la torre, que ya se habrá añadido al desplegable de torres.

Más tarde, podemos realizar los cambios que vayamos realizando, anotándolo en observaciones y así llevar un seguimiento.

Cada operador podrá ver los expedientes que le correspondan, independientemente de la empresa que haya hecho el trabajo.

Cada empresa instaladora podrá ver sus expedientes, independientemente del operador que sea.

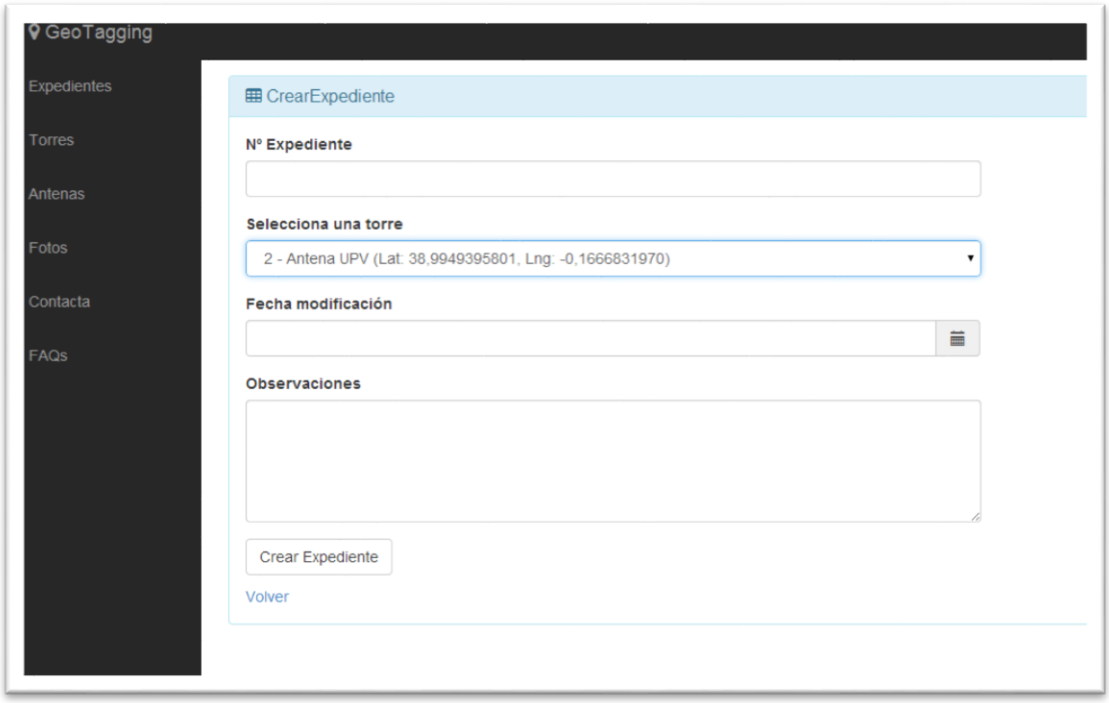


Figura 50. Pantalla principal en la que creamos un expediente.

Si en cambio enlazamos con el menú lateral de “Expedientes” obtendremos una vista de todos expedientes, en los que como operador podrá ver o modificar y como empresa podrá crear expedientes o borrarlos.

6.4.2. Pestaña “Torres”.

Para crear una nueva torre seguir el mismo procedimiento que un usuario no registrado (Ver apartado 6.3.3 Crear nueva torre).

6.4.3. Pestaña “Antenas”.

Si pinchamos en “Antenas” aparecerá en pantalla las antenas del operador.

A su vez estas antenas podremos leerlas, modificarlas o bien borrarlas.

En caso de querer crear una nueva antena ver apartado 6.3.3. Crear nueva antena.

6.4.4. Pestaña: Mis Fotos.

Si se enlaza con el menú fotos podremos ver las fotos, borrar las que son del operador y actualizar los Tags.

7. Conclusiones y líneas futuras.

Este capítulo se constituye de dos apartados.

