Universidad Zaragoza

Trabajo Fin de Grado

Sistema Web para la Integración de Estaciones Meteorológicas

Autor

David Enjuanes Gómez

Director

Francisco Javier Fabra Caro

Escuela de Ingeniería y Arquitectura

2017

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. David Enjuanes Gómez,

con nº de DNI 18059732-V en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Sistema Web para la Integración de Estaciones Meteorológicas

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 19 de Abril de 2017

Fdo: David Enjuanes Gómez

Agradecimientos

A los profesores, que me han transmitido un conocimiento más que valioso. Mi especial agradecimiento a Javier, que además de haber estado ahí durante la realización de este TFG me ha transmitido su interés por las tecnologías web.

Agradecer a mi familia todo el esfuerzo realizado, que no ha sido poco, para permitirme llegar hasta aquí.

Sistema Web para la Integración de Estaciones Meteorológicas

Resumen

En la actualidad existen multitud de fabricantes de estaciones meteorológicas. Muchas de ellas tienen la capacidad de conectarse a Internet, de forma que los datos meteorológicos tomados por las mismas quedan accesibles a través de la red de redes. Sin embargo, las plataformas a través de las que estos datos meteorológicos se publican ofrecen unas funcionalidades bastante pobres: cada fabricante dispone de su propia plataforma que solo muestra datos de las estaciones de dicho fabricante, no ofrecen información meteorológica histórica (en muchas ocasiones solo aparece la información meteorológica actual), no ofrecen un API (de forma que los datos meteorológicos no son fácilmente compartibles con otras plataformas terceras), etc.

La plataforma desarrollada en este TFG, denominada *ownmeteo.com*, viene a suplir las carencias descritas en el párrafo anterior. Es decir, *ownmeteo.com* busca: ser una plataforma en la cual se puedan registrar estaciones meteorológicas de diferentes fabricantes; ser una plataforma donde los datos de las estaciones meteorológicas registradas son monitorizados y almacenados periódicamente, de forma que puedan ser consultados en el futuro (es decir, la plataforma ofrece datos meteorológicos históricos) y, finalmente, ser una plataforma donde los datos meteorológicos tomados por las estaciones meteorológicas puedan ser compartidos con aplicaciones terceras a través de un API.

Para lograr estos objetivos se ha desarrollado una plataforma utilizando principalmente el conjunto de tecnologías o *stack* MEAN (MongoDB, Express, AngularJS y Node.js). Los principales componentes de la plataforma son: la base de datos (donde se ha utilizado MongoDB gracias a la flexibilidad que ofrece), el API REST (servidor implementado con Node.js y Express cuyo API es utilizado para la gestión de la plataforma: desde la creación de usuarios hasta el registro y compartición de los datos meteorológicos), la aplicación web (implementada principalmente con AngularJS, se encarga de ofrecer una interfaz amigable a los usuarios de la plataforma), el servidor de vistas (implementado con Node.js y Express se encarga de servir la aplicación web) y el Lector de Estaciones Meteorológicas (encargado de leer periódicamente los datos meteorológicos de las estaciones meteorológicas registradas en la plataforma).

La plataforma ha pasado una fase de validación en la cual se ha utilizado el *framework* Protractor, encargado de ejecutar de forma automática el juego de pruebas diseñado.

Tras la fase de validación, la plataforma fue puesta en producción, de forma que se encuentra accesible a través de la siguiente URL: https://www.ownmeteo.com.

Índice

Índice de Figuras

Índice de Tablas

1. Introducción

En la actualidad, la información es fundamental para hacer que un negocio sea competitivo, eficiente y próspero. De esta norma no escapan las explotaciones agrarias. Como es sabido, el sector agrario es totalmente dependiente de la meteorología, por lo cual, disponer de gran cantidad de información de la misma y de calidad puede ser un factor diferenciador. Este hecho fue la principal motivación para llevar a cabo este proyecto.

En este trabajo se ha desarrollado una aplicación, denominada *ownmeteo.com*, que permite integrar bajo una misma plataforma datos meteorológicos provenientes de estaciones meteorológicas de diferentes fabricantes. Es importante aclarar que la plataforma no solo está enfocada al sector agrario, sino que ha sido pensada para ser utilizada por cualquier interesado en la meteorología.

ownmeteo.com permite acceder a los datos meteorológicos tomados por las estaciones meteorológicas a través de un API REST, de forma que los datos pueden ser utilizados en aplicaciones de terceros (aplicaciones móviles, aplicaciones web, etc.). Además, se ha desarrollado una utilidad que permite a los usuarios de la aplicación insertar los datos de sus estaciones en un sitio web fácilmente.

1.1. Contexto y Motivación

En la actualidad existen multitud de modelos y fabricantes de estaciones meteorológicas. Cada fabricante (y en algunos casos incluso cada modelo) dispone de un sistema propio para publicar los datos de sus estaciones en Internet.

Muy pocos de estos sistemas ofrecen la posibilidad de guardar históricos de los datos meteorológicos. Es decir, solo aportan información de las variables meteorológicas actuales en el momento de la consulta.

En muchas ocasiones, estos sistemas ofrecen solo una interfaz gráfica para acceder a los datos meteorológicos y no una interfaz de programación de aplicaciones (API) estándar. Esto provoca que los datos meteorológicos no sean fácilmente accesibles por aplicaciones terceras. Además, algunos de estos sistemas ofrecen unas prestaciones pobres: altas latencias, baja disponibilidad, etc.

Este proyecto propone una herramienta eficaz e innovadora para la gestión de información meteorológica. Una herramienta que, por ejemplo, permite:

- Registrar los datos meteorológicos de diferentes temporadas de una explotación agraria para poder compararlos con las cosechas obtenidas en dichas temporadas.
- Integrar datos meteorológicos de diferentes estaciones (con diferentes variables meteorológicas medidas) bajo una misma plataforma.
- Registrar datos meteorológicos de diferentes áreas para, posteriormente, poder contrastarlos (por ejemplo; para analizar qué zona es más apropiada para la construcción de una infraestructura, para determinar que parcela agraria es más apropiada para cierto cultivo, etc.).

- Disponer de datos meteorológicos fiables en tiempo real, desde cualquier lugar y dispositivo para que los usuarios de la plataforma puedan actuar en consecuencia de esos datos (por ejemplo; desactivar el sistema de riego si se detecta que ya está lloviendo).

Una herramienta con todas estas características será de gran utilidad para todo aquel interesado en la meteorología: agricultores, administraciones públicas, meteorólogos, aficionados, etc.

1.2. Objetivos

A continuación se enumeran los objetivos del sistema desarrollado:

- Ofrecer una plataforma capaz de integrar información meteorológica proveniente de diferentes modelos y fabricantes de estaciones meteorológicas. Para ello, la plataforma deberá ser extensible, de forma que sea capaz de trabajar con nuevos modelos de estaciones meteorológicas e incluso nuevas variables meteorológicas no valoradas inicialmente.
- Almacenar los datos meteorológicos obtenidos por las estaciones vinculadas periódicamente, de forma que posteriormente se puedan obtener datos meteorológicos históricos.
- Permitir la compartición de datos con otros sistemas. Definir un API estándar accesible por aplicaciones terceras (aplicaciones web, aplicaciones móviles, etc.).
- Permitir a los usuarios gestionar sus estaciones meteorológicas, permitiéndoles añadir o quitar sus estaciones.
- Permitir a los usuarios gestionar la información publicada de sus estaciones (variables meteorológicas, unidades, etc.).

1.3. Organización de la Memoria

En este apartado se describe la organización del documento con el objetivo de facilitar la comprensión del mismo al lector. A continuación se explica brevemente el contenido de cada capítulo de la memoria:

- Capítulo 1, Introducción: introduce brevemente el proyecto, describiendo el contexto actual, la motivación y los objetivos del mismo.
- Capítulo 2, Contexto: contiene un análisis más profundo del contexto en el cual se realiza el proyecto.
- Capítulo 3, Propuesta de Solución: describe la solución propuesta.
- Capítulo 4, Desarrollo: describe el proceso de desarrollo del proyecto.
- Capítulo 5, Validación: explica el proceso de validación llevado a cabo para certificar el correcto funcionamiento de la plataforma desarrollada.
- Capítulo 6, Conclusiones: contiene las conclusiones del autor.
- Capítulo 7, Bibliografía: contiene la bibliografía consultada.

A continuación se explica brevemente el contenido de los Anexos:

- Anexo A: contiene la especificación del API REST desarrollado.
- Anexo B: contiene los mapas de navegación de la aplicación web desarrollada.
- Anexo C: describe las vistas de la aplicación web desarrollada.
- Anexo D: describe el juego de pruebas desarrollado para la validación de la plataforma.

- Anexo E: contiene la especificación de los códigos de error mostrados por la aplicación web desarrollada.
- Anexo F: describe el proceso de puesta en producción de *ownmeteo.com*.

2. Contexto

En esta sección se analiza el contexto en el cual se realiza el proyecto. En primera instancia se analizan los posibles actores interesados en el sistema desarrollado en este proyecto. En segundo lugar se analiza las soluciones actuales similares a la desarrollada en este proyecto.

2.1. Partes Interesadas

A continuación se enumeran las posibles partes interesadas en el uso del sistema desarrollado en este proyecto:

- Agricultores que quieren tener mayor información acerca de la meteorología de sus dominios (pluviometría, temperaturas, humedad, etc.).
- Agencias meteorológicas, para gestionar sus redes de estaciones meteorológicas.
- Agentes que precisen de información meteorológica precisa para la toma de decisiones (por ejemplo; para la construcción de una infraestructura en función de la meteorología de cierta zona).
- Aficionados.

En general, cualquier agente interesado en la meteorología sería un potencial usuario de la plataforma desarrollada en este proyecto.

2.2. Soluciones Actuales

Actualmente, los usuarios que quieran acceder a los datos de sus estaciones meteorológicas a través de Internet dependen principalmente de los sistemas proporcionados por los propios fabricantes de las estaciones meteorológicas. Únicamente se han encontrado dos alternativas a las plataformas de los fabricantes: *WeatherCloud* y *Weather Underground*. A continuación se describen las principales plataformas encontradas:

2.2.1. *Davis WeatherLink Network*

Davis Instruments Corp. [1] es uno de los fabricantes de estaciones meteorológicas profesionales más consolidados del mercado. Su solución para la publicación de los datos meteorológicos de sus estaciones en Internet es la *Davis WeatherLink Network* [2].

Las estaciones de este fabricante (únicamente las capaces de conectarse a Internet) envían los datos meteorológicos (tomados cada minuto) a la *Davis WeatherLink Network*. Una vez los datos son registrados por esta red, son accesibles a través de un navegador web por cualquier usuario interesado. Por ejemplo, a través de la URL http://www.weatherlink.com/user/ralonso es posible acceder a la lectura meteorológica más reciente realizada por una estación meteorológica *Davis Vantage Pro2* situada en la EINA.

Esta plataforma es accesible desde cualquier lugar y desde cualquier dispositivo con acceso a la web. Sin embargo, presenta importantes carencias:

- La plataforma únicamente muestra los datos en tiempo real de la estación, de forma que no se tiene ningún tipo de registro de datos meteorológicos históricos.
- La plataforma no tiene un API funcional con el que se pueda interactuar fácilmente con ella desde aplicaciones terceras. Por lo tanto, para acceder desde una aplicación tercera

a los datos de una estación en esta plataforma es necesario procesar el documento HTML.
- Si bien la información es accesible a través de cualquier navegador web, la experiencia para acceder a la misma desde un dispositivo móvil no es óptima.
- Obviamente, esta plataforma solo interactúa con las estaciones meteorológicas del fabricante.

Pese a ser una herramienta funcional y potente, esta plataforma tiene carencias que son solventadas con el desarrollo de este proyecto.

2.2.2. *Netatmo Connect*

Netatmo S.A. [3] es posiblemente el fabricante de estaciones meteorológicas domesticas más popular de la actualidad. Además de estaciones meteorológicas, esta firma francesa desarrolla productos como termostatos inteligentes y otros tipos de sensores y actuadores. Destaca el hecho de la excelente integración de sus dispositivos con los *smartphones* (a través de aplicaciones muy bien valoradas en las tiendas de aplicaciones principales).

En cuanto a la publicación de los datos meteorológicos de sus estaciones en Internet, Netatmo dispone de aplicaciones web y móviles para que los usuarios de sus estaciones puedan acceder a sus datos. Estas aplicaciones están bien valoradas y disponen de un gran abanico de funcionalidades: variables meteorológicas en tiempo real, gráficos históricos, etc.

Además de ello, Netatmo ofrece la plataforma *Netatmo Connect* [4]. Esta plataforma consiste en un API REST desde el cual se pueden obtener las variables meteorológicas de la estación indicada.

Pese a ofrecer buenas herramientas, estas solo con compatibles con estaciones Netatmo.

2.2.3. *Oregon Scientific Anywhere Weather*

Oregon Scientific Global Distribution Ltd. [5] es un fabricante de todo tipo de aparatos electrónicos: despertadores, radios, relojes, etc. En cuanto estaciones meteorológicas, ofrece algunos modelos capaces de publicar sus datos en Internet a través de la plataforma *Oregon Scientific Anywhere Weather* [6]. Esta plataforma consiste en diferentes aplicaciones (web y móviles) que permiten a los usuarios de las estaciones meteorológicas consultar los datos recogidos por las mismas en directo. Además, esta plataforma permite consultar datos históricos de la última semana.

Pese a que a priori la plataforma parece buena, esta presenta importantes carencias:

- No ofrece un API oficial con el cual se pueda interactuar desde aplicaciones terceras.
- El funcionamiento de la plataforma es ciertamente pésimo. Tanto las propias estaciones meteorológicas como las aplicaciones de la plataforma tienen valoraciones relativamente malas. Los clientes se refieren en sus valoraciones a las constantes caídas de la plataforma e incluso a las malas mediciones realizas por la misma o por las propias estaciones.
- Únicamente ofrece datos históricos de la última semana.
- La plataforma solo es compatible con estaciones Oregon Scientific.

2.2.4. *WeatherCloud*

Esta plataforma es completamente diferente a las anteriores, ya que no está vinculada a ningún fabricante. *WeatherCloud* [7] consiste en una suerte de red social, en la cual un usuario puede

añadir su estación meteorológica (el modelo de estación deberá ser reconocido por la plataforma) y compartir sus datos meteorológicos con el resto de usuarios de la plataforma.

Además de reconocer un amplio abanico de modelos de estaciones meteorológicas, esta plataforma permite a los usuarios ver los datos de las estaciones meteorológicas en tiempo real así como información histórica.

La plataforma está accesible desde cualquier dispositivo con navegador web a través de la URL https://weathercloud.net/.

Si bien se trata de una herramienta potente, se diferencia de la desarrollada en este proyecto en que *WeatherCloud* está más orientada al ámbito doméstico o no profesional. Por ejemplo, *WeatherCloud* no ofrece un API oficial con el que poder acceder fácilmente a los datos de una determinada estación.

2.2.5. *Weather Underground*

Al igual que la anterior, esta plataforma no es exclusiva de ningún fabricante y por lo tanto, los usuarios pueden vincular sus estaciones (independientemente del fabricante o modelo) a la misma. Esta plataforma está accesible a través de la URL https://www.wunderground.com/.

A diferencia de la plataforma anterior *Weather Underground* [8] si ofrece un API oficial a través del cual obtener información meteorológica de una estación indicada. Posiblemente esta solución sea la más similar a la propuesta en este proyecto, si bien tiene algunas deficiencias:

- Para poder registrar una estación meteorológica en esta plataforma, el usuario debe disponer de un software específico, denominado *datalogger*, que se encargará de enviar los datos a *Weather Underground*. Es decir, *Weather Underground* proporciona un ID y una contraseña que deberán ser introducidos por el usuario en su software *datalogger* de forma que este software envíe periódicamente los datos a *Weather Underground*. La solución propuesta en este proyecto pretende ser más sencilla, siendo la plataforma desarrollada la encargada de recoger los datos desde las plataformas de los fabricantes.
- *Weather Underground* no ofrece la posibilidad de incrustar la información meteorológica en el sitio web del usuario.
- Si bien se trata de una herramienta potente, cabe destacar su poca usabilidad.

2.2.6. Otros

Existen más fabricantes de estaciones meteorológicas, algunos de ellos tienen también plataformas con las cuales se puede acceder a los datos meteorológicos de las estaciones a través de Internet. Sin embargo no se han nombrado todos en este documento debido a su escasa penetración en el mercado (al menos en el mercado español). Algunos de estos fabricantes son: WeatherHawk, Campbell Scientific, Vaisala, etc.

2.2.7. Revisión de las Soluciones Actuales

Tal y como se ha comentado en este capítulo, existen varias soluciones para los usuarios que quieran consultar la información de sus estaciones meteorológicas a través de Internet. Este proyecto pretende sin embargo ir un paso más allá y ofrecer una solución sencilla, multiplataforma, que permita integrar diferentes estaciones meteorológicas de diferentes

fabricantes y que haga accesible la información meteorológica de forma gráfica (para usuarios a través de un navegador web) y programática (a través de un API oficial y estándar).

Por lo tanto, las principales carencias de las soluciones actuales (que han sido tenidas en cuenta en la especificación de este proyecto) son las siguientes:

- El número de estaciones reconocidas por las plataformas puede ser bastante limitado: la solución propuesta deberá ser fácilmente escalable en el sentido de que se permita añadir nuevos modelos de estación reconocidos.
- Muchas plataformas no ofrecen datos históricos, dando información únicamente de los datos meteorológicos contemporáneos al momento de la consulta.
- Los datos no son siempre accesibles a través de un API bien definido y sobre protocolos estándar, lo que dificulta la interoperabilidad entre sistemas.
- Algunas plataformas no se caracterizan por su usabilidad: en muchas ocasiones son complicadas de utilizar y la visualización de los datos no es del todo clara.
- Finalmente, algunas plataformas presentan altas latencias y rendimientos bastante pobres.

3. Propuesta de Solución

En esta sección del documento se detalla la solución propuesta en este proyecto. En primera instancia se realiza un análisis de los requisitos que debe satisfacer la plataforma desarrollada:

3.1. Análisis de Requisitos

A continuación se presentan los requisitos funcionales en forma de tabla de requisitos:

RF1	Existen dos tipos de usuarios reconocidos por la plataforma: administradores y no administradores.
RF2	Existen diferentes lectores de datos de estación, denominados *parsers*, que se encargan de leer los datos meteorológicos de una determinada estación meteorológica ubicados en la plataforma pertinente (por ejemplo, leer los datos de una estación Davis ubicados en la *Davis WeatherLink Network*).
RF3	Existen diferentes modelos de estación meteorológica reconocidos por la plataforma. Cada modelo de estación meteorológica deberá tener asociado un lector de datos de estación.
RF4	Los usuarios administradores tienen la posibilidad de crear/modificar/eliminar lectores de datos de estación.
RF5	Los usuarios administradores tienen la posibilidad de crear/modificar/eliminar modelos de estación meteorológica reconocidos por la plataforma.
RF6	Los usuarios no administradores tienen la posibilidad de registrar/modificar/eliminar una estación meteorológica, cuyo modelo sea reconocido por la plataforma.
RF7	Para cada estación registrada en la plataforma, se leerán sus datos meteorológicos y se almacenarán periódicamente.
RF8	Los usuarios administradores pueden registrar/modificar/eliminar estaciones meteorológicas de cualquier usuario.
RF9	Los usuarios no administradores podrán crear/modificar/eliminar publicaciones vinculadas a sus estaciones meteorológicas. La publicación mostrará los datos meteorológicos que el usuario desee para la estación indicada.
RF10	Las publicaciones podrán ser incrustadas en páginas web de terceros fácilmente.
RF11	Los usuarios administradores podrán crear/modificar/eliminar publicaciones sin restricción alguna.
RF12	La información meteorológica de las estaciones deberá ser accesible, tanto en tiempo real como información histórica.

Tabla 1: Requisitos funcionales.

Ahora se presentan los requisitos no funcionales de la plataforma:

RNF1	Los usuarios contendrán al menos la siguiente información: • Nombre • Apellidos • Correo Electrónico • Contraseña • Activo/Inactivo • Rol (administrador y/o usuario) • Fecha de Creación • Fecha de Último Acceso

RNF2	Los lectores de datos de estación (*parsers*) requerirán la información necesaria al usuario para que los primeros puedan leer correctamente los datos de las estaciones meteorológicas (por ejemplo: ID de estación) desde la plataforma pertinente.
RNF3	Los modelos de estación contendrán al menos la siguiente información: • Fabricante • Nombre • Número de Producto • Lector de Datos Asociado • Fecha de Creación
RNF4	Las estaciones meteorológicas deberán contener al menos la siguiente información: • Usuario Propietario • Nombre (de la estación, definido por el Usuario) • Modelo de Estación (de entre uno de los reconocidos) • Ubicación • Zona Horaria • Activa/Inactiva • Fecha de Creación • Datos Requeridos por el Lector de Estaciones
RNF5	Las publicaciones de información meteorológica deberán contener al menos la siguiente información: • Nombre (de la publicación, definido por el Usuario) • Estación Meteorológica • Datos Mostrados (por ejemplo: viento en km/h y temperatura en ºF) • Activa/Inactiva • Fecha de Creación
RNF6	La información meteorológica de las estaciones, tanto en tiempo real como histórica, estará accesible de forma gráfica y mediante un API REST.

Tabla 2: Requisitos no funcionales.

3.2. Arquitectura

En este apartado se detalla la arquitectura propuesta a alto nivel. En la figura 1 se puede apreciar un diagrama de la arquitectura propuesta:

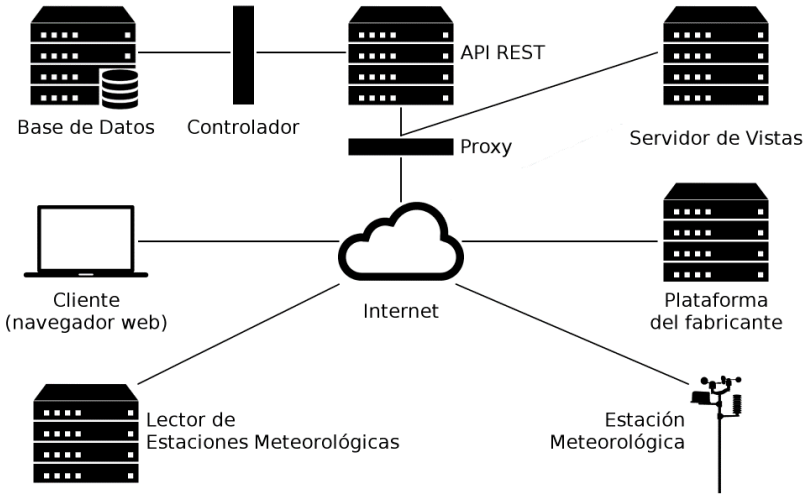

Figura 1: Arquitectura propuesta.

A continuación se detalla cada componente de la figura 1:

- Estación Meteorológica: la estación meteorológica se encarga de leer las variables meteorológicas de su entorno y enviarlas a través de Internet a la plataforma pertinente. Por ejemplo; si la estación meteorológica es una estación Davis, la estación enviará los datos meteorológicos recogidos a la plataforma *Davis WeatherLink Network* (descrita anteriormente).
- Plataforma del Fabricante: se encarga de recibir los datos meteorológicos tomados por las estaciones meteorológicas del fabricante en cuestión. Esta plataforma es completamente ajena a la plataforma desarrollada en este proyecto.
- Lector de Estaciones Meteorológicas: este componente es el encargado de, a través de Internet, tomar los datos meteorológicos de la estación indicada en la plataforma del fabricante de la misma. Una vez el Lector de Estaciones Meteorológicas ha leído los datos, los registra en *ownmeteo.com* consumiendo el API REST. Es decir, siguiendo el ejemplo anterior; los datos de la estación meteorológica Davis publicados en la *Davis WeatherLink Network* son leídos por el Lector de Estaciones Meteorológicas y este los registra a través del API REST. Es importante destacar que podría haber diferentes Lectores de Estaciones Meteorológicas trabajando simultáneamente, leyendo cada uno de ellos un subconjunto del conjunto total de estaciones registradas en la plataforma.
- API REST: el API REST se correspondería con la capa de modelo de negocio de una aplicación web tradicional de 3 capas (vista, modelo, datos). Este API se encarga de: recibir las peticiones de registro de información meteorológica por parte del Lector de Estaciones Meteorológicas, de recibir las peticiones de consulta de información meteorológica por parte de los Clientes, y de recibir peticiones de gestión de datos operativos por parte de los Clientes (por ejemplo: crear un nuevo usuario). Tal y como se aprecia en la figura 1, este componente se comunica con la Base de Datos para registrar y obtener datos.
- Servidor de Vistas: este componente es el encargado de servir la aplicación web al cliente. Es decir, se trata de un servidor web cuyo objetivo es entregar las vistas, *scripts* y demás componentes de la aplicación web ejecutada en el cliente (normalmente un navegador web).
- Servidor *proxy*: encargado de encaminar las peticiones recibidas desde los clientes al servidor pertinente (ya sea al API REST o al servidor de vistas). Gracias a este servidor, el cliente se abstrae de la existencia de varios servidores en la plataforma, ya que cree comunicarse con un único servidor. Más adelante en este documento se justificará más en profundidad la utilidad de este componente.
- Base de Datos: este componente se encarga de almacenar los datos de la plataforma. Cabe a destacar que la plataforma tiene dos grandes conjuntos de datos: datos meteorológicos y datos operativos (por ejemplo: usuarios, estaciones meteorológicas registradas, etc.).
- Cliente: este componente consulta el API REST para obtener la información meteorológica pertinente además de para gestionar los datos operativos de la aplicación. Es destacable que este componente puede ser la propia aplicación web desarrollada en este proyecto (tal y como se muestra en la figura 1) o una aplicación tercera.

3.3. Tecnologías

Principalmente, la tecnología empleada para el desarrollo de este proyecto es el denominado *stack* MEAN [9]. El *stack* MEAN es la conjunción de las siguientes tecnologías: MongoDB, Express, AngularJS y Node.js. Además de estas tecnologías se han utilizado otras como: HTML5, CSS3, Bootstrap, jQuery, NGINX, etc.

En la figura 2 se pueden apreciar las principales tecnologías empleadas en los diferentes componentes de la arquitectura propuesta en el apartado anterior:

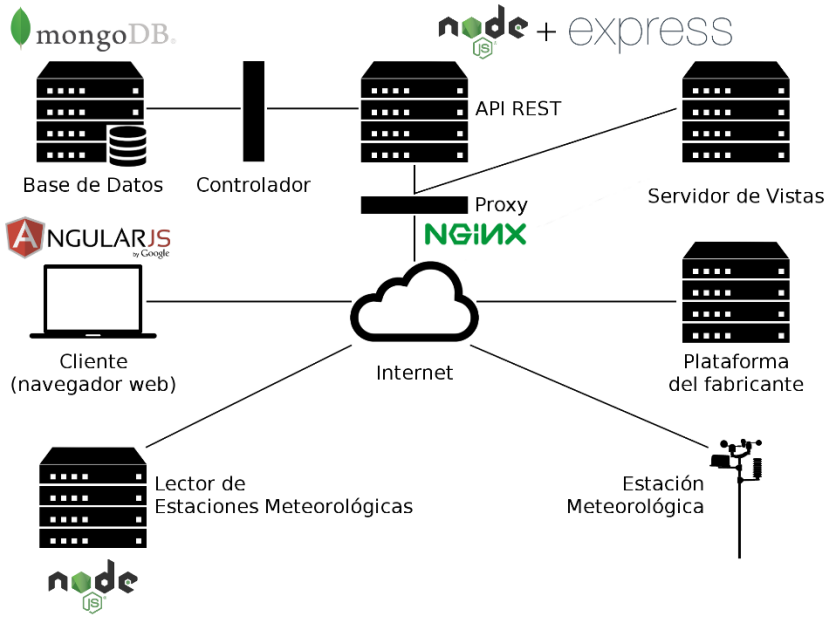

Figura 2: Principales tecnologías empleadas en los componentes de la arquitectura propuesta.

A continuación se describen detalladamente las principales tecnologías empleadas en el desarrollo de *ownmeteo.com*:

3.3.1. JavaScript

JavaScript [10] es un lenguaje de programación interpretado. Fue desarrollado originalmente para dar más dinamismo a sitios web (permite cambiar la posición o tamaño de los elementos de una página web, etc.), sin embargo, en la actualidad es utilizado en un amplio abanico de aplicaciones, ya sea en el lado del cliente (navegador web) como en el lado del servidor. JavaScript es un lenguaje interpretado de alto nivel y multiparadigma.

Excepto en la Base de Datos (para la que se ha utilizado un producto ya desarrollado; MongoDB), JavaScript ha sido utilizado en la implementación de todos los componentes de la plataforma desarrollada en este proyecto:

En la aplicación web del Cliente se utiliza principalmente para poder consumir el API REST y para darle más dinamismo.

Por otro lado, el API REST y el servidor de vistas han sido implementados utilizando Node.js, un entorno en tiempo de ejecución (del que se habla más adelante) basado en JavaScript.

Finalmente, los programas ejecutados por el Lector de Estaciones Meteorológicas para tomar los datos meteorológicos desde las plataformas de los fabricantes de estaciones también han sido implementados con Node.js.

Este proyecto ha sido implementado con JavaScript, no tanto por las virtudes de este lenguaje, sino por las ventajas que ofrecen las herramientas basadas en JavaScript que se han utilizado (y de las que se hablará más adelante). Sin embargo, el hecho de que todo el proyecto esté desarrollado en JavaScript ofrece algunas ventajas:

- La interoperabilidad entre los componentes de la aplicación es óptima. Esto se debe principalmente al uso de objetos JSON, que JavaScript soporta de forma nativa. Si hubiera un componente implementado en otro lenguaje, posiblemente hubiera sido necesario hacer uso de bibliotecas de terceros para poder codificar y decodificar los objetos JSON.
- Se trata de un lenguaje no propietario, de forma que no hay que pagar ningún tipo de licencia por su uso.
- JavaScript es uno de los lenguajes más conocidos y populares de la actualidad, lo que provoca que haya disponible gran cantidad de documentación del mismo.

3.3.2. Node.js

Tal y como se ha comentado anteriormente, Node.js [11] es un entorno en tiempo de ejecución basado en el motor JavaScript V8 de Google. Principalmente, su objetivo es utilizar JavaScript en el lado del servidor, aprovechando la arquitectura dirigida a eventos del lenguaje para poder desarrollar sistemas más escalables.

Así, Node.js se ha vuelto un entorno muy popular para el desarrollo de aplicaciones web debido a su escalabilidad. A diferencia de Apache[1] (y de la mayoría de servidores web), Node.js no es ni multi-proceso ni multi-hilo, ya que aprovecha la arquitectura dirigida a eventos de JavaScript. Al no tener que lidiar con varios procesos/hilos, Node.js suele ser más escalable (la coexistencia de múltiples procesos/hilos provoca dificultades y un gran consumo de recursos computacionales) que otras soluciones multi-proceso o multi-hilo.

A continuación se describen las principales ventajas ofrecidas por Node.js para el desarrollo del proyecto:

- Mayor escalabilidad.
- Entorno bajo la licencia libre MIT.
- Gran cantidad de *frameworks* disponibles en la red para Node.js.

Node.js ha sido utilizado en este proyecto para la implementación del API REST, del servidor de vistas y de los programas ejecutados por el Lector de Estaciones Meteorológicas.

3.3.3. Express

Express [12] es un *framework* para facilitar el desarrollo de aplicaciones web con Node.js. En este proyecto, Express ha sido utilizado para el desarrollo del API REST.

[1] Apache: servidor HTTP desarrollado por *Apache Software Foundation*, actualmente uno de los más populares del mercado.

3.3.4. MongoDB

MongoDB [13] es una base de datos NoSQL orientada a documentos. Básicamente, esta base de datos almacena documentos similares a objetos JSON sin un esquema predeterminado. Es decir, a diferencia de las bases de datos tradicionales SQL donde se deben especificar los campos de datos de una tabla, en MongoDB no se impone restricción alguna sobre la estructura de los documentos almacenados en una colección. Por lo tanto, en una misma colección pueden coexistir documentos con estructuras completamente diferentes.

La razón por la cual se ha escogido MongoDB como almacén de datos es precisamente la expuesta en el párrafo anterior. Como se ha comentado anteriormente, la plataforma desarrollada en este proyecto debe permitir el registro de estaciones meteorológicas con diferentes modelos e incluso diferentes fabricantes. Es por ello, que los datos a almacenar de cada estación serán diferentes: datos requeridos para el acceso a la plataforma del fabricante de dicha estación, datos meteorológicos leídos por dicha estación, etc. Así, con estos requisitos, MongoDB es una buena solución para gestionar los datos de la plataforma.

3.3.5. AngularJS

AngularJS [14] es un *framework* JavaScript para el cliente (navegador web) que facilita el desarrollo de aplicaciones web MVC (modelo, vista, controlador) y que se suele utilizar junto con las tecnologías descritas anteriormente en el desarrollo de aplicaciones web. AngularJS ha sido utilizado en la implementación de la aplicación web para el consumo del API REST.

4. Desarrollo

En este capítulo se describen las principales decisiones tomadas durante el proceso de desarrollo del proyecto. En primera instancia se explican las decisiones tomadas en el *back-end* de la plataforma.

4.1. *Back-end*

Esencialmente, el *back-end* de una aplicación web es la parte de la misma que contiene el grueso de la lógica de negocio, así como el almacén de datos. Por lo tanto, en *ownmeteo.com* el *back-end* está compuesto de: la Base de Datos, el API REST y el Lector de Estaciones Meteorológicas.

4.1.1. Base de Datos

En este apartado del documento se describe el modelo de datos diseñado para la plataforma. Es importante destacar que, pese a que MongoDB (el gestor de datos escogido para la plataforma) no precisa del diseño de un modelo de datos (ya que una misma colección puede albergar documentos completamente diferentes entre ellos), sí que es altamente recomendable realizar una pequeña especificación del mismo.

En la siguiente tabla se muestran las colecciones de datos utilizadas en la plataforma:

Usuarios	Colección de datos que almacena los datos de los usuarios de la plataforma. A continuación se muestran los datos almacenados en cada documento de esta colección: • Nombre: cadena de texto. • Apellidos: cadena de texto. • Correo Electrónico: cadena de texto. • Contraseña: cadena de texto (contraseña encriptada con SHA-512). • Activo/Inactivo: booleano. • Rol: vector de cadenas de texto con los roles del usuario. • Fecha de Creación: fecha. • Fecha de Último Acceso: fecha.
Tokens de Sesión	Colección que almacena los *tokens* de sesión. Los *tokens* de sesión son utilizados para autenticar a los clientes del API REST. A continuación se muestran los datos almacenados en cada documentos de la colección: • Usuario propietario: ID de documento. • *Token*: cadena de texto (cuyo contenido es el *token* que autentica al cliente en sus peticiones al API REST). • Fecha de Creación: fecha. • Fecha de Expiración: fecha.
Parsers	Colección que almacena los datos de los distintos programas necesarios para recoger los datos meteorológicos desde las plataformas de los fabricantes de estaciones meteorológicas. A continuación se muestran los datos almacenados en cada documento de la colección: • Nombre: cadena de texto. • Descripción: cadena de texto. • Fecha de Creación: fecha.

Datos Requeridos por los *Parsers*	Colección que almacena los datos requeridos por los *Parsers* para recoger los datos meteorológicos desde las plataformas de los fabricantes. Es decir, si por ejemplo una plataforma requiere del ID de la estación, existirá un documento en esta colección que indicará que el *Parser* de la plataforma requiere dicho ID. A continuación se muestran los datos almacenados en cada documento de la colección: • Nombre Interno: cadena de texto. • Nombre: cadena de texto. • Descripción: cadena de texto. • *Parser*: ID de documento (referenciando al *Parser* que requiere el dato descrito por el documento).
Modelos de Estación	Colección que almacena los diferentes Modelos de Estación reconocidos por la plataforma. A continuación se muestran los datos almacenados en cada documento de la colección: • Fabricante: cadena de texto. • Nombre: cadena de texto. • Número de Producto: cadena de texto. • *Parser*: ID de documento (referencia al *Parser* encargado de leer los datos de las estaciones meteorológicas cuyo modelo sea el descrito por el documento). • Fecha de Creación: fecha.
Unidades de Medida	Colección que almacena las posibles Unidades de Medida de las Variables Meteorológicas (por ejemplo: km/h, ºC, ºF, hPa, etc.). A continuación se muestran los datos almacenados en cada documento de la colección: • Unidad de Medida: cadena de texto.
Variables Meteorológicas	Colección de posibles Variables Meteorológicas. A continuación se muestran los datos almacenados en cada documento de la colección: • Nombre: cadena de texto (por ejemplo: Viento). • Unidad de Medida: ID de documento (referencia a la Unidad de Medida de la Variable Meteorológica descrita por el documento).
Estaciones Meteorológicas	Colección que almacena las Estaciones Meteorológicas registradas en la plataforma. A continuación se muestran los datos almacenados en cada documento de la colección: • Usuario Propietario: ID de documento (referencia al Usuario propietario de la Estación descrita por el documento). • Nombre: cadena de texto. • Modelo de Estación: ID de documento (referencia al Modelo de Estación de la Estación descrita por el documento). • Ubicación: cadena de texto. • Zona Horaria: número. • Activa/Inactiva: booleano. • Fecha de Creación: fecha. • Datos Requeridos por el *Parser*: objeto (objeto que contiene pares clave-valor con los Datos Requeridos por el *Parser*).

Publicaciones	Colección que almacena las Publicaciones de Información Meteorológica. A continuación se muestran los datos almacenados en cada documento de la colección: • Estación Meteorológica: ID de documento (referencia a la Estación Meteorológica de la cual se está publicando la información). • Nombre: cadena de texto. • Datos Mostrados: vector de IDs de documentos (referenciando a las Variables Meteorológicas mostradas en la Publicación). • Activa/Inactiva: booleano. • Fecha de Creación: fecha.
Accesos al API	Colección que almacena los Accesos al API. A continuación se muestran los datos almacenados en cada documento de la colección: • URL solicitada: cadena de texto. • Fecha de Acceso: fecha. • Código HTTP de Respuesta: número. • Usuario: ID de documento (referencia al Usuario que ha realizado el Acceso, si lo hubiere). • Dirección IP: cadena de texto. • Navegador Web (*User-Agent*): cadena de texto.
Incidencias	Colección que almacena las Incidencias acontecidas. A continuación se muestran los datos almacenados en cada documento de la colección: • Código: cadena de texto. • Nivel: cadena de texto. • Descripción: cadena de texto. • Fecha: fecha.

Tabla 3: Colecciones de datos de la base de datos.

4.1.2. API REST

Como ya se ha comentado previamente, el API REST [15] es la interfaz con la cual los usuarios gestionarán los datos de la aplicación desde el exterior. Este API añade la lógica de negocio pertinente sobre los datos de la plataforma para que estos sean manipulados correctamente.

Los tipos de recursos de este API REST son, principalmente, los tipos de datos descritos en el modelo de datos (apartado anterior). Así, cada tipo de recurso tendrá una URI base o *endpoint* a partir del cual se realizarán las operaciones pertinentes a dicho tipo de recurso. En la tabla 4 se pueden apreciar los *endpoints* asociados a los diferentes tipos de recursos del API:

URI	Tipo de Recurso
/api/users	Usuarios
/api/sessions	*Tokens* de Sesión
/api/parsers	*Parsers*
/api/parser-required-data	Datos Requeridos por los *Parsers*
/api/station-models	Modelos de Estación
/api/meteo-vars	Variables Meteológicas
/api/stations	Estaciones Meteorológicas
/api/posts	Publicaciones
/api/api-accesses	Accesos al API
/api/incidences	Incidencias

Tabla 4: *Endpoints* de los recursos expuestos por el API.

Al ser un API REST, las operaciones se realizan sobre el protocolo HTTP [16]. El protocolo HTTP ofrece diferentes métodos de petición (o verbos), los cuales se utilizan en función de la acción que se desee realizar sobre el recurso indicado. Algunos de estos métodos de petición son: GET, POST, HEAD, PUT, DELETE, TRACE, etc.

Para el desarrollo del API REST de *ownmeteo.com* se han utilizado los métodos de petición correspondientes al repertorio CRUD (*Create*, *Retrieve*, *Update*, *Delete*) de HTTP, que son los siguientes:

- GET: este método se ha utilizado para obtener información de los recursos del API. Si en la URI de la petición GET solo se especifica el tipo de recurso, el API responderá con el listado de recursos existentes para ese tipo de recurso. Si además la URI incluye un ID, el API responderá con el recurso del tipo indicado cuyo ID se corresponda con el de la URI de la petición. Por ejemplo, si se lanza una petición con la URI /api/users, el API devolverá el listado completo de Usuarios; si se lanza una petición con la URI /api/users/:id[2] el API devolverá la información del usuario cuyo ID sea el indicado en la URI.
- POST: este método ha sido utilizado para la creación de nuevos recursos en función del tipo de recurso indicado en la URI de la petición. Así, la petición POST deberá incluir en la URI el tipo de recurso a crear y, en el cuerpo de la petición (*payload*) la información del nuevo recurso. En casos aislados, el método POST se ha utilizado también para obtener información de forma parametrizada (i.e.: estableciendo algún filtro en el *payload*).
- PUT: este método ha sido utilizado para actualizar la información del recurso indicado en la URI de la petición. Por lo tanto, la URI de la petición PUT deberá incluir el tipo de recurso a actualizar y su ID y, en el cuerpo de la petición (*payload*) deberá estar la nueva información del recurso a actualizar.
- DELETE: este método ha sido utilizado para eliminar el recurso indicado en la URI de la petición. Así pues, en las peticiones DELETE se deberá incluir en la URI el tipo de recurso y el ID del recurso a eliminar.

En la tabla mostrada a continuación se muestran las diferentes URIs expuestas por el API REST y los métodos de petición que cada URI acepta:

URI	GET	POST	PUT	DELETE
/api/users	✓	✓		
/api/users/:id	✓		✓	✓
/api/users/:id/sessions	✓			
/api/users/:id/stations	✓			
/api/sessions	✓	✓		
/api/sessions/:id	✓			✓
/api/sessions/current	✓			✓
/api/parsers	✓	✓		
/api/parsers/:id	✓		✓	✓
/api/parsers/:id/parser-required-data	✓	✓		
/api/parsers/:id/station-models	✓			
/api/parser-required-data	✓			
/api/parser-required-data/:id	✓			✓
/api/station-models	✓	✓		

[2] :id indica una ID válida, por ejemplo: 58cac30322601d18308b4594

/api/station-models/:id	✓		✓	✓
/api/station-models/:id/stations	✓			
/api/meteo-vars	✓			
/api/stations	✓	✓		
/api/stations/:id	✓		✓	✓
/api/stations/:id/public	✓			
/api/stations/:id/available-meteo-vars	✓			
/api/stations/:id/weather	✓	✓	✓	
/api/stations/:id/posts	✓			
/api/posts	✓	✓		
/api/posts/:id	✓		✓	✓
/api/api-accesses	✓	✓		
/api/api-accesses/:id	✓			
/api/incidences	✓	✓	✓	
/api/incidences/:id	✓			

Tabla 5: Relación de métodos de petición aceptados por los diferentes *endpoints* del API REST.

En el Anexo A se describen las acciones llevadas a cabo por el API cuando se lanzan peticiones contra las URIs mostradas en la tabla anterior.

El API requiere que los clientes del mismo se autentiquen para ciertas operaciones. La autenticación se lleva a cabo creando una nueva sesión; invocando mediante el verbo POST el *endpoint* /api/sessions. Al crear una sesión, el cliente recibe un *token* de sesión que le permite acceder a las operaciones del API que requieren autenticación (para más información acerca de la creación de sesiones, consultar el Anexo A). Este *token* debe ser enviado por el cliente en forma de *cookie*[3] en cada petición que realice al API. Al llegar cada petición, el API analiza el *token* de sesión recibido y trata de autenticar al usuario propietario de dicho *token*. A continuación se muestra el diagrama de secuencia que ilustra el proceso de autenticación:

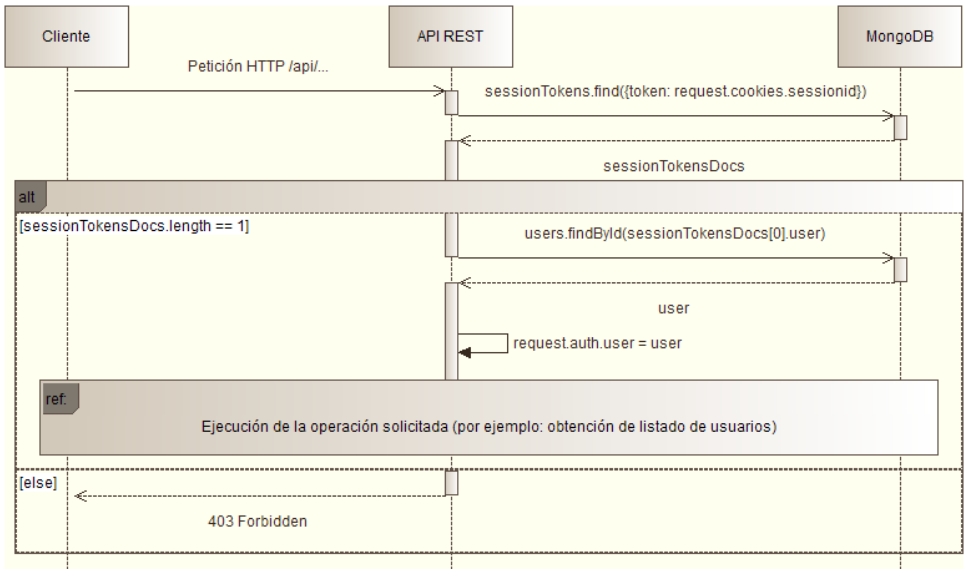

Figura 3: Diagrama de secuencia del proceso de autenticación.