En primer lugar se expondrán las conclusiones, que pretenden mostrarnos los objetivos planteados al inicio del proyecto que se han conseguido. Seguidamente se explicarán las aportaciones conseguidas tras la realización de este ejercicio final de carrera.

Por otra parte propondremos unas líneas futuras de investigación, opcionales, si se quiere realizar un estudio más completo y amplio.

7.1. Conclusiones.

El principal objetivo de este proyecto ha consistido en desarrollar una herramienta de consulta para los usuarios, con la que puedan comparar diferentes compañías y ver las estaciones base que tienen cerca.

Pero además, podemos concluir nuestro trabajo resumiendo los principales puntos que hemos realizado:

- Se ha conseguido que empresas instaladoras y operadores de telefonía móvil tengan una herramienta común independientemente de quien trabaje con quien.
- Se ha logrado una herramienta sencilla y muy visual.
- Se ha conseguido posicionar las antenas y las torres en un mapa, con independencia del zoom que tenga el mapa.
- Se ha podido crear una aplicación web que funciona en todo tipo de dispositivos independientemente de su tamaño o resolución.
- He aprendido y/o ampliado mis conocimientos en lenguajes de programación, siguiendo una metodología de trabajo que se está aplicando a proyectos grandes y que podré aplicar en mi desarrollo profesional.

En consecuencia, pensamos que esta aplicación web puede resultar útil al sector de telefonía móvil por su adaptabilidad, sencillez y bajo coste de mantenimiento.

7.2. Futuras ampliaciones.

Este proyecto nos proporciona una herramienta que permite localizar antenas de telefonía móvil y consultar datos de operadores y realizar filtros.

Sin embargo, esta aplicación puede ser mejorada. Por este motivo se proponen una serie de líneas futuras de trabajo:

- Hacer que la página web sea mucho más rápida cuando se cargue por completo. Cambiar de hosting de manera que sea mucho más rápido.
- Hacer mapas de color según la cobertura: buena, muy buena o sin cobertura.
- Incluir más datos según vayan necesitando las empresas o lo requieran los operadores.
- Crear botón “cambiar de usuario”.
- Establecer protocolos de seguridad de los datos.
- Ley de protección de datos (LOPD).
- Crear un perfil administrador que pueda modificar y borrar cualquier dato que no sea realista.

8. Bibliografía

LIBROS:

1. Aguilar, José María, *Desarrollo Web con ASP.NET MVC 4*. CampusMVP, 2013.
2. Ceballos Sierra, Francisco Javier, *Microsoft C#: curso de programación.* Editorial Ra-ma, 2011.
3. Debrauwer, Laurent; van der Heyde, Fien, *UML2: iniciación, ejemplos y ejercicios corregidos*. Editorial ENI, 2013.
4. Gabillaud, Jérôme, *SQL Server 2012, transact SQL: Diseño y creación de una base de datos*. Editorial ENI, 2013, cap. 2-4.
5. Orós Cabello, Juan Carlos, *Diseño de páginas Web con XHTML, JavaScript y CSS*. Editorial RA-MA, 2010.
6. Zakas, Nicholas C., *Profesional JavaScript para desarrolladores Web*. Ediciones Anaya Multimedia, 2006

SITIOS WEB:

7. *Mvcmailer 4.5.0*, 2013, https://www.nuget.org/packages/MvcMailer [Consulta: Miércoles, 11 de diciembre de 2013]
8. *Overlapping Marker Spiderfier for Google Maps API v3*, 2014, https://github.com/jawj/OverlappingMarkerSpiderfier [Consulta: Lunes, 17 de febrero de 2014]
9. *Using MVCMailer in asp.net MVC4 application, keep getting error on email send*, 2014, http://stackoverflow.com/questions/18565422/using-mvcmailer-in-asp-net-mvc4-application-keep-getting-error-on-email-send [Consulta: Lunes, 12 de mayo de 2014]
10. *Versión 3 del API de JavaScript de Google Maps*, 2013, https://developers.google.com/maps/documentation/javascript/?hl=es [Consulta: Jueves, 14 de noviembre de 2013]