[3] *Cookie*: pequeño fragmento de información enviado por un servidor web que es almacenado en el navegador del cliente. El navegador enviará dicha *cookie* en las futuras peticiones que realice al servidor.

El proceso de autenticación se realiza antes de ejecutar la operación solicitada por el cliente. Solo si el proceso de autenticación concluye satisfactoriamente se procederá con la acción solicitada por el cliente.

Finalmente, es importante destacar que el cuerpo de todos los mensajes enviados y recibidos por el API estará codificado en formato JSON.

4.1.3. Lector de Estaciones Meteorológicas

El último componente del *back-end* de *ownmeteo.com* es el Lector de Estaciones Meteorológicas. Tal y como se ha comentado anteriormente, el Lector de Estaciones Meteorológicas es un nodo separado del API REST pero dependiente de este último.

Básicamente este componente se encarga de, periódicamente, consultar la información meteorológica recogida por las estaciones registradas en la plataforma y enviarla al API REST. Esta información es recogida desde las plataformas de los diferentes fabricantes de estaciones mediante el uso de los *parsers* de *ownmeteo.com*.

Los *parsers* son los programas (implementados con Node.js) encargados de recoger la información meteorológica de las estaciones a través de la plataforma del fabricante de dichas estaciones. Es decir, estos programas contienen la lógica necesaria para leer los datos meteorológicos de una estación indicada en la plataforma de su fabricante. Además de leer la información meteorológica en la plataforma del fabricante, el *parser* también se encarga de enviar dicha información meteorológica al API de *ownmeteo.com* de forma que quede registrada.

Como ya se ha comentado anteriormente, los *parsers* son gestionados por los usuarios administradores de la plataforma, de forma que estos últimos pueden crear nuevos *parsers* (especificando su nombre y descripción y subiendo su código fuente) o modificar los existentes. Estos *parsers* serán vinculados posteriormente (por los propios administradores) a los modelos de estación meteorológica.

Así pues, el componente descrito en este apartado (el Lector de Estaciones Meteorológicas) procede de la siguiente manera:

1. En primera instancia, el Lector de Estaciones Meteorológicas obtiene (siempre a través del API REST) el listado de estaciones meteorológicas registradas en la plataforma.
2. A continuación, obtiene el listado de modelos de estación.
3. En tercer lugar, obtiene el listado de *parsers*.
4. Posteriormente, obtiene el código fuente de todos los *parsers*.
5. Finalmente, para cada estación selecciona su *parser* correspondiente (consultado el modelo de dicha estación) y lo ejecuta con los parámetros de la estación, de forma que el *parser* obtiene la información meteorológica y la registra en *ownmeteo.com* a través del API REST.

A continuación se muestra un diagrama de secuencia que ilustra de forma simplificada este comportamiento:

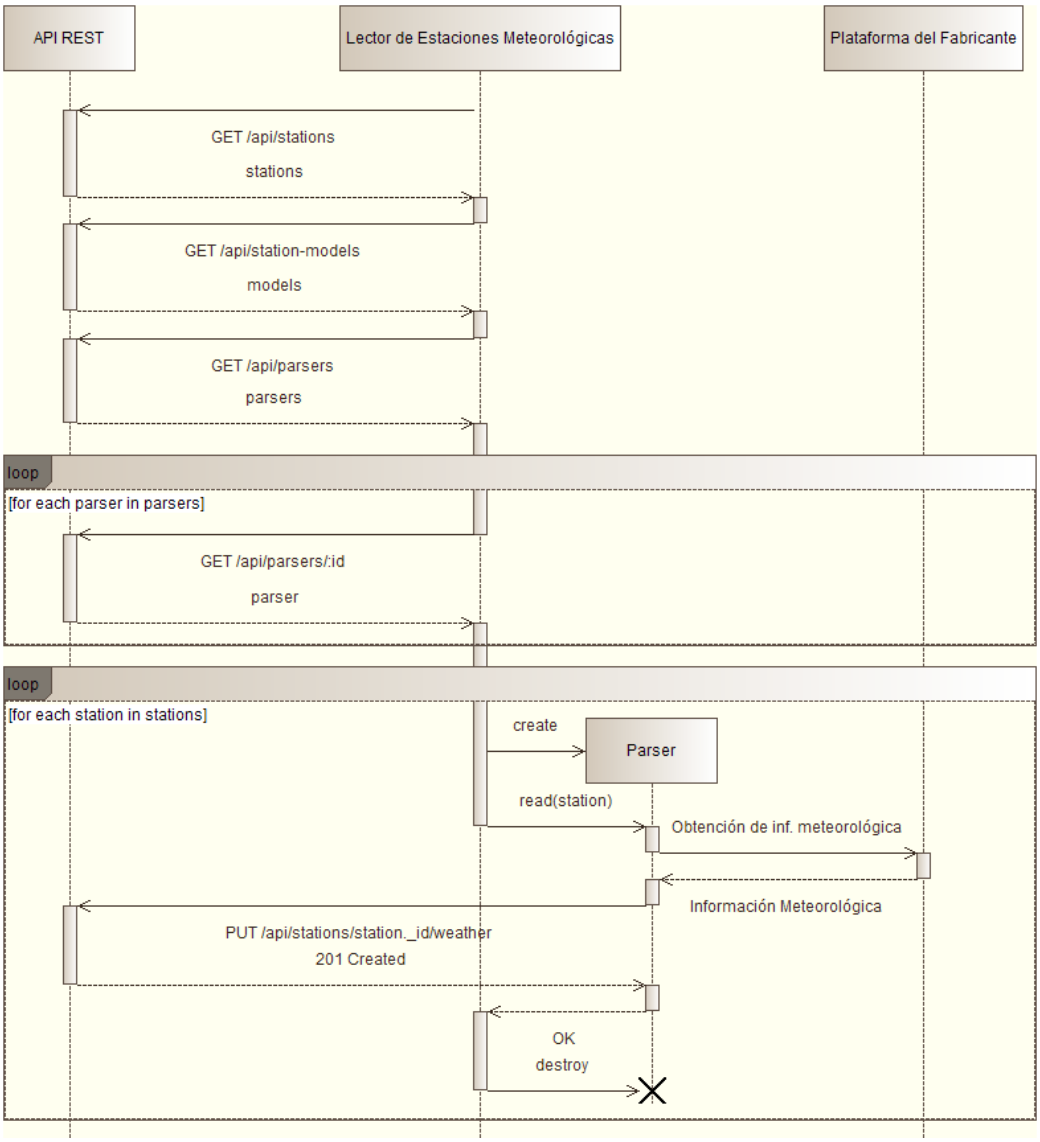

Figura 4: Diagrama de secuencia de funcionamiento del Lector de Estaciones Meteorológicas.

Finalmente, es importante destacar que, antes de ejecutar el comportamiento descrito, el Lector de Estaciones Meteorológicas debe autenticarse contra el API REST. Así pues, el Lector de Estaciones Meteorológicas se autentica como un cliente más, creando una sesión, de forma que en cada nueva petición que realice al API enviará su *token* de sesión. Debido a que las operaciones que solicita el Lector de Estaciones Meteorológicas al API REST requieren estar autenticado como usuario administrador, el Lector de Estaciones Meteorológicas se autentica como usuario administrador.

4.2. *Front-end*

El *front-end* de *owmeteo.com* consiste en la aplicación web desarrollada principalmente con el *framework* AngularJS. Además de AngularJS, para el desarrollo del *front-end* se han utilizado tecnologías como: HTML5, CSS3, JavaScript, jQuery, Bootstrap, etc. Sin embargo, la lógica de la aplicación web está implementada con AngularJS, por lo que en este documento se va a profundizar en la parte implementada con esta tecnología.

Normalmente, las aplicaciones desarrolladas con AngularJS se componen de diferentes partes. Existen multitud de tipos de partes, sin embargo las partes más importantes de una aplicación AngularJS son las siguientes:

- Módulo (*Module*): contenedor de diferentes partes de la aplicación (como por ejemplo: controladores, servicios, etc.).
- Vista (*View*): parte de la aplicación que muestra los datos al usuario (es decir, lo que el usuario ve).
- Modelo (*Model*): datos mostrados al usuario y con los cuales interactúa.
- Ámbito (*Scope*): ámbito donde el modelo es almacenado, de forma que el controlador, vista… puedan acceder a él.
- Controlador (*Controller*): parte de la aplicación que contiene la lógica de negocio detrás de las vistas (encargada de manipular el modelo a través del *scope*).
- Servicio (*Service*): parte de la aplicación que contiene lógica reusable (por ejemplo, una operación muy común que se ejecuta en múltiples vistas podría estar implementada en un servicio en lugar de en un controlador). AngularJS incorpora algunos servicios fundamentales (como por ejemplo el servicio `$http`, que permite realizar peticiones HTTP), pero además permite al desarrollador definir sus propios servicios.
- Directivas: ofrecen la posibilidad de extender HTML, permitiendo al desarrollador definir sus propios elementos y atributos, que serán procesados por AngularJS cuando la aplicación se ejecute.

Existen muchas más partes, sin embargo las aquí descritas son las más importantes. Una vez explicadas las principales partes de una aplicación AngularJS se procede a descomponer la aplicación web desarrollada, describiendo como se han utilizado estas partes:

4.2.1. Módulos

Como se ha comentado previamente, un módulo es un contenedor de diferentes partes de la aplicación. En este proyecto se ha utilizado un único módulo. Este módulo, denominado `ownmeteo` contiene los diferentes controladores, servicios, etc. de la aplicación web.

En los apartados posteriores se describe el contenido de este módulo.

4.2.2. Vistas y Controladores

En los sitios web tradicionales, únicamente se intercambia información con el servidor cuando el usuario cambia de página o vista, bien a través de un hipervínculo o bien a través del envío de un formulario. Es decir, para recibir nueva información, el cliente cambia de vista descargando un nuevo documento HTML desde el servidor.

En la aplicación web desarrollada, el cliente puede intercambiar información con el servidor sin cambiar de vista. Por lo tanto, la vista tiene la capacidad de enviar y recibir información. Evidentemente un documento HTML aislado no tiene capacidad para intercambiar información con el servidor, por lo que ahí es donde entran los controladores AngularJS.

Como se ha comentado previamente, los controladores AngularJS contienen la lógica de negocio de la aplicación web. Estos controladores se encargan de manipular el modelo (*model*) de la aplicación a través del *scope*, de forma que los cambios en el modelo se vean reflejados en la vista. Así, un controlador puede enviar o recibir información del servidor y mostrarla en la vista, evitando por lo tanto el cambio de vista para intercambiar información con el servidor. Además

de esto último, un controlador puede realizar multitud de tareas que con un documento HTML aislado no se podría: cambiar componentes de la vista, verificar la integridad de los datos de un formulario, etc.

La funcionalidad descrita en el párrafo anterior podría implementarse sin la necesidad de incluir AngularJS en la aplicación, a través del uso de JavaScript puro (o con la ayuda de otros *frameowrks* como jQuery). Así pues, la razón por la que se ha utilizado AngularJS en esta aplicación es el denominado *two-way data binding*. Mediante el uso de JavaScript puro, los cambios en la vista deben especificarse explícitamente, mientras que con AngularJS son implícitos.

Por ejemplo; si en JavaScript se modifica una variable debido a la descarga de información desde el servidor y se quiere mostrar al usuario dicho cambio, se deberá añadir una instrucción que lo haga:

```
var username = getUsernameFromServer();
document.getElementById("username-div").innerHTML = username;
```

En AngularJS no es necesario añadir dicha instrucción, ya que gracias al *two-way data binding*, las variables del controlador siempre están enlazadas con la vista (siempre y cuando dichas variables formen parte del *scope*). Por lo tanto, los componentes de la vista enlazados a una variable indicada siempre estarán actualizados. Así pues, si se tiene un componente en la vista como el siguiente:

```
<div ng-bind="username"></div>
```

Cuando la variable cambie en el controlador, dicho cambio se apreciará automáticamente en la vista. Es decir, con la siguiente instrucción sería suficiente para reflejar un cambio en la vista:

```
$scope.username = getUsernameFromServer();
```

Es importante destacar que, en AngularJS los cambios son bidireccionales. Es decir, si el usuario modifica una variable en la vista, esta modificación será accesible desde el controlador automáticamente.

A continuación se muestra un diagrama que ilustra el funcionamiento típico de JavaScript puro comparado con el *two-way data binding* de AngularJS:

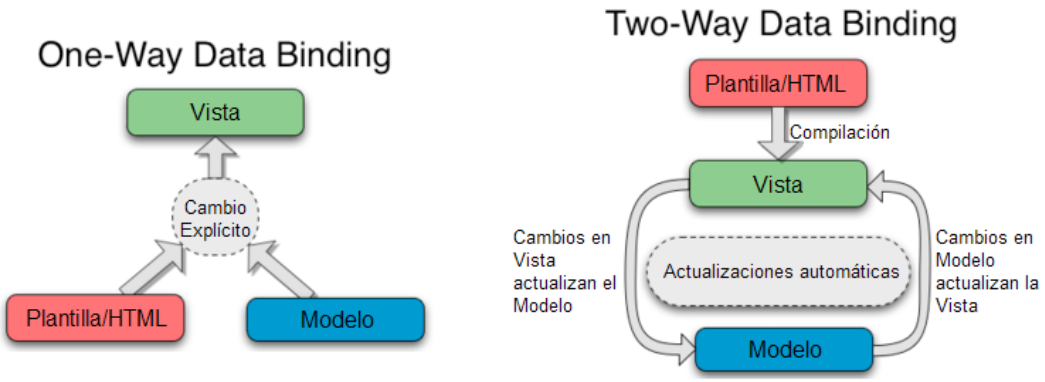

Figura 5: Comparación del enlace de datos Modelo-Vista.

En este proyecto, se han desarrollado múltiples vistas, cada una con uno o más controladores (normalmente solo uno), de las que se hablará más adelante.

4.2.3. Servicios

Como ya se ha explicado anteriormente, AngularJS dispone de un componente denominado servicio (*service*) cuya finalidad es encapsular funcionalidades altamente reutilizables. Los servicios pueden ser utilizados por los controladores e incluso por otros servicios. Estos servicios pueden ser definidos por el desarrollador, si bien AngularJS proporciona algunos servicios predefinidos.

En la aplicación web desarrollada en este proyecto se han utilizado tanto servicios predefinidos por AngularJS como servicios desarrollados a propósito.

En cuanto a los servicios predefinidos o incorporados en AngularJS, el más utilizado y relevante para este proyecto es el servicio `$http`. Este servicio permite al desarrollador realizar peticiones HTTP de forma sencilla y directa. A continuación se muestra un ejemplo de utilización de este servicio:

```
$http({
    method: "POST",
    url: "https://ownmeteo.com/api/sessions",
    data: {email: email, password: password}
}).then(function(res) {
    console.log("Session Created!");
}, function(res) {
    console.log("Error");
});
```

En cuanto a los servicios definidos por el desarrollador, se han creado dos: `api` y `gui`.

El servicio `api` tiene el objetivo de facilitar y simplificar el acceso al API REST de la plataforma. Este servicio abstrae algunos aspectos de acceso al API (como URIs, verbos HTTP, etc.), de forma que implementar dichos accesos es más rápido y sencillo. El servicio es utilizado por los controladores que realizan accesos al API. A continuación se muestra un ejemplo de uso de este servicio:

```
api.sessions.create($scope.user, callbackFunction);
```

El fragmento de código anterior crea una sesión a partir de los credenciales de usuario pasados. Tal y como se puede comprobar, el desarrollador que utiliza este servicio desconoce la URL del servicio web consumido o el verbo HTTP utilizado.

Es importante destacar que el servicio `api` utiliza el servicio predefinido de AngularJS `$http` para realizar los accesos al API.

Finalmente, el servicio `gui` simplifica la manipulación de la GUI para algunas operaciones muy frecuentes como por ejemplo; mostrar una advertencia al usuario.

4.2.4. Directivas

En última instancia, se van a comentar las directivas creadas para el desarrollo de la aplicación web de la plataforma *ownmeteo.com*. Como ya se ha comentado previamente, las directivas permiten al desarrollador establecer sus propios elementos y atributos en el documento HTML. Estas directivas serán procesadas por AngularJS cuando la página se ejecute, transformándolas en elementos HTML que el navegador si pueda renderizar. Es importante destacar que las directivas pueden tener sus propios controladores AngularJS.

Las directivas creadas para este proyecto son las siguientes: `navbar` y `spinner`.

La directiva `navbar` consiste principalmente en la barra de navegación del sitio web. Como es habitual, la mayoría de sitios web utilizan una barra de navegación común a lo largo de sus páginas o vistas, como es el caso de *ownmeteo.com*. Así pues, para facilitar la inclusión de la barra de navegación en las diferentes vistas de *ownmeteo.com* se ha creado la directiva `navbar`.

Gracias a esta directiva, incluir la barra de navegación en una vista del sitio web es tan sencillo como añadir la siguiente línea de código al documento HTML de dicha vista:

```
<navbar data-user-required-roles=""></navbar>
```

Como se puede apreciar, la barra de navegación contiene un atributo (que en el ejemplo anterior no tiene un valor asignado). En función del valor de este atributo, el navegador permitirá o no el uso de la vista donde esté insertada la directiva. Pueden darse los siguientes casos:

- El valor del atributo es `user`: el navegador permitirá el uso de la vista donde está insertada la directiva siempre y cuando el usuario esté autenticado como usuario en la plataforma. En caso contrario, se redirigirá al usuario a la página inicial del sitio web.
- El valor del atributo es `admin`: el navegador permitirá el uso de la vista donde está insertada la directiva siempre y cuando el usuario esté autenticado como administrador en la plataforma. En caso contrario, se redirigirá al usuario a la página inicial del sitio web.

Este comportamiento está implementado en el propio controlador de la directiva `navbar`. Es decir, el controlador de la directiva comprobará como está (si lo está) autenticado el usuario en el API REST y tras ello determinará si dicho usuario puede permanecer o no en la vista.

La barra de navegación varía en función del tipo de usuario que la esté utilizando. Si el usuario no está autenticado en el API REST (i.e. no tiene un *token* de sesión válido), en la barra de navegación aparecerán hipervínculos que permitan al usuario autenticarse o registrarse. Si el usuario está autenticado en el API como un usuario no administrador, la barra de navegación contendrá hipervínculos a sus recursos: estaciones meteorológicas, publicaciones de información meteorológica, etc. Si el usuario está autenticado en el API como un usuario administrador, la barra de navegación contendrá hipervínculos a recursos solo manipulables por administradores: *parsers*, modelos de estaciones, etc. Así pues, el controlador de la barra de navegación también contiene la lógica necesaria para variar los componentes de la barra de navegación en función del tipo de usuario que la esté utilizando.

Finalmente la directiva `spinner` únicamente contiene una animación de carga. Puesto que esta animación no se trata de un GIF tradicional, sino de un conjunto de elementos HTML acompañados de animaciones CSS, se concluyó que sería positivo agruparlos en una directiva.

4.3. Integración de *back-end* y *front-end*

Una vez descritos el *back-end* y *front-end* de la plataforma se explica brevemente su integración. Tal y como se ha explicado previamente, el *back-end* de la aplicación expone un API REST a través del cual se pueden manipular los datos almacenados en la base de datos. Puesto que el API REST funciona sobre el protocolo HTTP, el acceso al mismo desde un navegador web es relativamente sencillo.

Así pues, el *front-end* dispone de múltiples vistas que ofrecen al usuario una interfaz amigable con la cual manipular los datos de la aplicación. Esta interfaz descarga pues los datos de cierto recurso que el usuario desea manipular, consumiendo la operación pertinente del API REST. Una

vez descargados, los datos son mostrados al usuario, de forma que este puede editarlos si así lo desea. Si los datos son editados por el usuario y este guarda los cambios, la vista del *front-end* enviará una petición al API REST con el recurso modificado por el usuario. El recurso modificado será analizado por el API y si los datos son válidos (no se incumple ninguna restricción, por ejemplo; campos vacíos) los cambios serán guardados en la base de datos.

Tal y como se puede apreciar, la integración no entraña mayor dificultad. Cabe tener en cuenta que, además, al estar implementados tanto *front-end* como *back-end* con JavaScript, el intercambio de información es trivial gracias a los objetos JSON.

Sin embargo sí hay una dificultad en la integración: el API REST es un componente independiente del servidor web de vistas. Es decir, el API REST y el servidor de vistas son dos instancias que pueden incluso ser ejecutadas en diferentes máquinas. Así, podría darse que, por ejemplo, el servidor de vistas se ubicase en una máquina con nombre DNS "ownmeteo.com" mientras que el servidor del API REST se ubicase en una máquina con nombre "api.ownmeteo.com". O podría darse que ambos servidores se ubicasen en la misma máquina, pero entonces deberían utilizar dos puertos TCP diferentes.

El problema radica en que el navegador web no permite realizar peticiones HTTP con JavaScript (y por lo tanto, con AngularJS) a sitios web servidos por otro servidor que no sea el propio servidor de vistas. Es decir, para que el navegador web permita realizar una petición desde cierta página web con JavaScript, la URI solicitada deberá ser una URL que apunte al mismo servidor desde el que la vista se descargó.

Por lo tanto los navegadores web, en principio, no permitirán a los controladores AngularJS acceder al API, ya que este no está ubicado en el mismo servidor que el servidor de vistas. Evidentemente, este problema debe ser subsanado, ya que de no ser así, la aplicación web no podría funcionar. A continuación se enumeran las posibles soluciones:

- Ubicar en un mismo servidor las vistas y el API REST. Sería una solución funcional, sin embargo, el servidor resultante podría ser difícil de mantener.
- Habilitar las cabeceras HTTP *Cross-origin resource sharing* (CORS) [17]. Las cabeceras CORS permiten realizar peticiones con JavaScript a sitios web externos, por lo que solucionarían el problema. Si bien sería una solución válida, las cabeceras CORS no son fácilmente configurables (y más aún cuando las peticiones HTTP deben trabajar con *cookies* como en este caso).
- Habilitar un nuevo servidor que actué de *proxy*. Esta ha sido la solución escogida. Consiste en habilitar un servidor que haga de intermediario (servidor *proxy*) entre el cliente y los servidores del API y de vistas, de forma que el cliente cree estar comunicándose con un único servidor.

Así, se ha habilitado un tercer componente que es el servidor *proxy*. Este servidor recibirá en primera instancia las peticiones de los clientes, ya sean dirigidas al API o al servidor de vistas. El servidor *proxy* analizará la petición recibida y en función de la URI solicitada, reexpedirá la petición al servidor involucrado (el servidor del API o el de vistas). La petición será resuelta por el servidor pertinente (el servidor del API o el de vistas) y este último enviará la respuesta al servidor *proxy*. Finalmente el servidor *proxy* recibirá la respuesta del servidor involucrado (el servidor del API o el de vistas) y la reexpedirá al cliente. A continuación se muestra una figura que ilustra este esquema:

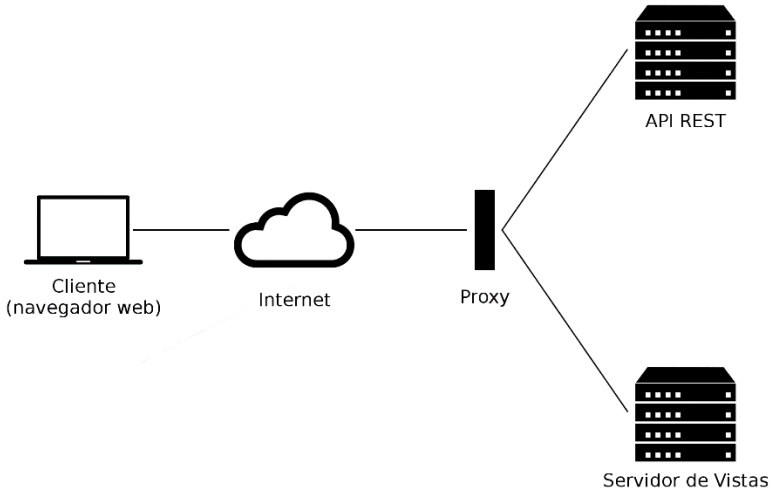

Figura 6: Servidor *proxy* propuesto.

Pese a que hay varios servidores involucrados en la resolución de peticiones, el cliente cree estar comunicándose con un único servidor, el servidor *proxy*. Esto soluciona la problemática expuesta inicialmente.

Así pues, el servidor *proxy* consiste en un servidor NGINX [18]. NGINX es un servidor web muy popular en el mercado debido a su ligereza y fácil configuración (frente a otras soluciones como el servidor web de Apache). La solución aquí propuesta no es nueva, ya que la arquitectura propuesta en este apartado es una arquitectura muy utilizada actualmente en cualquier desarrollo con Node.js. Si bien no hay bibliografía que avale esta arquitectura, esta es ampliamente utilizada, ya que NGINX puede hacer las funciones de un balanceador de carga entre instancias de Node.js, puede encriptar las comunicaciones creando así una conexión HTTPS, etc.

Así, aprovechando NGINX, se ha decidido encriptar las peticiones a *ownmeteo.com* mediante el uso del protocolo HTTPS [19]. Para ello, ha sido necesario instalar un certificado SSL en la máquina donde se ubica el servidor *proxy* y configurar NGINX para hacer uso de dicho certificado. Además, se configuró NGINX para aceptar solo peticiones HTTPS.

4.4. Aspecto Final

Por último, para finalizar el capítulo de Desarrollo, se añaden algunas capturas de pantalla relevantes de la aplicación web que muestran el aspecto final de la cara visible de *ownmeteo.com*:

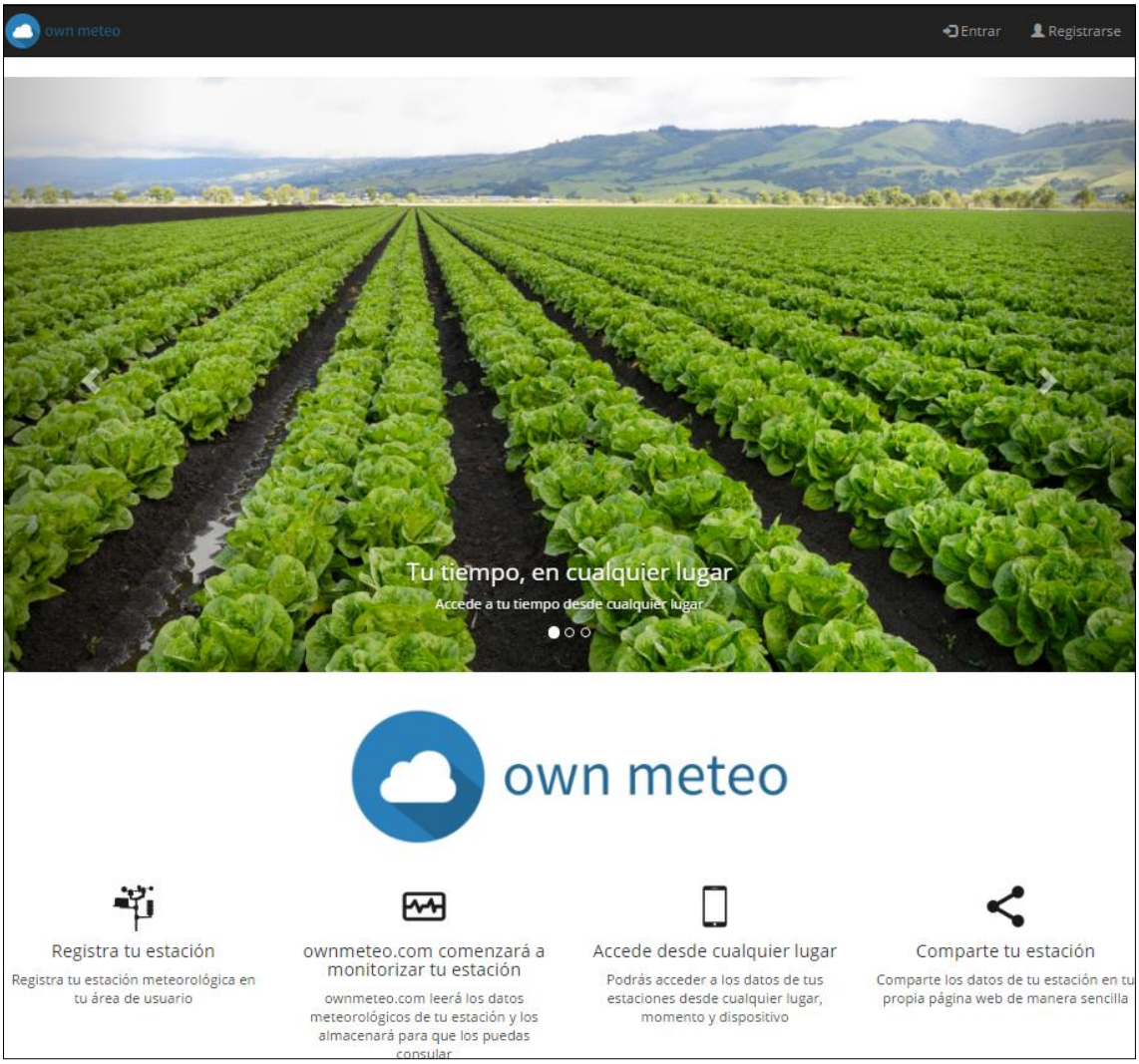

Figura 7: Captura de la portada del sitio.

Figura 8: Captura de la vista de estaciones meteorológicas de usuario.

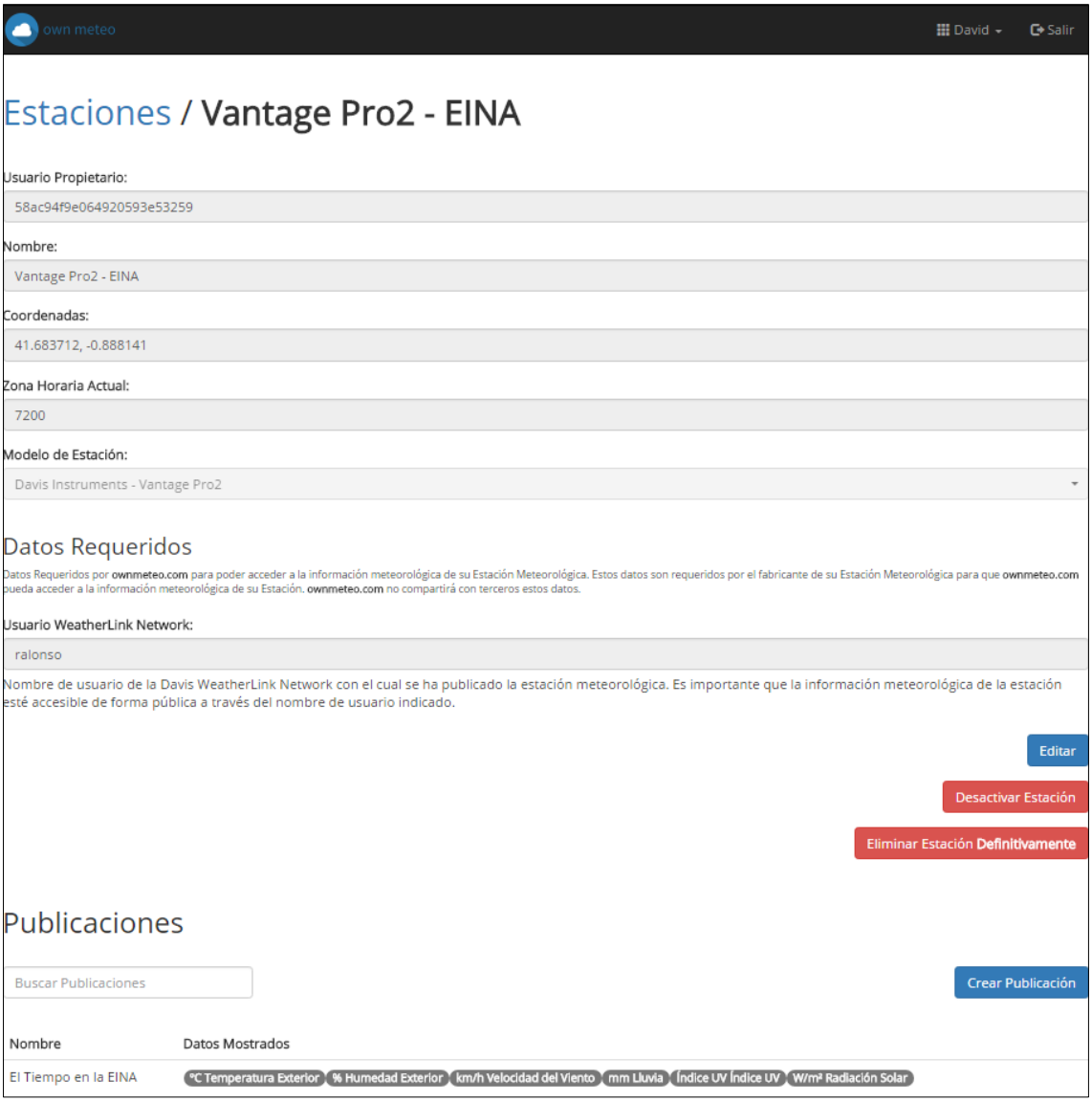

Figura 9: Captura de la vista de detalle de estación meteorológica de usuario.

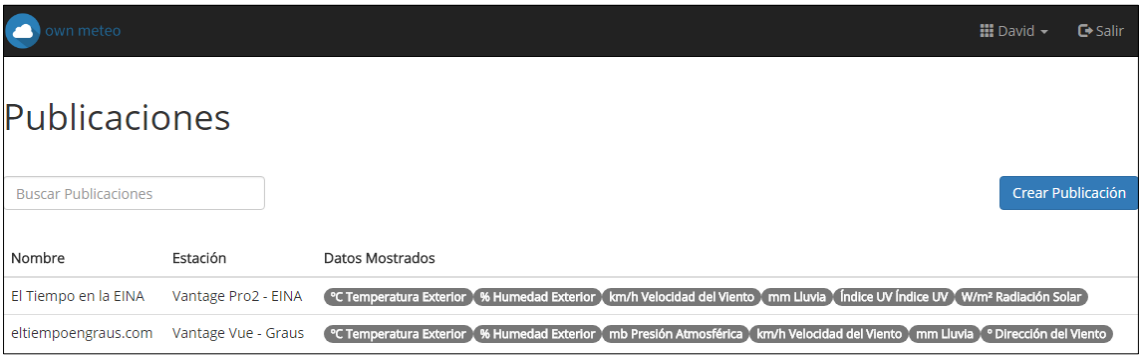

Figura 10: Captura de la vista de publicaciones de información meteorológica de usuario.

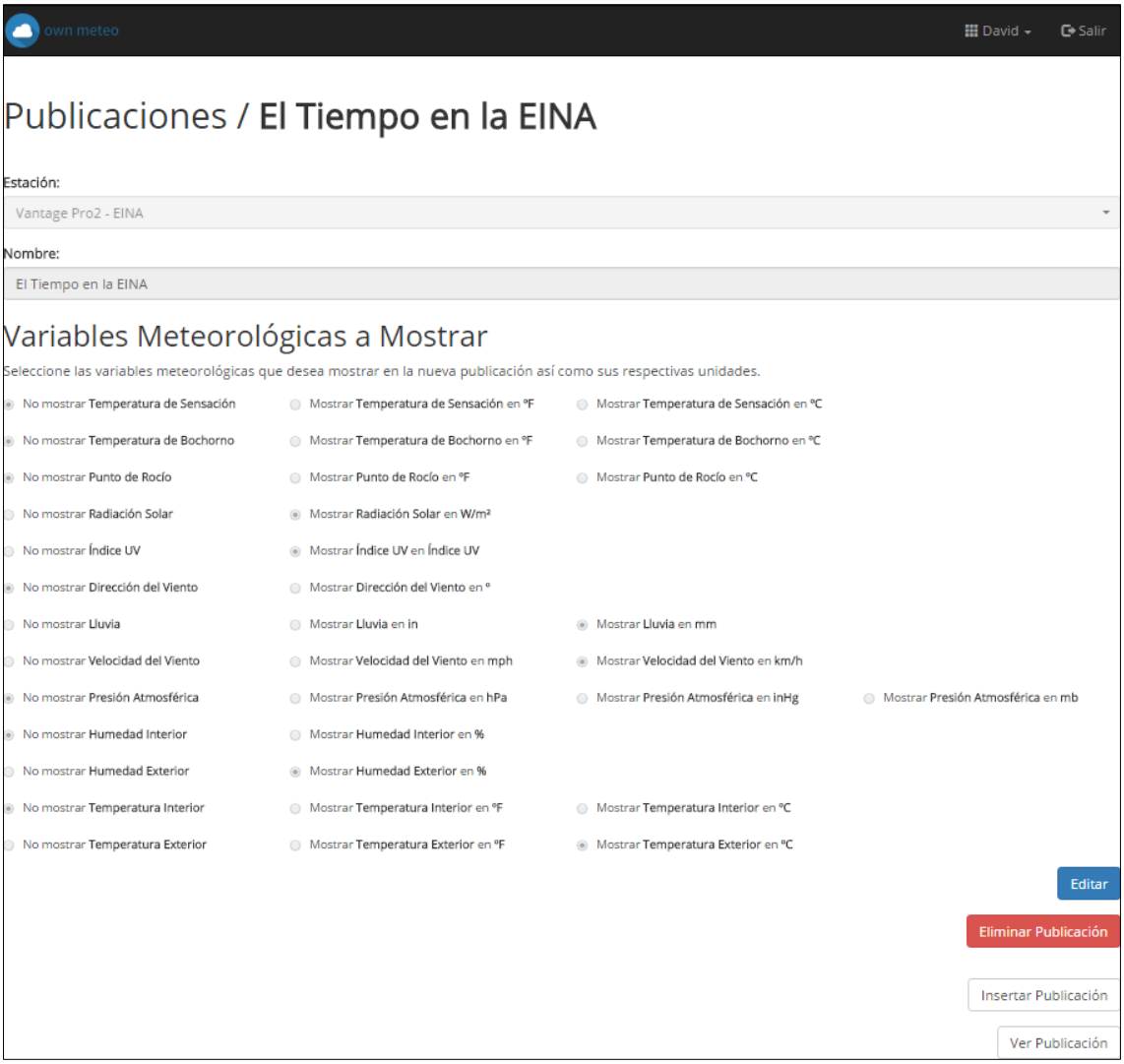

Figura 11: Captura de la vista de detalle de publicación de información meteorológica de usuario.

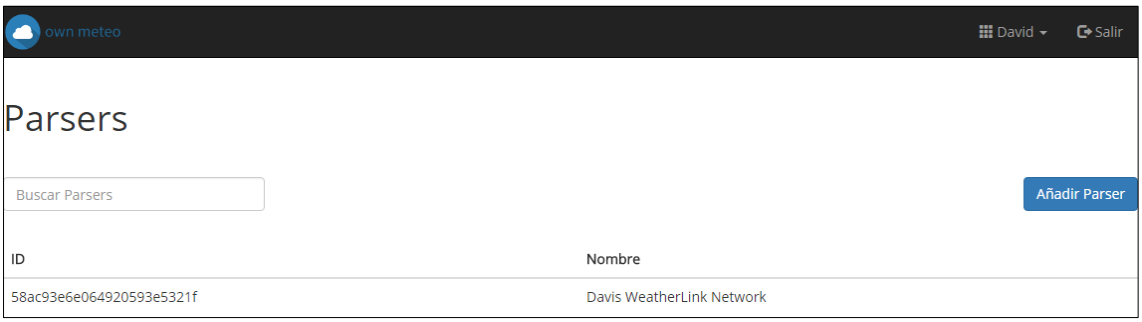

Figura 12: Captura de la vista de *parsers*.

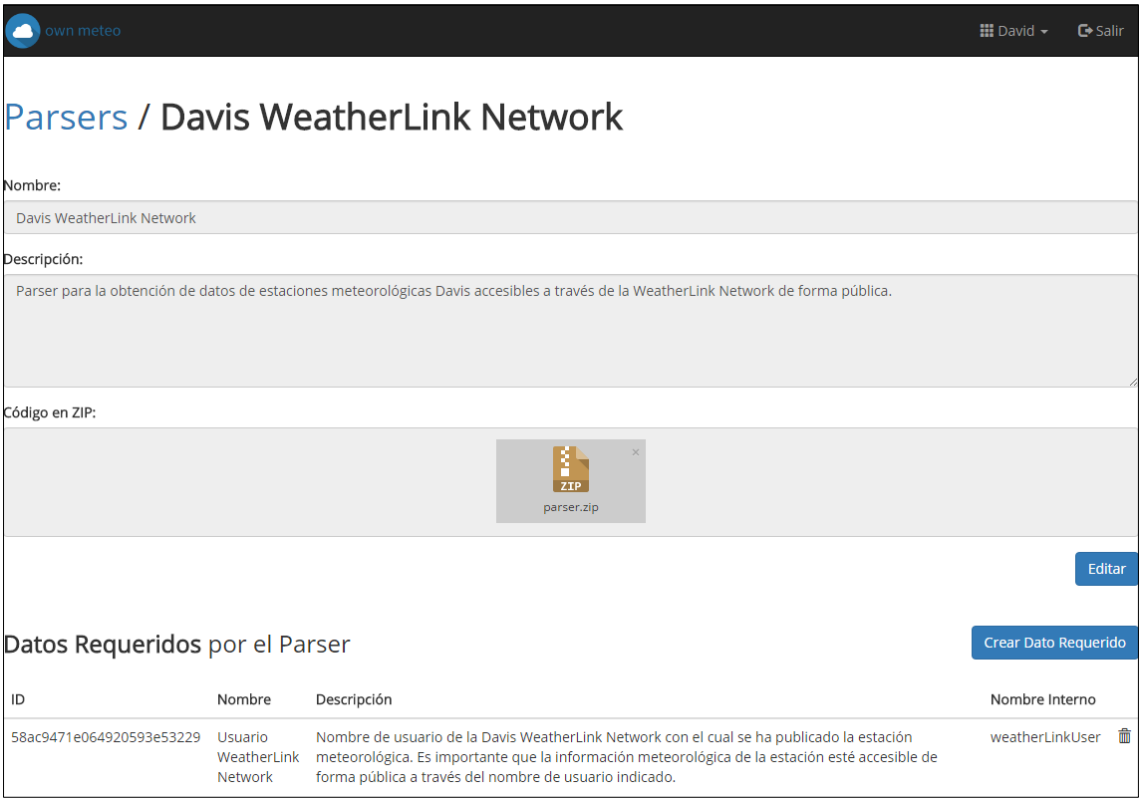

Figura 13: Captura de la vista de detalle de *parser*.

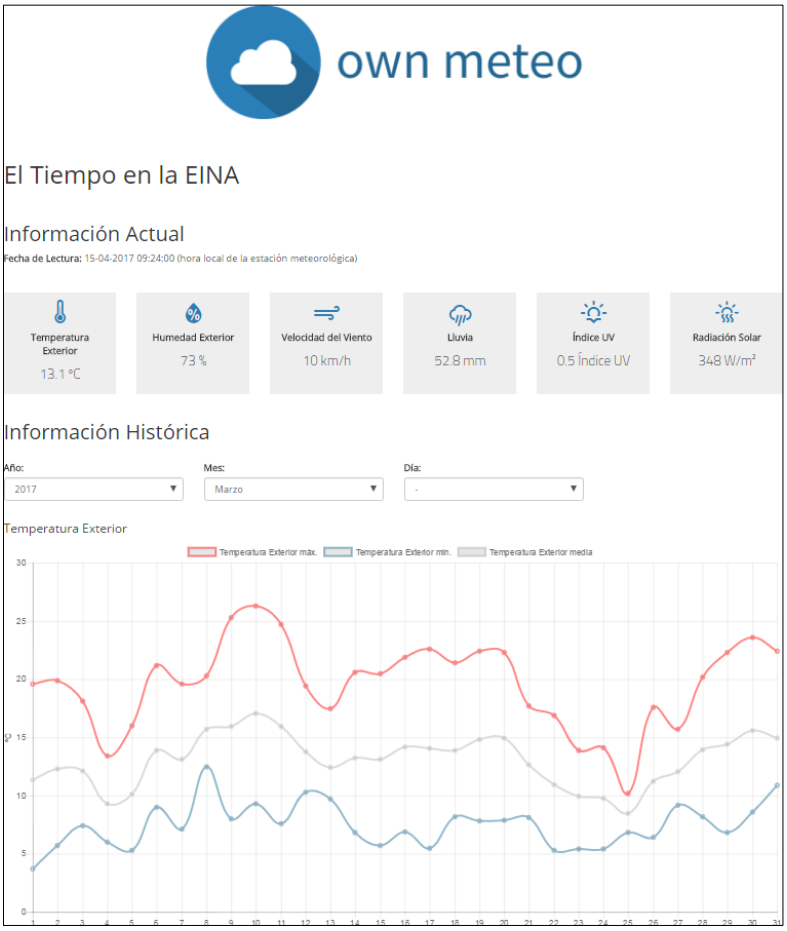

Figura 14: Captura de la vista de información meteorológica histórica.

Como se puede apreciar en las capturas anteriores, se ha optado por un diseño de la interfaz gráfica simple y poco cargado. Es importante destacar que la interfaz es adaptativa o *responsive*, de forma que se adapta al tamaño de pantalla del dispositivo donde se ejecuta.

En el Anexo B se encuentran los mapas de navegación de la aplicación, en función del nivel de autenticación del usuario (usuario o administrador). En el Anexo C se describen en detalle las vistas desarrolladas así como sus controladores.

5. Validación

Para la validación de la plataforma se ha desarrollado un juego de pruebas automatizado, de forma que certificar el correcto funcionamiento de la aplicación resulte sencillo, rápido y eficaz. A continuación se describe la herramienta de automatización de pruebas utilizada para el desarrollo del juego de pruebas automatizadas desarrollado en este proyecto:

5.1. Protractor

Para el desarrollo del juego de pruebas automatizado se ha utilizado el *framework* Protractor. Protractor [20] es una herramienta cuyo objetivo es el de probar aplicaciones web desarrolladas con Angular (más conocido como Angular 2) o AngularJS (también conocido como Angular 1, la versión utilizada en este proyecto). Para ello, Protractor ejecuta las pruebas indicadas por el desarrollador sobre un navegador web real, emulando las acciones que realizaría un usuario real.

Este *framework* se ejecuta sobre Node.js, por lo tanto las pruebas desarrolladas por el propio programador han de ser escritas en JavaScript.

Es importante destacar que Protractor se comunica con un servidor Selenium [21], el cual (normalmente) se ejecuta de forma local (en la misma máquina donde se ejecuta Protractor). El servidor Selenium es el encargado de controlar el navegador web, según las indicaciones que le haga Protractor a través del API del propio servidor Selenium. A continuación se muestra una figura que ilustra (de forma muy simplificada) este comportamiento:

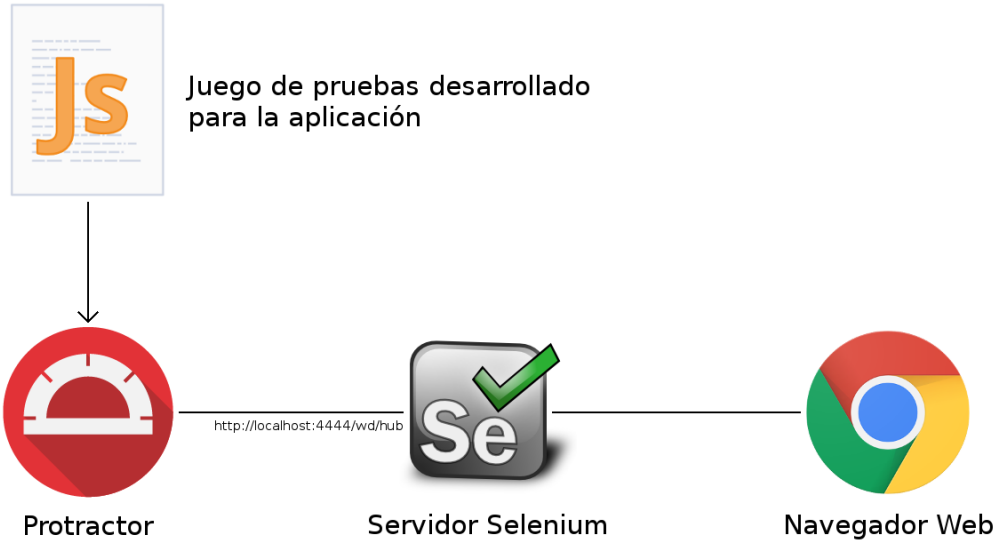

Figura 15: Funcionamiento de Protractor.

5.2. Pruebas Propuestas

Se ha tratado que las pruebas abarquen todas las vistas de la aplicación, de forma que tanto las propias vistas como el API REST sean probados en su completitud. Además, para cada vista, se han tratado de generar todos sus posibles errores así como todos sus posibles casos de éxito.

En el Anexo D se pueden encontrar las pruebas propuestas para cada vista.

5.3. Validación

Tras haber diseñado el juego de pruebas, estas fueron implementadas utilizando el entorno Protractor.

Además de ello, se implementó un pequeño programa auxiliar que facilita aún más la ejecución del juego de pruebas. Dicho programa (ejecutable en plataformas Windows y Linux) ha sido implementado con Node.js y, principalmente, ejecuta el juego de pruebas con Protractor y finalmente indica si la aplicación web las ha superado con éxito o no. Además, si el programa es invocado con la opción `-info`, muestra el juego de pruebas y acaba (no las ejecuta).

A continuación se muestra la salida del programa si las pruebas son superadas con éxito:

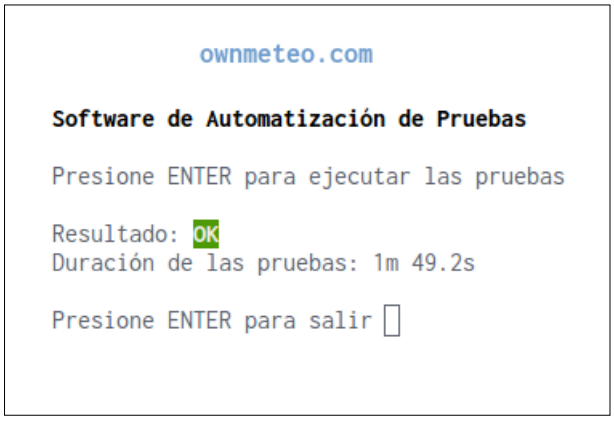

Figura 16: Resultado del programa de pruebas en caso de éxito.

A continuación se muestra la salida del programa si las pruebas no son superadas con éxito:

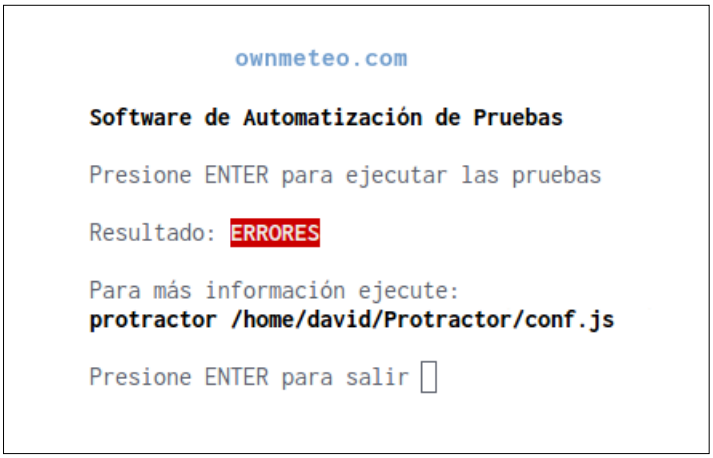

Figura 17: Resultado del programa de pruebas en caso de fracaso.

Es importante destacar que la ejecución del programa de pruebas es prácticamente inocua. Es decir, tras haber ejecutado dicho programa no habrá nuevos recursos como nuevas estaciones, nuevas publicaciones… en la plataforma, ya que los recursos creados durante las pruebas serán eliminados posteriormente por el propio programa (si no han sido eliminados ya en las propias pruebas). Los recursos existentes previamente a la ejecución de las pruebas y que son modificados durante la ejecución de las mismas también serán reestablecidos a su estado inicial.

Por último, debe mencionarse que las pruebas han sido finalmente superadas con éxito.

6. Conclusiones

En este capítulo se trata de abordar el resultado del proyecto desde un punto de vista crítico por parte de su autor. A continuación se encuentran las conclusiones tras la realización del proyecto, el trabajo propuesto para el futuro, una valoración personal y la gestión del proyecto.

6.1. Conclusiones

Este proyecto fue planteado como una herramienta capaz de integrar estaciones meteorológicas de diferentes fabricantes en una misma plataforma, de forma que esta plataforma fuera capaz de registrar los datos meteorológicos para que posteriormente fueran accesibles por los propietarios de las estaciones, otros usuarios, otras aplicaciones, etc.

Sin lugar a dudas, la plataforma desarrollada es capaz de realizar estas tareas. Como ya se ha ido mencionando a lo largo del documento, la plataforma permite a los usuarios administradores crear e integrar en la misma programas lectores de información meteorológica (*parsers*) para los diferentes modelos de estaciones, de forma que estos se encarguen de leer la información de las estaciones. Así, los usuarios administradores podrán crear e integrar nuevos *parsers* que podrán leer nuevos modelos de estación.

Desgraciadamente, para el desarrollo de un *parser* suele ser preciso disponer de (al menos) una estación meteorológica sobre la cual poder realizar las pruebas de lectura. Las estaciones meteorológicas conectadas a Internet (requisito indispensable para poder acceder a sus datos meteorológicos) no se caracterizan por ser baratas (sus precios oscilan desde un mínimo de 100€-150€ hasta miles de euros). Este hecho ha provocado que no se hayan podido desarrollar tantos *parsers* como el autor hubiese querido (además de las limitaciones temporales).

Pese a este último factor, la plataforma desarrollada es capaz de leer un conjunto de modelos de estaciones meteorológicas suficiente como para poder certificar su correcto funcionamiento.

Otro aspecto positivo a tener en cuenta de este proyecto es que está completamente funcional y accesible a través de la URL https://ownmeteo.com. Cualquier persona con acceso a Internet puede conectarse a la plataforma y comenzar a monitorizar su estación meteorológica (siempre y cuando su modelo sea reconocido). Por otra parte, los usuarios administradores también pueden conectarse a la plataforma (a través de la misma aplicación web) y gestionarla desde el panel de administración (sin necesidad de tener que manipular directamente la base de datos).

En definitiva, los objetivos planteados al inicio del proyecto se han cumplido de forma exitosa.

6.2. Trabajo Futuro

Como ya se ha comentado en el apartado anterior, la principal tarea de cara al futuro es el desarrollo de nuevos *parsers* que permitan a la plataforma reconocer nuevos modelos de estación meteorológica.

Otro aspecto a trabajar más en el futuro sería la validación de la plataforma en su conjunto (tanto aplicación web como API). Pese a que se han realizado pruebas que certifican el correcto funcionamiento de la plataforma, sería muy interesante probar la misma en condiciones extremas: pruebas de volumen (por ejemplo; pruebas con millones de registros de información meteorológica), pruebas de seguridad (por ejemplo; entradas peligrosas, cumplimiento de

restricciones…), pruebas de carga y rendimiento (por ejemplo; con miles de usuarios concurrentes), etc.

Al registrarse información meteorológica periódicamente del conjunto de estaciones registradas, el acceso a la misma puede ser costoso (computacionalmente hablando). Por esta razón, mirando al futuro, sería también muy interesante trabajar aspectos como: cachear consultas de información meteorológica, distribuir la carga computacional sobre un conjunto relativamente grande de máquinas, replicar el sistema sobre una infraestructura *cloud*, etc.

Finalmente, también sería muy interesante trabajar más la usabilidad de la plataforma. Sería apropiado, al menos, realizar test de usabilidad a usuarios externos a la plataforma para certificar que esta es sencilla e intuitiva (o en su defecto, solucionar los problemas de usabilidad).

6.3. Valoración Personal

Este proyecto ha supuesto un reto para la persona que escribe este documento. La primera dificultad consistió en diseñar un sistema capaz de tomar información de fuentes totalmente heterogéneas. No fue sencillo analizar cómo cada fabricante publicaba la información meteorológica de sus estaciones en Internet. Si bien algunos fabricantes disponen de aun API pública y bien documentada, otros todo lo contrario. En algunos casos, fue necesario incluso utilizar técnicas de ingeniería inversa (análisis de paquetes HTTP con Wireshark[4]).

Tampoco ha sido sencilla la gestión y desarrollo de un proyecto de tamaño no despreciable con un conjunto de tecnologías bastante nuevas para la persona que escribe este texto.

Que decir tiene la redacción de este documento, el cual debe estar a la altura del trabajo desarrollado durante el proyecto.

Sin embargo, estoy más que satisfecho (orgulloso) con el trabajo realizado. He aprendido a organizar el desarrollo de un proyecto de cierto tamaño, he aprendido una nueva tecnología que probablemente me abra puertas en el futuro y he mejorado mis habilidades a la hora de redactar (algo que considero bastante importante).

En definitiva, este trabajo ha hecho uso de todos los conocimientos que he ido adquiriendo estos últimos años y, sinceramente, pienso que los resultados han sido positivos.

6.4. Gestión del Proyecto

El proyecto se ha dividido en las siguientes fases: análisis y definición de requisitos, elección de la tecnología y formación, diseño, implementación, pruebas, puesta en marcha y documentación.

En la primera fase se analizó el mercado de estaciones meteorológicas y las soluciones propuestas similares a la elaborada en este proyecto. Se analizó como las estaciones meteorológicas publican sus datos en Internet y como se puede acceder a ellos. Se analizó también el funcionamiento de las plataformas de la competencia, sus carencias y sus fortalezas. Finalmente se propusieron los requisitos que debía satisfacer la plataforma desarrollada en este proyecto para destacar sobre el resto.

[4] Wireshark: analizador de paquetes de red de software libre.

La segunda fase fue la de búsqueda de las tecnologías más apropiadas para el desarrollo del proyecto. Una vez escogido el conjunto de tecnologías, se comenzó un periodo de formación.

La tercera fase consistió en el diseño de una solución con las tecnologías escogidas. En esta fase se definió la arquitectura de alto nivel de la solución, el API REST, las vistas de la aplicación web, etc.

La cuarta fase fue la de implementación del diseño propuesto en la fase anterior.

En la quinta fase del proyecto se definieron las pruebas a realizar y se implementaron. Se verificó el correcto funcionamiento del sistema y se depuraron algunos errores.

La sexta fase del proyecto fue la de puesta en producción. En esta fase se publicó de forma definitiva la aplicación en Internet.

Finalmente, en la séptima fase del proyecto se finalizó la documentación y se cerró el proyecto.

A continuación se muestra una tabla donde se puede apreciar los esfuerzos invertidos en cada fase:

Fase	Horas	% del total
1ª - Análisis y requisitos	45	11.81%
2ª - Elección de tecnologías y formación	24	6.3%
3ª - Diseño	65	17.06%
4ª - Implementación	170	44.62%
5ª - Pruebas	24	6.3%
6ª - Puesta en producción	8	2.1%
7ª - Documentación	45	11.81%
Total	381	100%

Tabla 6: Relación de esfuerzos invertidos en las diferentes fases del proyecto.

Finalmente es importante destacar que al final de cada fase se realizaron reuniones con el director del TFG con el objetivo de certificar el correcto progreso del proyecto. En estas reuniones, además de comprobar el progreso realizado, se fijaban nuevos hitos y se programaba la siguiente reunión.

7. Bibliografía

[1] Davis Instruments Corp. Accedido el 16 de Abril de 2017.
http://www.davisnet.com/

[2] *Davis WeatherLink Network*. Accedido el 16 de Abril de 2017.
http://www.weatherlink.com/

[3] Netatmo S.A. Accedido el 16 de Abril de 2017.
https://www.netatmo.com/es-ES/site/

[4] *Netatmo Connect*. Accedido el 16 de Abril de 2017.
https://www.netatmo.com/es-ES/site/connect/general

[5] Oregon Scientific Global Distribution Ltd. Accedido el 16 de Abril de 2017.
http://global.oregonscientific.com/

[6] *Oregon Scientific Anywhere Weather*. Accedido el 16 de Abril de 2017.
http://www.osanywhereweather.com/

[7] *WeatherCloud*. Accedido el 16 de Abril de 2017.
https://weathercloud.net/

[8] *Weather Underground*. Accedido el 16 de Abril de 2017.
https://www.wunderground.com/

[9] MEAN. Accedido el 16 de Abril de 2017.
http://mean.io/

[10] JavaScript, Wikipedia. Accedido el 16 de Abril de 2017.
https://es.wikipedia.org/wiki/JavaScript

[11] Node.js. Accedido el 16 de Abril de 2017.
https://nodejs.org/en/

[12] Express. Accedido el 16 de Abril de 2017.
http://expressjs.com/

[13] MongoDB. Accedido el 16 de Abril de 2017.
https://www.mongodb.com/

[14] AngularJS. Accedido el 16 de Abril de 2017.
https://angularjs.org/

[15] Transferencia de Estado Representacional, Wikipedia. Accedido el 16 de Abril de 2017.
https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional

[16] *Hypertext Transfer Protocol*, Wikipedia. Accedido el 16 de Abril de 2017.
https://es.wikipedia.org/wiki/Hypertext_Transfer_Protocol

[17] *Cross-origin resource sharing*, Wikipedia. Accedido el 16 de Abril de 2017.
https://en.wikipedia.org/wiki/Cross-origin_resource_sharing

[18] NGINX. Accedido el 16 de Abril de 2017.
https://nginx.org/

[19] *Hypertext Transfer Protocol Secure*, Wikipedia. Accedido el 16 de Abril de 2017. https://es.wikipedia.org/wiki/Hypertext_Transfer_Protocol_Secure

[20] Protractor. Accedido el 16 de Abril de 2017. http://www.protractortest.org/#/

[21] Selenium. Accedido el 16 de Abril de 2017. http://www.seleniumhq.org/

Anexo A - Operaciones API REST

En este Anexo se describen las operaciones del API REST en función de la URI (*endpoint*) y el método de petición (o verbo) HTTP. Las operaciones se presentan clasificadas según el tipo de recurso que manipulan (por ejemplo: usuarios).

A.1. Usuarios

GET /api/users

Obtiene el listado de usuarios registrados en la plataforma. A continuación se muestra el cuerpo (*payload*) de una respuesta de ejemplo para esta operación:


```
[
  {
    "_id": "58ac8f470aa073210bda2437",
    "createdAt": "2017-02-21T19:04:39.704Z",
    "enabled": true,
    "password": "c7ad44cbad762a5da0a...452f9e854fdc1e07ec",
    "email": "admin@ownmeteo.com",
    "lastName": "Enjuanes",
    "name": "David",
    "__v": 0,
    "lastAccess": "2017-03-16T18:15:24.560Z",
    "role": [
      "admin"
    ],
    "uri": "https://ownmeteo.com/api/users/58ac8f470aa073210bda2437"
  },
  {
    "_id": "58ac94f9e064920593e53259",
    "createdAt": "2017-02-21T19:28:57.041Z",
    "enabled": true,
    "password": "a917d01789b58dfd3a7...715496269886f4929",
    "email": "...@hotmail.com",
    "lastName": "Enjuanes",
    "name": "David",
    "__v": 0,
    "lastAccess": "2017-03-12T09:20:24.408Z",
    "role": [
      "user"
    ],
    "uri": "https://ownmeteo.com/api/users/58ac94f9e064920593e53259"
  }
]
```


Es importante destacar que esta operación solo la pueden realizar los clientes autenticados en la plataforma como usuarios administradores.

POST /api/users

Registra un nuevo usuario en el sistema. A continuación se muestra un ejemplo de petición y respuesta del API para esta operación:

Petición:

```
POST /api/users HTTP/1.1
Content-Type: application/json;charset=UTF-8
...

{
  "enabled":true,
  "role":["user"],
  "email":"nuevo@example.com",
  "name":"Ejemplo",
  "lastName":"Ejemplo Ejemplo",
  "password":"ejemplo"
}
```

Respuesta:

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=utf-8
...

{
  "error": false,
  "status": "created",
  "uri": "https://ownmeteo.com/api/users/58cbb678b9fa0f04f8381576"
}
```

Para realizar esta operación, el cliente no necesita estar autenticado a no ser que el usuario a crear tenga derechos de administración, en cuyo caso el cliente que realice la petición deberá estar autenticado como usuario administrador.

A continuación se muestra un diagrama de secuencia que muestra en detalle las actuaciones del API cuando se invoca esta operación (se asume que el cliente que realiza la operación está autenticado como usuario administrador para mayor simplicidad):

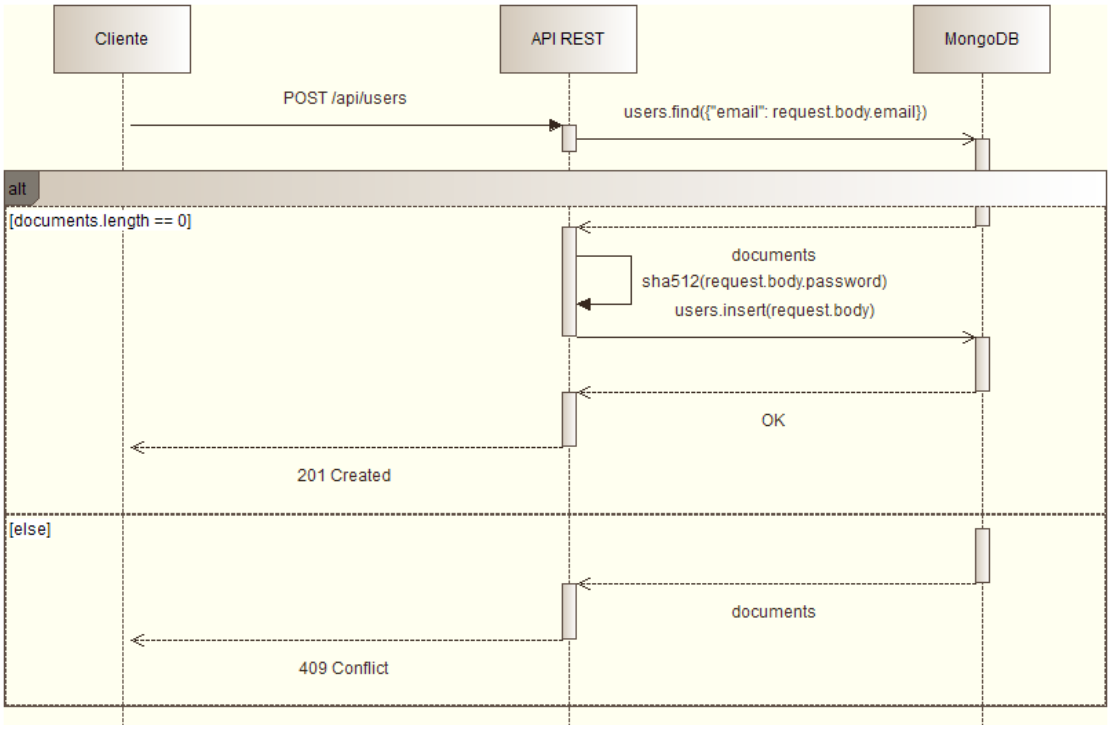

Figura 18: Diagrama de secuencia de creación de usuario.

GET /api/users/:id

Obtiene la información del usuario cuyo ID sea el indicado en la URI de la petición. A continuación se muestra el cuerpo de una respuesta de ejemplo para esta operación:

```
{
  "_id": "58ac8f470aa073210bda2437",
  "createdAt": "2017-02-21T19:04:39.704Z",
  "enabled": true,
  "password": "c7ad44cbad762a5da0a...452f9e854fdc1e07ec",
  "email": "admin@ownmeteo.com",
  "lastName": "Enjuanes",
  "name": "David",
  "__v": 0,
  "lastAccess": "2017-03-16T18:15:24.560Z",
  "role": [
    "admin"
  ],
  "uri": "https://ownmeteo.com/api/users"
}
```

Esta operación requiere que el cliente esté autenticado. Si el cliente está realizando una petición de información de su propio usuario, bastará con que el cliente esté autenticado como usuario normal (no administrador). Si por el contrario el cliente está realizando una petición de información que no se corresponde con su propio usuario, esta operación requerirá que el cliente esté autenticado como usuario administrador.

PUT /api/users/:id

Modifica la información del usuario cuyo ID sea el indicado en la URI de la petición. La nueva información deberá viajar en el cuerpo de la petición. A continuación se muestra un ejemplo de petición y respuesta para esta operación:

Petición:

```
PUT /api/users/58ac8f470aa073210bda2437 HTTP/1.1
Content-Type: application/json;charset=UTF-8
...

{
  "enabled": true,
  "password": "c7ad44cbad762a5da0a...538dc69dd8de9077ec",
  "email": "admin@ownmeteo.com",
  "lastName": "Enjuanes Gómez",
  "name": "David",
  "role": [
    "admin"
  ]
}
```

Respuesta:

```
HTTP/1.1 200 OK
...

{
  "error": false,
  "status": "updated",
  "uri": "https://ownmeteo.com/api/users/58ac8f470aa073210bda2437"
}
```

Al igual que la operación anterior, si el cliente desea modificar su propia información de usuario, únicamente deberá estar autenticado como usuario no administrador. Si por el contrario, el

cliente desea modificar información de un usuario que no es el suyo propio deberá estar autenticado como usuario administrador. Además de esto, la modificación del rol de usuario solo podrá ser llevada a cabo por usuarios administradores (es decir, los usuarios no administradores no podrán cambiar el rol de su usuario).

A continuación se muestra un diagrama de secuencia que muestra en detalle las actuaciones del API cuando se invoca esta operación (se asume que el cliente que realiza la operación está autenticado como usuario administrador para mayor simplicidad):

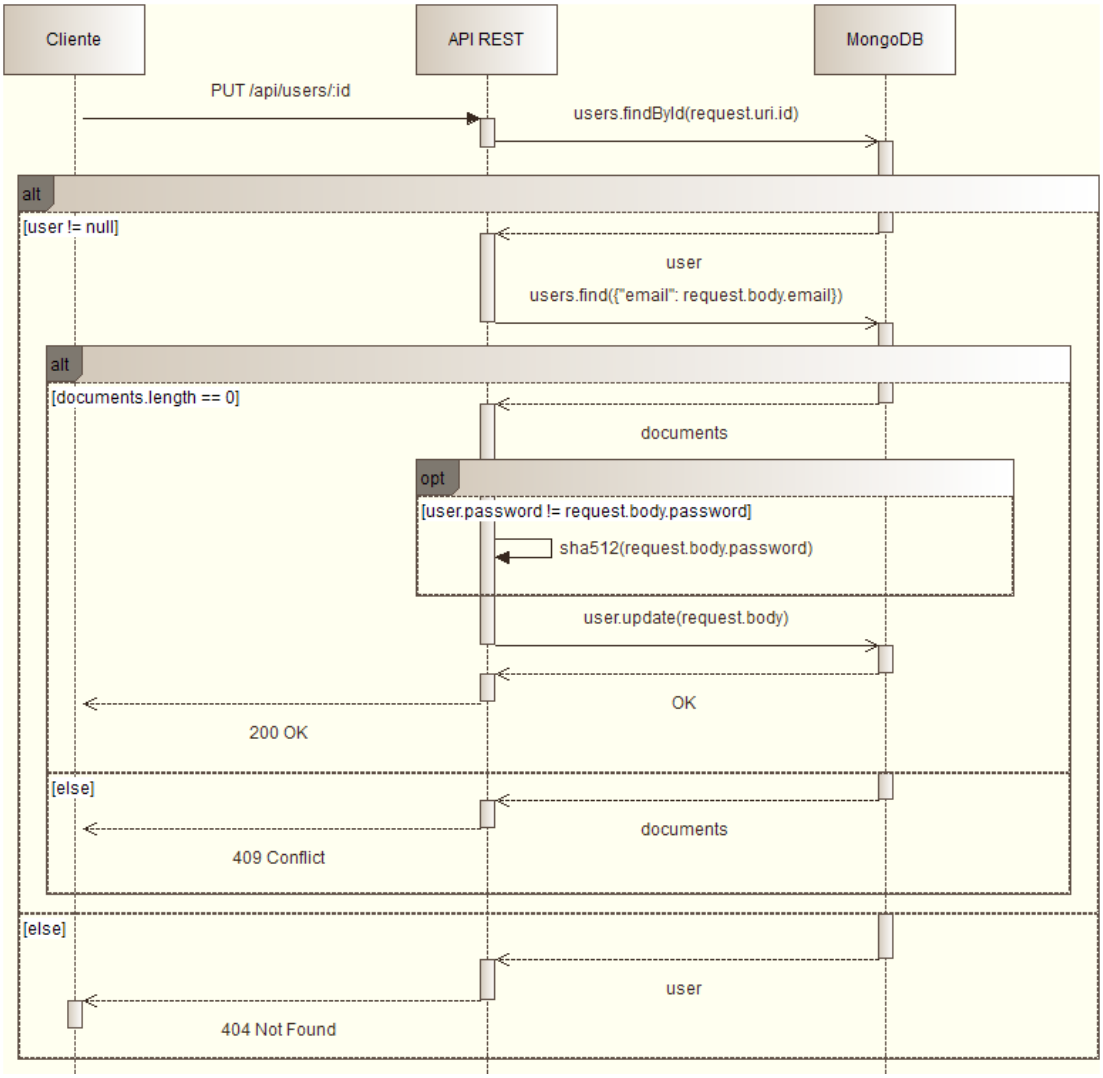

Figura 19: Diagrama de secuencia de modificación de información de usuario.

DELETE /api/users/:id

Elimina el usuario cuyo ID sea el indicado en la URI de la petición. Esta operación solo puede ser llevada a cabo por usuarios administradores. Es importante destacar que esta operación elimina por completo la información del usuario de la base de datos.

Esta operación no debe ser confundida con una desactivación de usuario (la cual se lleva a cabo a través del método de petición PUT).

GET /api/users/:id/sessions

Obtiene las sesiones del usuario cuyo ID sea el indicado en la URI de la petición. A continuación se muestra el cuerpo de una respuesta de ejemplo para esta operación:


```
[
  {
    "_id": "58ac9646e064920593e532eb",
    "expiresAt": "2018-02-21T19:34:30.521Z",
    "createdAt": "2017-02-21T19:34:30.521Z",
    "token": "05d6431a0a024abbcd...405ca8694fd7072b649",
    "user": "58ac8f470aa073210bda2437",
    "__v": 0,
    "uri": "https://ownmeteo.com/api/sessions/58ac9646e064920593e532eb"
  },
  {
    "_id": "58ad401ae064920593e5334c",
    "expiresAt": "2018-02-22T07:39:06.720Z",
    "createdAt": "2017-02-22T07:39:06.720Z",
    "token": "2947cb142c359067e7...73122f8a874d74fe791",
    "user": "58ac8f470aa073210bda2437",
    "__v": 0,
    "uri": "https://ownmeteo.com/api/sessions/58ad401ae064920593e5334c"
  }
]
```


Un cliente autenticado como usuario no administrador solo podrá consultar las sesiones de su propio usuario. Un usuario administrador podrá consultar las sesiones de cualquier usuario.

GET /api/users/:id/stations

Obtiene las estaciones meteorológicas del usuario cuyo ID sea el indicado en la URI de la petición. A continuación se muestra el cuerpo de una respuesta de ejemplo para esta operación:


```
[
  {
    "_id": "58ac953ce064920593e53271",
    "parserRequiredData": {
      "weatherLinkUser": "enjuanes"
    },
    "createdAt": "2017-02-21T19:30:04.284Z",
    "enabled": true,
    "currentTimezone": 3600,
    "location": "42.192117,0.342791",
    "model": "58ac948ae064920593e53239",
    "name": "Vantage Vue - Graus",
    "userOwner": "58ac94f9e064920593e53259",
    "__v": 0,
    "uri": "https://ownmeteo.com/api/stations/58ac953ce064920593e53271"
  },
  {
    "_id": "58b4476c23437204bc98cb24",
    "parserRequiredData": {
      "weatherLinkUser": "ralonso"
    },
    "createdAt": "2017-02-27T15:36:12.356Z",
    "enabled": true,
    "currentTimezone": 3600,
    "location": "41.683712,-0.888141",
    "model": "58ac94c7e064920593e53246",
    "name": "Vantage Pro2 - EINA",
    "userOwner": "58ac94f9e064920593e53259",
    "__v": 0,
    "uri": "https://ownmeteo.com/api/stations/58b4476c23437204bc98cb24"
  }
]
```

Un cliente autenticado como usuario no administrador solo podrá consultar las estaciones meteorológicas de su propio usuario, mientras que un usuario administrador podrá consultar las estaciones de cualquier usuario.

A.2. *Tokens* de Sesión

GET /api/sessions

Devuelve el listado de sesiones almacenadas en el sistema. A continuación se muestra el cuerpo de una respuesta de ejemplo para esta operación:

```
[
  {
    "_id":"58b455d4029fd704ee7ff1d2",
    "expiresAt":"2018-02-27T16:37:40.376Z",
    "createdAt":"2017-02-27T16:37:40.376Z",
    "token": "05d6431a0a024abbcd...405ca8694fd7072b649",
    "user":"58ac94f9e064920593e53259",
    "__v":0,
    "uri":"https://ownmeteo.com/api/sessions/58b455d4029fd704ee7ff1d2"
  },
  {
    "_id":"58bfdd00029fd704ee807dac",
    "expiresAt":"2018-03-08T10:29:20.971Z",
    "createdAt":"2017-03-08T10:29:20.971Z",
    "token": "2947cb142c359067e7...73122f8a874d74fe791",
    "user":"58ac94f9e064920593e53259",
    "__v":0,
    "uri":"https://ownmeteo.com/api/sessions/58bfdd00029fd704ee807dac"
  }
]
```

Esta operación solo puede ser ejecutada por un cliente autenticado como administrador.

POST /api/sessions

Crea una nueva sesión a partir de los credenciales de usuario pasados en el cuerpo de la petición (i.e. correo electrónico y contraseña). A continuación se muestra un ejemplo de petición y respuesta para esta operación:

Petición:

```
POST /api/sessions HTTP/1.1
Content-Type: application/json;charset=UTF-8
...

{
  "email": "...@hotmail.com",
  "password": "david"
}
```

Respuesta:

```
HTTP/1.1 201 Created
Set-Cookie: sessionid=775243e8...70c18f55; Max-Age=31536000; Path=/; Expires=Mon, 19 Mar
2018 11:20:36 GMT; HttpOnly
...

{
  "error":false,
  "status":"created",
  "_id":"58ce6984b9fa0f04f8383a70",
  "token":"775243e8...70c18f55",
```

```
  "uri":"https://ownmeteo.com/api/sessions/58ce6984b9fa0f04f8383a70"
}
```

Es importante destacar que la cabecera HTTP *Set-Cookie* mostrada en la respuesta anterior no contiene saltos de línea; se han añadido en este documento para que se pueda apreciar correctamente.

A continuación se muestra un diagrama de secuencia que muestra en detalle las actuaciones del API cuando se invoca esta operación:

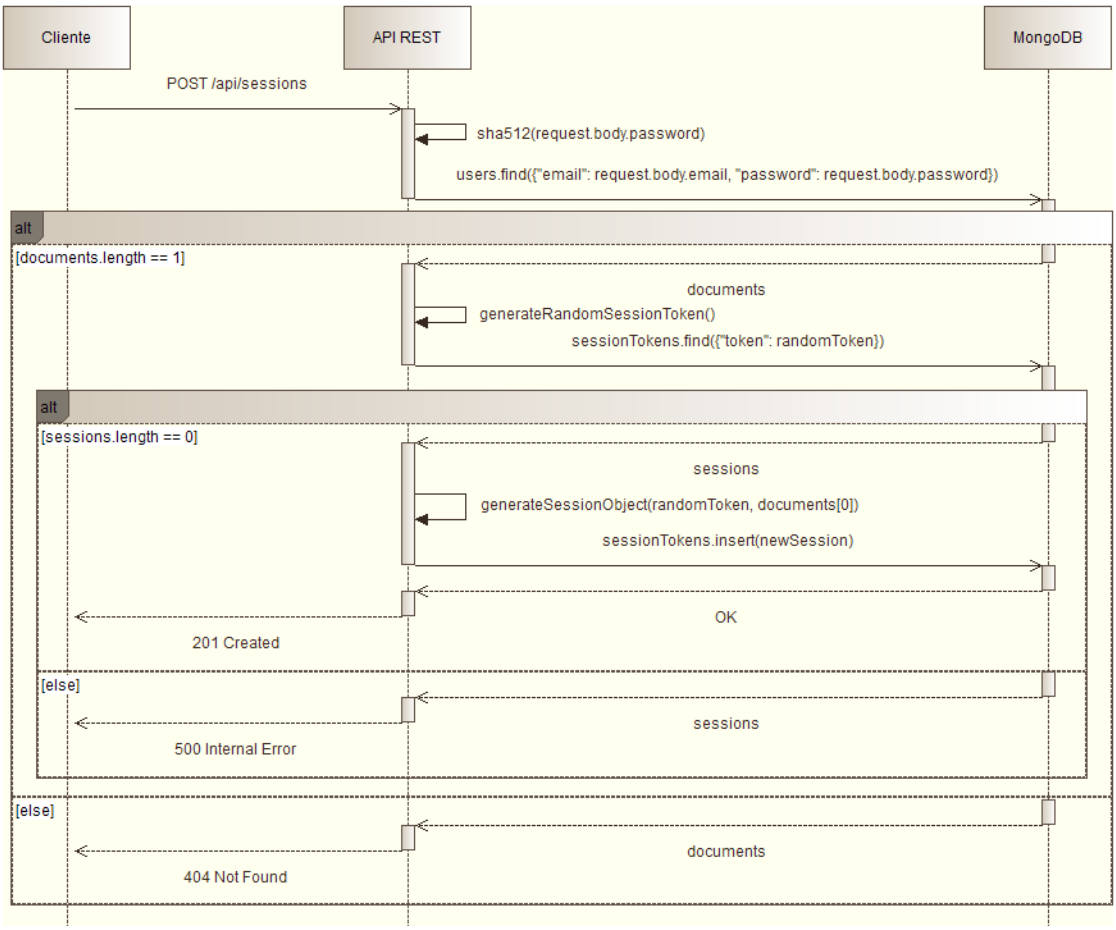

Figura 20: Diagrama de secuencia de creación de sesión.

GET /api/sessions/:id

Obtiene la información de la sesión cuyo ID sea el indicado en la URI de la petición. A continuación se muestra el cuerpo de una respuesta de ejemplo para esta operación:

```
{
  "_id":"58bfdd00029fd704ee807dac",
  "expiresAt":"2018-03-08T10:29:20.971Z",
  "createdAt":"2017-03-08T10:29:20.971Z",
  "token": "05d6431a0a024abbcd...405ca8694fd7072b649",
  "user":"58ac94f9e064920593e53259",
  "__v":0,
  "uri":"https://ownmeteo.com/api/sessions/58bfdd00029fd704ee807dac"
}
```

Si el cliente está autenticado como usuario no administrador, únicamente podrá obtener información de sesiones de su propiedad. Por el contrario, si el cliente está autenticado como usuario administrador podrá obtener información de cualquier sesión almacenada en el sistema.

DELETE /api/sessions/:id

Elimina por completo la sesión cuyo ID sea el indicado en la URI de la petición. Al igual que la operación anterior, si el cliente está autenticado como usuario no administrador, únicamente podrá eliminar sesiones de su propiedad. Por el contrario, si el cliente está autenticado como usuario administrador podrá eliminar cualquier sesión almacenada en el sistema.

GET /api/sessions/current

Obtiene la información de la sesión con la cual se ha invocado la propia operación.

DELETE /api/sessions/current

Elimina por completo la sesión con la cual se ha invocado la propia operación.

A.3. *Parsers*

GET /api/parsers

Obtiene la información de todos los *parsers* del sistema. A continuación se muestra el cuerpo de una respuesta de ejemplo para esta operación:

```
[
  {
    "_id": "58ac93e6e064920593e5321f",
    "createdAt": "2017-02-21T19:24:22.081Z",
    "description": "Parser para la obtención de datos de estaciones meteorológicas Davis
accesibles a través de la WeatherLink Network de forma pública.",
    "name": "Davis WeatherLink Network",
    "__v": 0,
    "uri": "https://ownmeteo.com/api/parsers/58ac93e6e064920593e5321f"
  }
]
```

Esta operación solo puede ser utilizada por clientes autenticados como usuarios administradores.

POST /api/parsers

Crea un nuevo *parser* a partir de los datos pasados en el cuerpo de la petición. El código fuente del *parser* deberá viajar en la petición como una carpeta comprimida en ZIP y codificada en Base 64. A continuación se muestra un ejemplo de petición y respuesta para esta operación:

Petición:

```
POST /api/parsers HTTP/1.1
Content-Type: application/json;charset=UTF-8
...

{
  "name":"Nuevo Parser",
  "description":"Ejemplo de Parser.",
  "zippedCode":"UEsDB...zdR4AAAA="
}
```

Respuesta:

```
HTTP/1.1 201 Created
...

{
  "error":false,
  "status":"created",
  "uri":"https://ownmeteo.com/api/parsers/58ce8cd7b9fa0f04f8383c74"
}
```

Esta operación solo puede ser invocada por clientes autenticados como usuario administrador.

A continuación se muestra un diagrama de secuencia que muestra en detalle las actuaciones del API cuando se invoca esta operación:

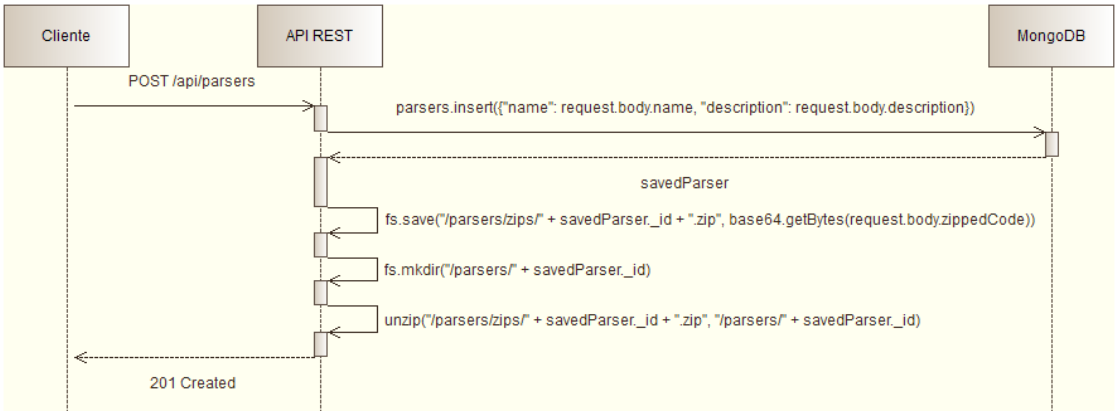

Figura 21: Diagrama de secuencia de creación de *parser*.

GET /api/parsers/:id

Obtiene la información del *parser* cuyo ID sea el indicado en la URI de la petición, así como el código fuente del mismo (comprimido en ZIP y codificado en Base 64). A continuación se muestra el cuerpo de una respuesta de ejemplo para esta operación:

```
{
  "_id": "58ac93e6e064920593e5321f",
  "createdAt": "2017-02-21T19:24:22.081Z",
  "description": "Parser para la obtención ... WeatherLink Network de forma pública.",
  "name": "Davis WeatherLink Network",
  "__v": 0,
  "uri": "https://ownmeteo.com/api/parsers/58ac93e6e064920593e5321f",
  "zippedCode": "UEsDBAoAAAA...B4AAAA="
}
```

Esta operación solo puede ser invocada por clientes autenticados como usuario administrador.

PUT /api/parsers/:id

Modifica la información del *parser* cuyo ID sea el indicado en la URI de la petición. La nueva información deberá viajar en el cuerpo de la petición. Es importante destacar que, en esta operación, no es obligatorio adjuntar el código fuente del *parser*, de manera que si no se adjunta, el código original no será modificado. Esto se ha decidido así para minimizar tiempos de transferencia a través de la red. A continuación se muestra un ejemplo de petición y respuesta para esta operación:

Petición:

```
PUT /api/parsers/58ce8cd7b9fa0f04f8383c74 HTTP/1.1
Connection: keep-alive
Content-Type: application/json;charset=UTF-8
...

{
  "name":"Nuevo nombre del Parser",
  "description":"Ejemplo de nueva descripción del Parser."
}
```

Respuesta:

```
HTTP/1.1 200 OK
Connection:keep-alive
Content-Type:application/json; charset=utf-8
...

{
  "error":false,
  "status":"updated",
  "uri":"https://ownmeteo.com/api/parsers/58ce8cd7b9fa0f04f8383c74"
}
```

Esta operación solo puede ser invocada por clientes autenticados como usuario administrador.

A continuación se puede apreciar el diagrama de secuencia correspondiente a esta operación.

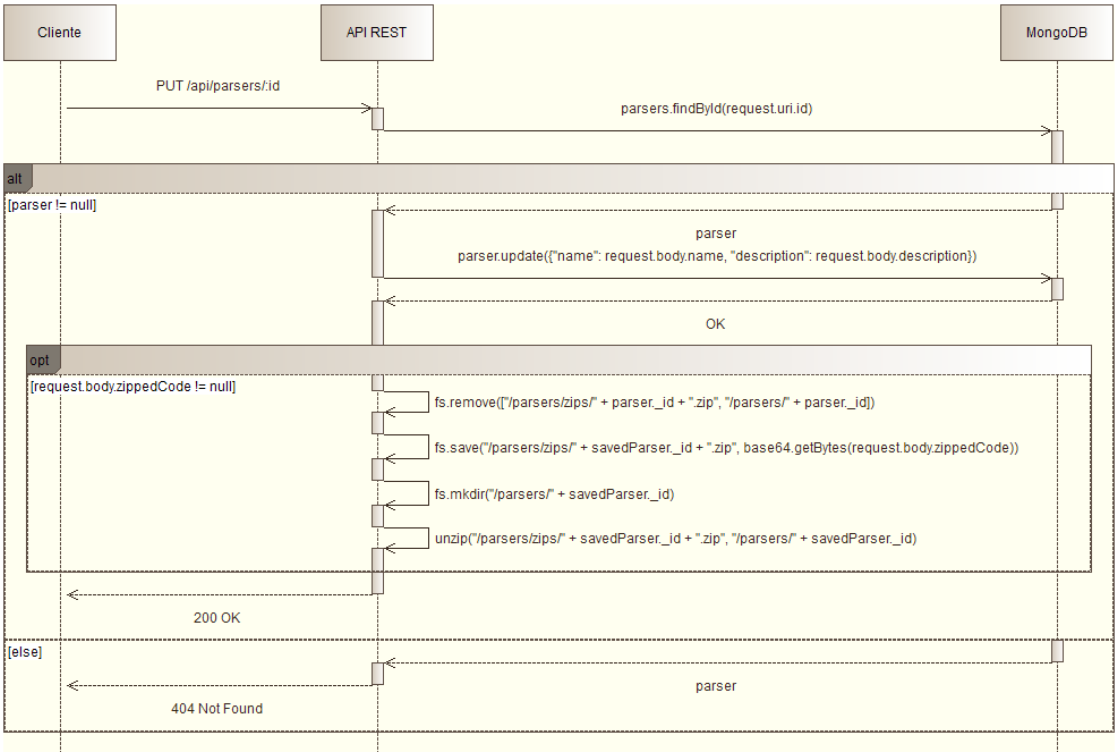

Figura 22: Diagrama de secuencia de modificación de *parser*.

DELETE /api/parsers/:id

Elimina el *parser* cuyo ID sea el indicado en la URI de la petición. Esta operación elimina los datos del *parser* de la base de datos MongoDB así como el código fuente descomprimido, sin embargo no elimina el código comprimido en ZIP. Esto se ha decidido así por motivos de seguridad, de forma que siempre se dispondrá de versiones antiguas de un *parser*.

Es importante destacar que esta operación solo puede ser invocada por clientes autenticados como usuarios administradores.

GET /api/parsers/:id/parser-required-data

Obtiene el listado de datos requeridos por el *parser* cuyo ID sea el indicado en la URI de la petición. A continuación se muestra el cuerpo de una respuesta de ejemplo para esta operación:


```
[
  {
    "_id": "58ac9471e064920593e53229",
    "parser": "58ac93e6e064920593e5321f",
    "description": "Nombre de usuario de la Davis ... de usuario indicado.",
    "name": "Usuario WeatherLink Network",
    "internalName": "weatherLinkUser",
    "__v": 0,
    "uri": "https://ownmeteo.com/api/parser-required-data/58ac9471e064920593e53229"
  }
]
```


Esta operación solo puede ser invocada por clientes autenticados como usuario administrador.

POST /api/parsers/:id/parser-required-data

Crea un nuevo dato requerido por el *parser* cuyo ID sea el indicado en la URI de la petición. A continuación se muestra un ejemplo de petición y respuesta para esta operación:

Petición:

```
POST /api/parsers/58ce8cd7b9fa0f04f8383c74/parser-required-data HTTP/1.1
...

{
  "name":"Nombre del Dato Requerido",
  "description":"Descripción del Dato Requerido.",
  "internalName": "internalNameExample"
}
```

Respuesta:


```
HTTP/1.1 201 Created
...

{
  "error": false,
  "status": "created",
  "uri": "https://ownmeteo.com/api/parser-required-data/58da708fb9fa0f04f838def3"
}
```


Esta operación solo puede ser invocada por clientes autenticados como usuario administrador.

GET /api/parsers/:id/station-models

Obtiene el listado de modelos de estación meteorológica cuyo *parser* para obtener la información meteorológica es el indicado en la URI de la petición. A continuación se muestra el cuerpo de una respuesta de ejemplo para esta operación:


```
[
  {
    "_id": "58ac948ae064920593e53239",
    "createdAt": "2017-02-21T19:27:06.472Z",
    "parser": "58ac93e6e064920593e5321f",
    "productNumber": "6250",
```

```
    "name": "Vantage Vue",
    "manufacturer": "Davis Instrumens",
    "__v": 0,
    "uri": "https://ownmeteo.com/api/station-models/58ac948ae064920593e53239"
  },
  {
    "_id": "58ac94c7e064920593e53246",
    "createdAt": "2017-02-21T19:28:07.247Z",
    "parser": "58ac93e6e064920593e5321f",
    "productNumber": "6152",
    "name": "Vantage Pro2",
    "manufacturer": "Davis Instruments",
    "__v": 0,
    "uri": "https://ownmeteo.com/api/station-models/58ac94c7e064920593e53246"
  }
]
```


Esta operación solo puede ser invocada por clientes autenticados como usuario administrador.

A.4. Datos Requeridos por los *Parser*

GET /api/parser-required-data

Obtiene el listado de datos requeridos por *parser* registrados en el sistema. A continuación se muestra el cuerpo de una respuesta de ejemplo para esta operación:


```
[
  {
    "_id": "58ac9471e064920593e53229",
    "parser": "58ac93e6e064920593e5321f",
    "description": "Nombre de usuario de la ...",
    "name": "Usuario WeatherLink Network",
    "internalName": "weatherLinkUser",
    "__v": 0,
    "uri": "https://ownmeteo.com/api/parser-required-data/58ac9471e064920593e53229"
  },
  {
    "_id": "58da708fb9fa0f04f838def3",
    "parser": "58ce89a2b9fa0f04f8383c3c",
    "description": "Descripción del Dato Requerido.",
    "name": "Nombre del Dato Requerido",
    "internalName": "internalNameExample",
    "__v": 0,
    "uri": "https://ownmeteo.com/api/parser-required-data/58da708fb9fa0f04f838def3"
  }
]
```


Esta operación solo puede ser invocada por clientes autenticados como usuario administrador.

GET /api/parser-required-data/:id

Obtiene la información del dato requerido por *parser* cuyo ID sea el indicado en la URI de la petición. A continuación se muestra el cuerpo de una respuesta de ejemplo para esta operación:


```
{
  "_id": "58ac9471e064920593e53229",
  "parser": "58ac93e6e064920593e5321f",
  "description": " Nombre de usuario de la ...",
  "name": "Usuario WeatherLink Network",
  "internalName": "weatherLinkUser",
  "__v": 0,
  "uri": "https://ownmeteo.com/api/parser-required-data"
}
```

Esta operación puede ser invocada por cualquier cliente autenticado como usuario (ya sea como usuario administrador o no) en el sistema.

DELETE /api/parser-required-data/:id

Elimina el dato requerido por *parser* cuyo ID sea el indicado en la URI de la petición. Esta operación solo puede ser llevada a cabo por clientes autenticados como usuarios administradores.

A.5. Modelos de Estación

GET /api/station-models

Obtiene el listado de modelos de estación meteorológica registrados en la plataforma. A continuación se muestra el cuerpo de una respuesta de ejemplo para esta operación:

```
[
  {
    "_id": "58ac948ae064920593e53239",
    "createdAt": "2017-02-21T19:27:06.472Z",
    "parser": "58ac93e6e064920593e5321f",
    "productNumber": "6250",
    "name": "Vantage Vue",
    "manufacturer": "Davis Instrumens",
    "__v": 0,
    "uri": "https://ownmeteo.com/api/station-models/58ac948ae064920593e53239"
  },
  {
    "_id": "58ac94c7e064920593e53246",
    "createdAt": "2017-02-21T19:28:07.247Z",
    "parser": "58ac93e6e064920593e5321f",
    "productNumber": "6152",
    "name": "Vantage Pro2",
    "manufacturer": "Davis Instruments",
    "__v": 0,
    "uri": "https://ownmeteo.com/api/station-models/58ac94c7e064920593e53246"
  }
]
```

Esta operación puede ser invocada por cualquier cliente autenticado en la plataforma, ya sea como usuario administrador o no.

POST /api/station-models

Crea un nuevo modelo de estación meteorológica a partir de los datos pasados en el cuerpo de la petición. A continuación se muestra un ejemplo de petición y respuesta para esta operación:

Petición:

```
POST /api/station-models HTTP/1.1
...

{
  "manufacturer":"Davis Instruments",
  "name":"Vantage Pro2 Plus",
  "productNumber":"6222",
  "parser":"58ac93e6e064920593e5321f"
}
```

Respuesta:

```
HTTP/1.1 201 Created
...

{
  "error": false,
  "status": "created",
  "uri": "https://ownmeteo.com/api/station-models/58da81abb9fa0f04f838dff1"
}
```

A continuación se puede apreciar el diagrama de secuencia correspondiente a esta operación:

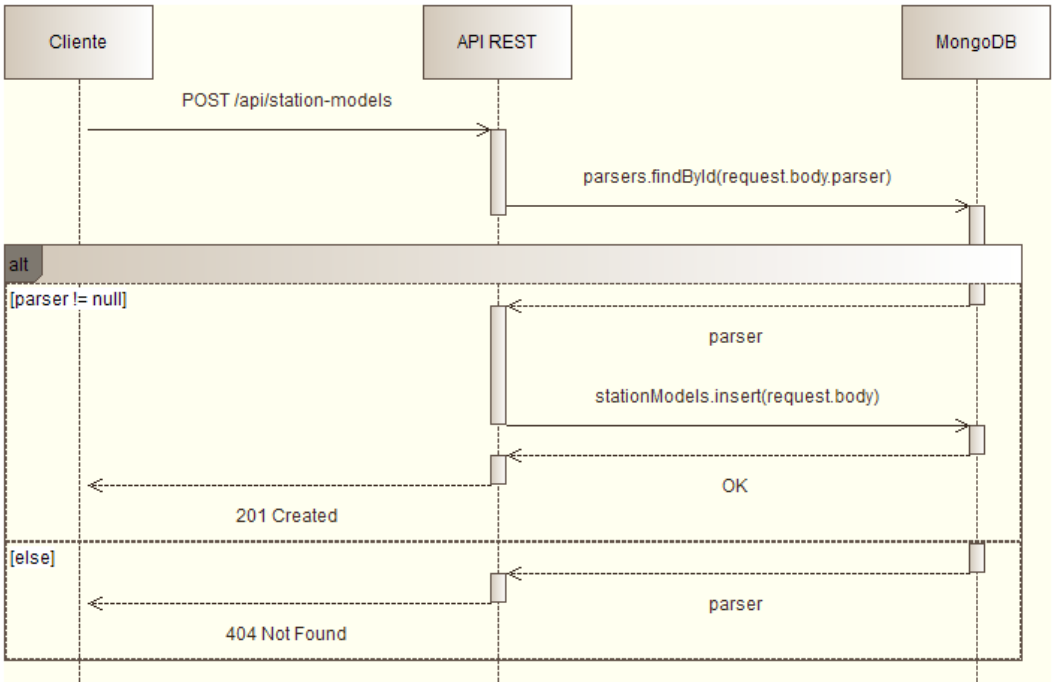

Figura 23: Diagrama de secuencia de creación de modelo de estación.

Esta operación solo puede ser invocada por clientes autenticados como usuario administrador.

GET /api/station-models/:id

Obtiene la información de la estación meteorológica cuyo ID sea el indicado en la URI de la petición. A continuación se muestra el cuerpo de una respuesta de ejemplo para esta operación:

```
{
  "_id": "58ac948ae064920593e53239",
  "manufacturer": "Davis Instrumens",
  "name": "Vantage Vue",
  "productNumber": "6250",
  "parser": "58ac93e6e064920593e5321f",
  "createdAt": "2017-02-21T19:27:06.472Z",
  "requiredData": [
    {
      "_id": "58ac9471e064920593e53229",
      "uri": "https://ownmeteo.com/api/parser-required-data/58ac9471e064920593e53229"
    }
  ],
  "uri": "https://ownmeteo.com/api/station-models"
}
```

Tal y como se puede apreciar en la traza anterior, además de la información del propio modelo de estación meteorológica, esta operación devuelve también los datos requeridos por el *parser*

asociado a este modelo. De esta manera, el cliente tendrá información acerca de los datos requeridos para crear una estación de este modelo.

Esta operación puede ser invocada por cualquier cliente autenticado en la plataforma, ya sea como usuario administrador o no.

PUT /api/station-models/:id

Modifica la información del modelo de estación cuyo ID sea el indicado en la URI de la petición. La nueva información deberá viajar en el cuerpo de la petición. A continuación se muestra un ejemplo de petición y respuesta para esta operación:

Petición:

```
PUT /api/station-models/58ac948ae064920593e53239 HTTP/1.1
...

{
  "manufacturer": "Davis Instrumens",
  "name": "Vantage Vue",
  "productNumber": "2020",
  "parser": "58ac93e6e064920593e5321f"
}
```

Respuesta:

```
HTTP/1.1 200 OK
...

{
  "error": false,
  "status": "updated",
  "uri": "https://ownmeteo.com/api/station-models/58ac948ae064920593e53239"
}
```

En la figura 24 se muestra el diagrama de secuencia correspondiente a esta operación.

Esta operación solo puede ser invocada por clientes autenticados en la plataforma como usuarios administradores.

DELETE /api/station-models/:id

Elimina el modelo de estación cuyo ID sea el indicado en la URI de la petición. Esta operación solo puede ser invocada por clientes autenticados como usuarios administradores.

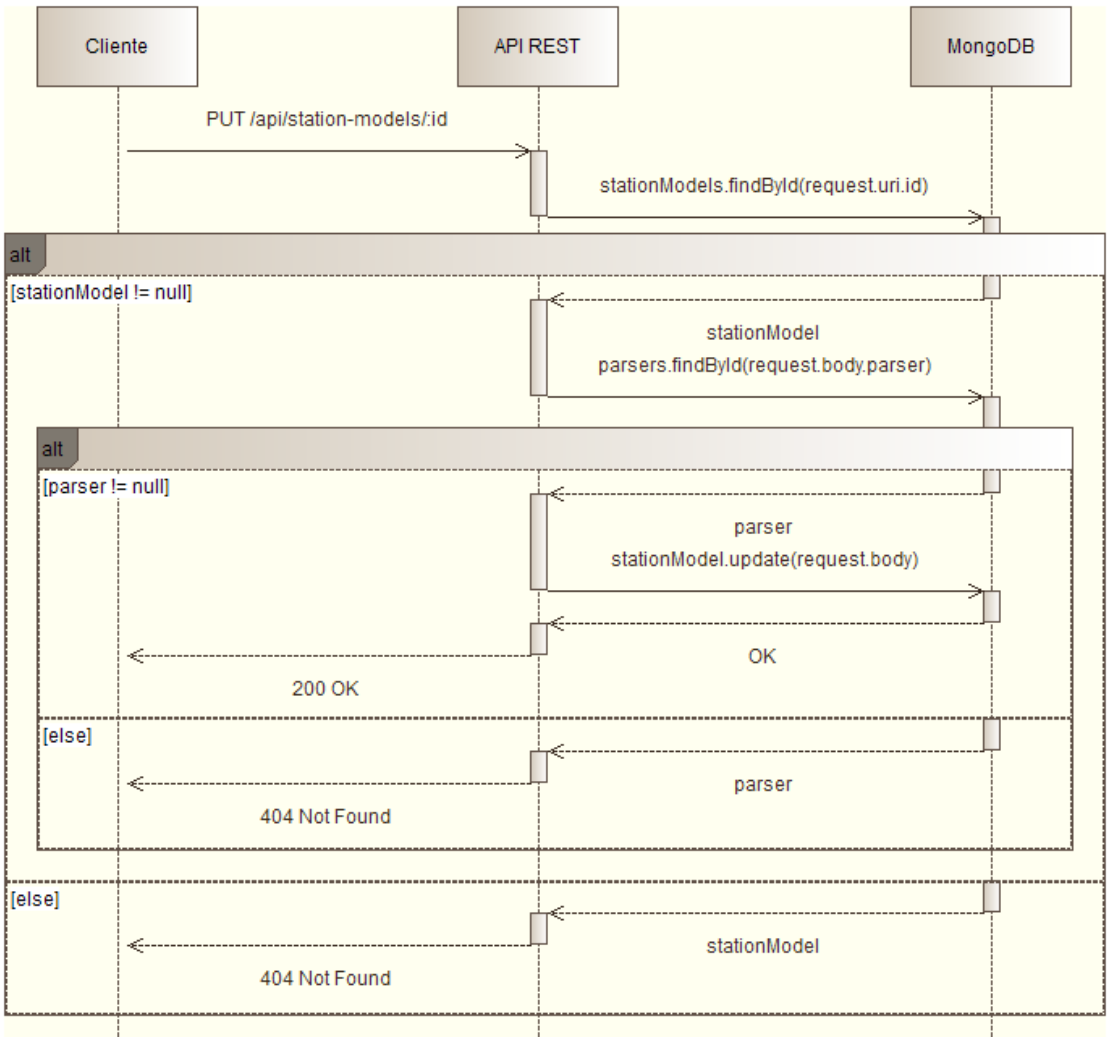

Figura 24: Diagrama de secuencia de modificación de modelo de estación.

GET /api/station-models/:id/stations

Obtiene el listado de estaciones meteorológicas registradas en el sistema cuyo modelo tiene el ID indicado en la URI de la petición. A continuación se muestra el cuerpo de una respuesta de ejemplo para esta operación:


```
[
  {
    "_id": "58ac953ce064920593e53271",
    "parserRequiredData": {
      "weatherLinkUser": "enjuanes"
    },
    "createdAt": "2017-02-21T19:30:04.284Z",
    "enabled": true,
    "currentTimezone": 3600,
    "location": "42.192117,0.342791",
    "model": "58ac948ae064920593e53239",
    "name": "Vantage Vue - Graus",
    "userOwner": "58ac94f9e064920593e53259",
    "__v": 0,
    "uri": "https://ownmeteo.com/api/stations/58ac953ce064920593e53271"
  }
]
```

Esta operación solo puede ser llevada a cabo por clientes autenticados como usuarios administradores.

A.6. Variables Meteorológicas

GET /api/meteo-vars

Obtiene el listado de variables meteorológicas, con sus respectivas unidades, reconocidas por la plataforma. A continuación se muestra el cuerpo de una respuesta de ejemplo para esta operación:


```
{
  "meteoVars": [
    {
      "_id": "58ac8f470aa073210bda2447",
      "unit": "58ac8f470aa073210bda2446",
      "name": "Concentración CO2",
      "__v": 0,
      "unitName": "ppm"
    },
    {
      "_id": "58ac8f470aa073210bda2448",
      "unit": "58ac8f470aa073210bda2445",
      "name": "Ruido Ambiental",
      "__v": 0,
      "unitName": "dB"
    },
    {
      "_id": "58ac8f470aa073210bda2449",
      "unit": "58ac8f470aa073210bda2439",
      "name": "Temperatura de Sensación",
      "__v": 0,
      "unitName": "°F"
    },
    {
      "_id": "58ac8f470aa073210bda244a",
      "unit": "58ac8f470aa073210bda2438",
      "name": "Temperatura de Sensación",
      "__v": 0,
      "unitName": "°C"
    },
    {
      "_id": "58ac8f480aa073210bda2452",
      "unit": "58ac8f470aa073210bda2442",
      "name": "Lluvia",
      "__v": 0,
      "unitName": "in"
    },
    ...
  ],
  "uri": "https://ownmeteo.com/api/meteo-vars"
}
```


Esta operación no requiere autenticación.

A.7. Estaciones Meteorológicas

GET /api/stations

Obtiene el listado de estaciones meteorológicas registradas en el sistema. A continuación se muestra el cuerpo de una respuesta de ejemplo para esta operación:


```
[
  {
    "_id": "58ac953ce064920593e53271",
    "parserRequiredData": {
      "weatherLinkUser": "enjuanes"
    },
    "createdAt": "2017-02-21T19:30:04.284Z",
    "enabled": true,
    "currentTimezone": 3600,
    "location": "42.192117,0.342791",
    "model": "58ac948ae064920593e53239",
    "name": "Vantage Vue - Graus",
    "userOwner": "58ac94f9e064920593e53259",
    "__v": 0,
    "uri": "https://ownmeteo.com/api/stations/58ac953ce064920593e53271"
  },
  {
    "_id": "58b4476c23437204bc98cb24",
    "parserRequiredData": {
      "weatherLinkUser": "ralonso"
    },
    "createdAt": "2017-02-27T15:36:12.356Z",
    "enabled": true,
    "currentTimezone": 3600,
    "location": "41.683712,-0.888141",
    "model": "58ac94c7e064920593e53246",
    "name": "Vantage Pro2 - EINA",
    "userOwner": "58ac94f9e064920593e53259",
    "__v": 0,
    "uri": "https://ownmeteo.com/api/stations/58b4476c23437204bc98cb24"
  },
  {
    "_id": "58b448d323437204bc98cb80",
    "parserRequiredData": {
      "weatherLinkUser": "mostin"
    },
    "createdAt": "2017-02-27T15:42:11.109Z",
    "enabled": true,
    "currentTimezone": 3600,
    "location": "42.587086,0.540507",
    "model": "58b448b423437204bc98cb6e",
    "name": "Vantage Pro - Cerler",
    "userOwner": "58b4483523437204bc98cb51",
    "__v": 0,
    "uri": "https://ownmeteo.com/api/stations/58b448d323437204bc98cb80"
  }
]
```


Esta operación solo puede ser invocada por clientes autenticados como usuario administrador.

POST /api/stations

Registra una nueva estación meteorológica en la plataforma. A continuación se muestra un ejemplo de petición y respuesta para esta operación:

Petición:

```
POST /api/stations HTTP/1.1
...

{
  "parserRequiredData": {
    "weatherLinkUser": "linsolescynp"
  },
  "enabled": true,
  "currentTimezone": 3600,
  "location": "42.192117,0.342791",
  "model": "58ac948ae064920593e53239",
  "name": "Vantage Vue - Benasque",
  "userOwner": "58ac94f9e064920593e53259"
}
```

Respuesta:

```
HTTP/1.1 201 Created
...

{
  "error": false,
  "status": "created",
  "uri": "https://ownmeteo.com/api/stations/58db74b3b9fa0f04f838eceb"
}
```

En la figura 25 se puede apreciar el diagrama de secuencia correspondiente a esta operación.

Para llevar a cabo esta operación, el cliente deberá estar autenticado en la plataforma. Si está autenticado como usuario administrador, se le permitirá crear una estación meteorológica asociada a cualquier usuario. Si por el contrario el cliente está autenticado como usuario no administrador, solo podrá crear estaciones meteorológicas asociadas a su propio usuario.

GET /api/stations/:id

Obtiene la información de la estación meteorológica cuyo ID sea el indicado en la URI de la petición. A continuación se muestra el cuerpo de una respuesta de ejemplo para esta operación:

```
{
  "_id": "58db74b3b9fa0f04f838eceb",
  "parserRequiredData": {
    "weatherLinkUser": "linsolescynp"
  },
  "createdAt": "2017-03-29T08:47:47.314Z",
  "enabled": true,
  "currentTimezone": 3600,
  "location": "42.192117,0.342791",
  "model": "58ac948ae064920593e53239",
  "name": "Vantage Vue - Benasque",
  "userOwner": "58ac94f9e064920593e53259",
  "__v": 0,
  "availableMeteoVars":
"https://ownmeteo.com/api/stations/58db74b3b9fa0f04f838eceb/available-meteo-vars",
  "uri": "https://ownmeteo.com/api/stations"
}
```

Para llevar a cabo esta operación, el cliente deberá estar autenticado. Si el cliente está autenticado como usuario administrador, este podrá consular la información de cualquier estación meteorológica registrada en el sistema. Si el cliente está autenticado como usuario no administrador, solo podrá consular la información de las estaciones meteorológicas asociadas a su propio usuario.

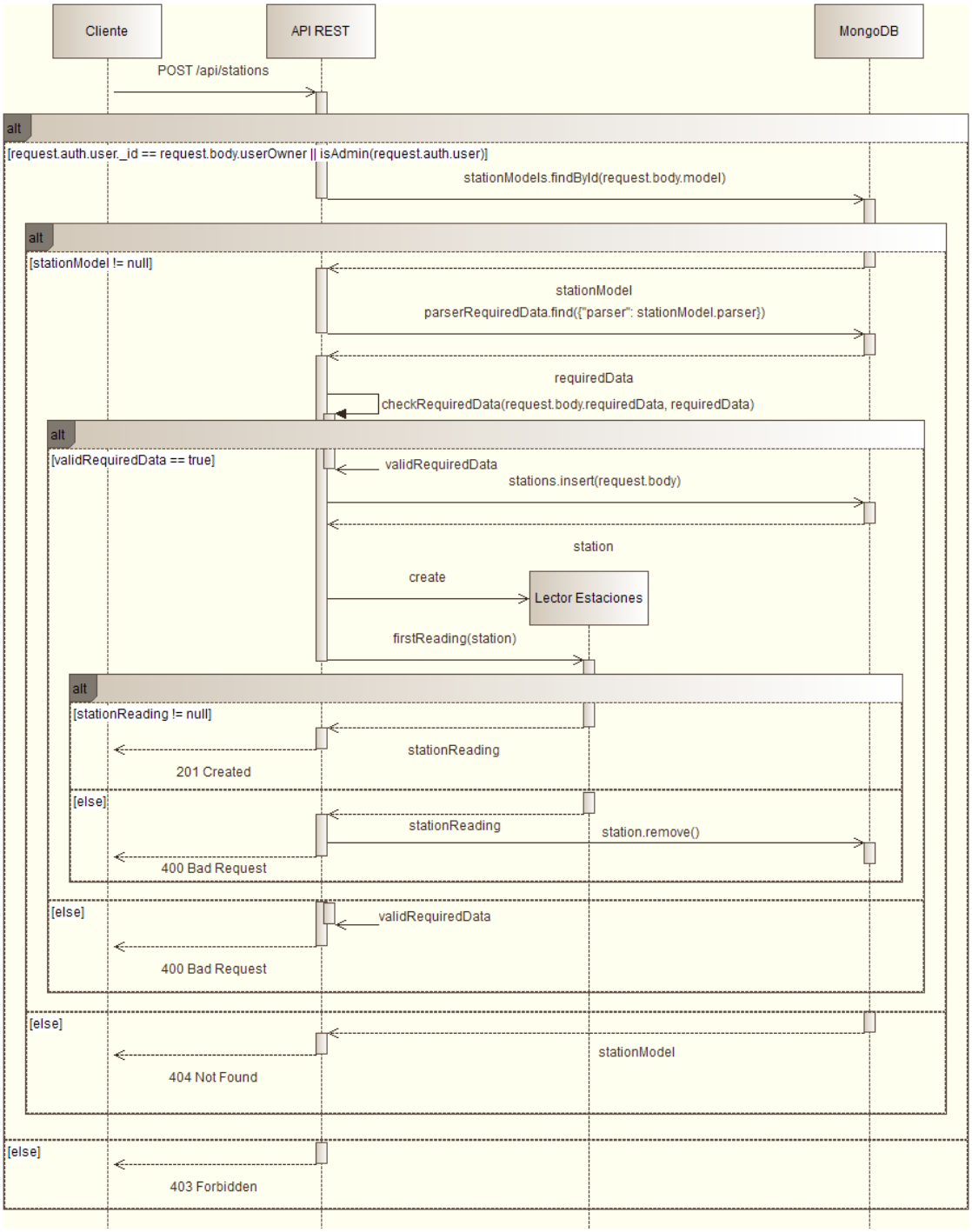

Figura 25: Diagrama de secuencia de creación de estación meteorológica.

PUT /api/stations/:id

Modifica la información de la estación meteorológica cuyo ID sea el indicado en la URI de la petición. La nueva información deberá viajar en el cuerpo de la petición. A continuación se muestra un ejemplo de petición y respuesta para esta operación:

Petición:

```
PUT /api/stations/58db74b3b9fa0f04f838eceb HTTP/1.1
...

{
  "parserRequiredData": {
    "weatherLinkUser": "linsolescynp"
  },
  "enabled": true,
  "currentTimezone": 3600,
  "location": "42.192117,0.342791",
  "model": "58ac948ae064920593e53239",
  "name": "Vantage Vue - Linsoles",
  "userOwner": "58ac94f9e064920593e53259"
}
```

Respuesta:

```
HTTP/1.1 200 OK
...

{
  "error": false,
  "status": "updated",
  "uri": "https://ownmeteo.com/api/stations/58db74b3b9fa0f04f838eceb"
}
```

En la figura 26 se muestra el diagrama de secuencia correspondiente a esta operación.

Al igual que en la operación anterior, esta operación requiere que el cliente esté autenticado en la plataforma. Si lo está como usuario administrador, podrá modificar la información de cualquier estación meteorológica, mientras que si lo está como usuario no administrador, únicamente podrá modificar estaciones de su propiedad.

DELETE /api/stations/:id

Elimina la estación meteorológica cuyo ID sea el indicado en la URI de la petición. Esta operación solo puede ser invocada por clientes autenticados como usuarios administradores.

GET /api/stations/:id/public

Obtiene un pequeño fragmento de información de la estación meteorológica cuyo ID sea el indicado en la URI de la petición. A continuación se muestra el cuerpo de una respuesta de ejemplo para esta operación:

```
{
  "_id": "58db74b3b9fa0f04f838eceb",
  "currentTimezone": 3600,
  "location": "42.192117,0.342791",
  "createdAt": "2017-03-29T08:47:47.314Z",
  "availableMeteoVars":
"https://ownmeteo.com/api/stations/58db74b3b9fa0f04f838eceb/available-meteo-vars",
  "uri": "https://ownmeteo.com/api/stations"
}
```

Esta operación puede ser llevada a cabo desde cualquier cliente, esté autenticado o no. La finalidad de la esta operación es la de dar una mínima información de la estación meteorológica a usuarios externos a la plataforma que puedan estar viendo una publicación de la propia estación meteorológica.

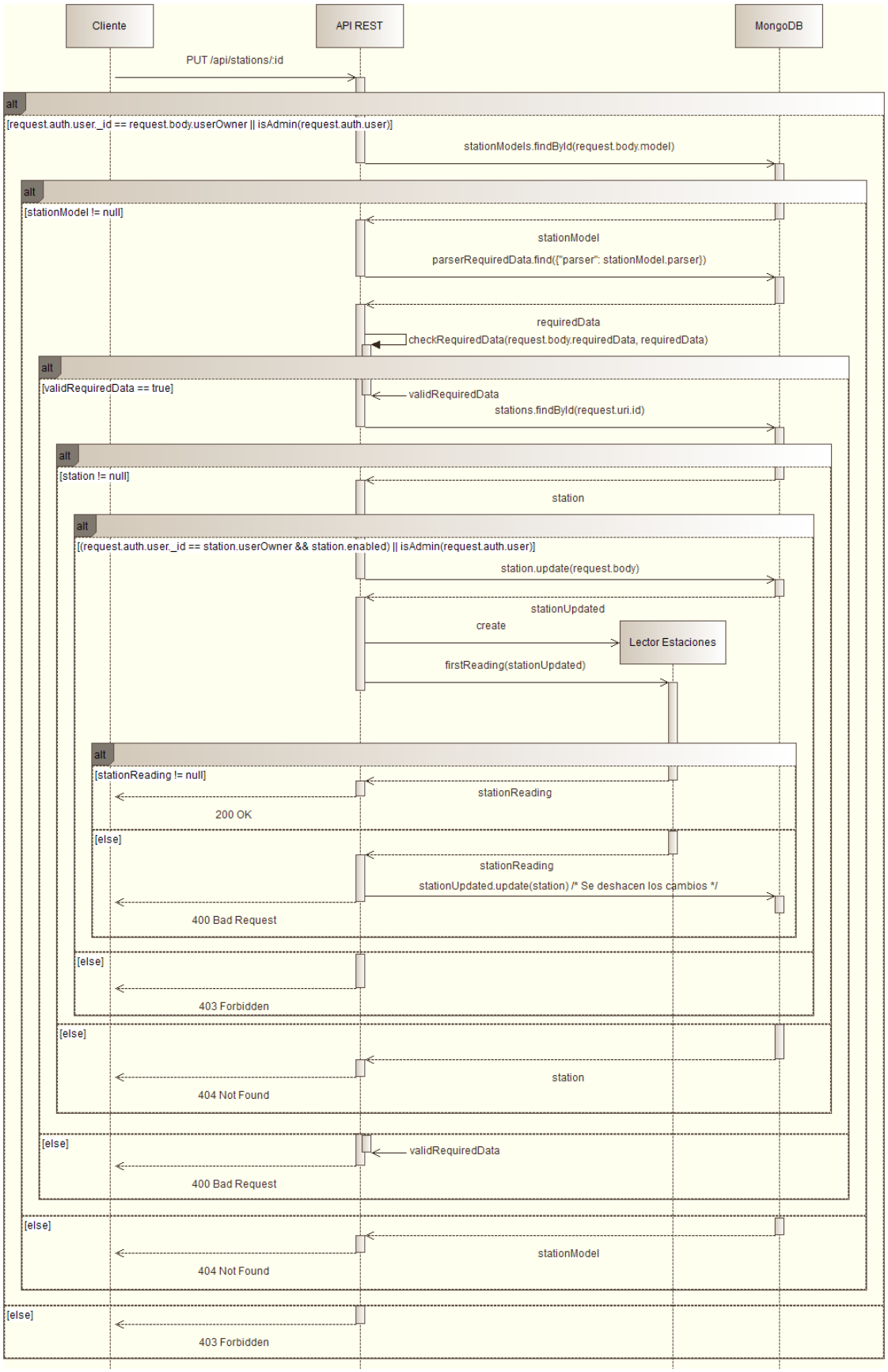

Figura 26: Diagrama de secuencia de modificación de estación meteorológica.

GET /api/stations/:id/available-meteo-vars

Obtiene el listado de variables meteorológicas leídas por la estación meteorológica cuyo ID sea el indicado en la URI de la petición. A continuación se muestra el cuerpo de una respuesta de ejemplo para esta operación:


```
{
  "station": "58db74b3b9fa0f04f838eceb",
  "uri": "https://ownmeteo.com/api/stations/58db74b3b9fa0f04f838eceb",
  "availableMeteoVars": [
    "outsideTemp",
    "outsideHum",
    "insideTemp",
    "insideHum",
    "heatIndex",
    "windChill",
    "dewPoint",
    "atmPressure",
    "windSpeed",
    "windDirection",
    "rain"
  ]
}
```


Esta operación puede ser ejecutada por cualquier cliente, esté o no autenticado en la plataforma.

GET /api/stations/:id/weather

Obtiene la última lectura meteorológica de la estación cuyo ID sea el indicado en la URI de la petición. A continuación se muestra el cuerpo de una respuesta de ejemplo para esta operación:


```
{
  "_id": "58dbdb34b9fa0f04f838f372",
  "readTime": "2017-03-29T17:04:00.000Z",
  "station": "58ac953ce064920593e53271",
  "__v": 0,
  "data": [
    {
      "outsideTemp": 20.4
    },
    {
      "outsideHum": 45
    },
    {
      "insideTemp": 19.1
    },
    {
      "insideHum": 44
    },
    {
      "heatIndex": 19.4
    },
    {
      "atmPressure": 1004.9
    },
    {
      "rain": 232.8
    }
  ],
  "uri": "https://ownmeteo.com/api/stations/58ac953ce064920593e53271/weather"
}
```


Esta operación puede ser ejecutada por cualquier cliente, esté o no autenticado en la plataforma.

POST /api/stations/:id/weather

Obtiene estadísticas de datos meteorológicos históricos de la estación cuyo ID sea el indicado en la URI de la petición. El cuerpo de la petición deberá tener el periodo del cual se desee obtener la información meteorológica.

A continuación se muestran algunos cuerpos de peticiones de ejemplo para esta operación:

```
{
  "period": {}
}
```

En el ejemplo anterior, no se especifica periodo alguno, por lo que la operación devolverá la información meteorológica tomada por la estación desde que fue registrada en la plataforma hasta la actualidad, año a año (es decir, se devolverá la información meteorológica agrupada por años).

```
{
  "period": { "year": 2017 }
}
```

En el ejemplo anterior se ha especificado como periodo el año 2017, por lo que la operación devolverá la información meteorológica tomada por la estación en dicho año, mes a mes (es decir, se devolverá la información meteorológica agrupada por meses).

```
{
  "period": { "year": 2017, "month": 1 }
}
```

En el ejemplo anterior se ha especificado como periodo el mes de febrero (enero es el mes 0) de 2017, por lo que la operación devolverá la información meteorológica tomada por la estación en dicho mes, día a día (es decir, se devolverá la información meteorológica agrupada por días).

```
{
  "period": { "year": 2017, "month": 1, "day": 28 }
}
```

En el ejemplo anterior se especifica como periodo el día 28 de febrero de 2017, por lo que la operación devolverá la información meteorológica tomada por la estación en dicho día, hora a hora (es decir, se devolverá la información meteorológica agrupada por horas).

Finalmente se muestra a modo de ejemplo la respuesta de esta operación a una petición donde no se especifica periodo alguno (es decir, como el primer ejemplo):

```
[
  {
    "period":{
      "year":2017
    },
    "start":"2016-12-31T22:00:00.000Z",
    "end":"2017-12-31T21:59:59.999Z",
    "data":[
      {
        "outsideTemp":{
          "max":25.8,
          "min":-2.8,
          "avg":9.83
        }
      },
      {
        "outsideHum":{
          "max":97,
```

```
          "min":22,
          "avg":72.13
        }
      },
      {
        "insideTemp":{
          "max":22.1,
          "min":18.2,
          "avg":19.73
        }
      },
      {
        "insideHum":{
          "max":50,
          "min":39,
          "avg":43.85
        }
      },
      {
        "heatIndex":{
          "max":25,
          "min":-2.8,
          "avg":9.42
        }
      },
      {
        "atmPressure":{
          "max":1012.4,
          "min":971.5,
          "avg":999.01
        }
      },
      {
        "rain":{
          "fallen":235.8
        }
      }
    ]
  }
]
```


Tal y como se puede apreciar, se ha devuelto información meteorológica tomada por la estación desde que fue registrada en la plataforma (febrero de 2017) hasta la actualidad, agrupada en años (en este caso solo uno).

Esta operación puede ser ejecutada por cualquier cliente, esté o no autenticado en la plataforma.

PUT /api/stations/:id/weather

Registra una nueva lectura meteorológica para la estación cuyo ID sea el indicado en la URI de la petición a partir de los datos contenidos en el cuerpo de la petición. A continuación se muestra una petición y su respuesta a modo de ejemplo:

Petición:


```
PUT /api/stations/58b448d323437204bc98cb80/weather HTTP/1.1
...

{
  "readTime":"2017-04-01T17:27:00.000Z",
  "data":[
    {
      "outsideTemp":8.3
    },
    {
      "outsideHum":47
    },
```

```
    {
      "insideTemp":20.8
    },
    {
      "insideHum":30
    },
    {
      "heatIndex":7.8
    },
    {
      "windChill":7.8
    },
    {
      "dewPoint":-2.8
    },
    {
      "atmPressure":1010.9
    },
    {
      "windSpeed":3
    },
    {
      "windDirection":319
    },
    {
      "rain":300.6
    }
  ],
  "station":"58b448d323437204bc98cb80"
}
```

Respuesta:

```
HTTP/1.1 201 Created
...

{
  "error":false,
  "status":"created",
  "uri":"https://ownmeteo.com/api/stations/58b448d323437204bc98cb80/weather"
}
```

Esta operación solo se puede invocar por clientes autenticados en la plataforma como usuarios administradores. Normalmente, esta operación es utilizada principalmente por los Lectores de Estaciones Meteorológicas (nodos encargados de leer la información meteorológica de las estaciones y registrarla periódicamente a través de esta operación).

GET /api/stations/:id/posts

Obtiene las publicaciones de información meteorológica asociadas a la estación cuyo ID sea el especificado en la URI de la petición. A continuación se muestra el cuerpo de una respuesta de ejemplo para esta operación:

```
[
  {
    "_id":"58ac9553e064920593e53283",
    "enabled":true,
    "name":"eltiempoengraus.com",
    "station":"58ac953ce064920593e53271",
    "__v":1,
    "shownData":[
      "58ac8f480aa073210bda245e",
      "58ac8f480aa073210bda245a",
      "58ac8f480aa073210bda2458",
      "58ac8f480aa073210bda2455",
      "58ac8f480aa073210bda2453",
      "58ac8f480aa073210bda2451"
```

```
    ],
    "uri":"https://ownmeteo.com/api/posts/58ac9553e064920593e53283"
  }
]
```

Solo los clientes autenticados en la plataforma pueden hacer uso de esta operación. Si los clientes están autenticados como usuarios no administradores únicamente podrán consultar las publicaciones asociadas a estaciones meteorológicas de su propiedad. Por el contrario, los usuarios administradores podrán consultar las publicaciones asociadas a cualquier estación meteorológica.

A.8. Publicaciones

GET /api/posts

Obtiene el listado completo de publicaciones de información meteorológica registradas en el sistema. A continuación se muestra el cuerpo de una respuesta de ejemplo para esta operación:

```
[
  {
    "_id": "58ac9553e064920593e53283",
    "enabled": true,
    "name": "eltiempoengraus.com",
    "station": "58ac953ce064920593e53271",
    "__v": 1,
    "shownData": [
      "58ac8f480aa073210bda245e",
      "58ac8f480aa073210bda245a",
      "58ac8f480aa073210bda2458",
      "58ac8f480aa073210bda2455",
      "58ac8f480aa073210bda2453",
      "58ac8f480aa073210bda2451"
    ],
    "uri": "https://ownmeteo.com/api/posts/58ac9553e064920593e53283"
  },
  {
    "_id": "58bfde04029fd704ee807ddf",
    "enabled": true,
    "name": "El Tiempo en Wellington",
    "station": "58bfddce029fd704ee807dcd",
    "__v": 0,
    "shownData": [
      "58ac8f480aa073210bda245d",
      "58ac8f480aa073210bda245a",
      "58ac8f480aa073210bda2458",
      "58ac8f480aa073210bda2455",
      "58ac8f480aa073210bda2453",
      "58ac8f470aa073210bda2450",
      "58ac8f470aa073210bda244f"
    ],
    "uri": "https://ownmeteo.com/api/posts/58bfde04029fd704ee807ddf"
  }
]
```

Esta operación solo puede ser llevada a cabo por clientes autenticados como usuarios administradores.

POST /api/posts

Crea una nueva publicación de información meteorológica utilizando los datos contenidos en el cuerpo de la petición. A continuación se muestra una petición y su respuesta a modo de ejemplo:

Petición:

```
POST /api/posts HTTP/1.1
...

{
  "name":"El Tiempo en la EINA",
  "shownData":[
    "58ac8f470aa073210bda244f",
    "58ac8f470aa073210bda2450",
    "58ac8f480aa073210bda2453",
    "58ac8f480aa073210bda2455",
    "58ac8f480aa073210bda2458",
    "58ac8f480aa073210bda245a",
    "58ac8f480aa073210bda245e"
  ],
  "enabled":true,
  "station":"58b4476c23437204bc98cb24"
}
```

Respuesta:

```
HTTP/1.1 201 Created
...

{
  "error":false,
  "status":"created",
  "uri":"https://ownmeteo.com/api/posts/58dfe651c9b64e04fc5657c9"
}
```

En la figura 27 se muestra el diagrama de secuencia correspondiente a esta operación.

Esta operación requiere que el cliente esté autenticado en la plataforma. Si el cliente está autenticado como usuario no administrador únicamente podrá crear publicaciones vinculadas a estaciones meteorológicas de su propiedad. En caso contrario, si está autenticado como usuario administrador, podrá crear publicaciones asociadas a cualquier estación meteorológica.

GET /api/posts/:id

Obtiene la información de la publicación de información meteorológica cuyo ID sea el indicado en la URI de la petición. A continuación se muestra el cuerpo de una respuesta de ejemplo para esta operación:

```
{
  "_id":"58ac9553e064920593e53283",
  "enabled":true,
  "name":"eltiempoengraus.com",
  "station":"58ac953ce064920593e53271",
  "__v":1,
  "shownData":[
    "58ac8f480aa073210bda245e",
    "58ac8f480aa073210bda245a",
    "58ac8f480aa073210bda2458",
    "58ac8f480aa073210bda2455",
    "58ac8f480aa073210bda2453",
    "58ac8f480aa073210bda2451"
  ],
  "uri":"https://ownmeteo.com/api/posts"
}
```

Esta operación no requiere que el cliente esté autenticado en la plataforma.

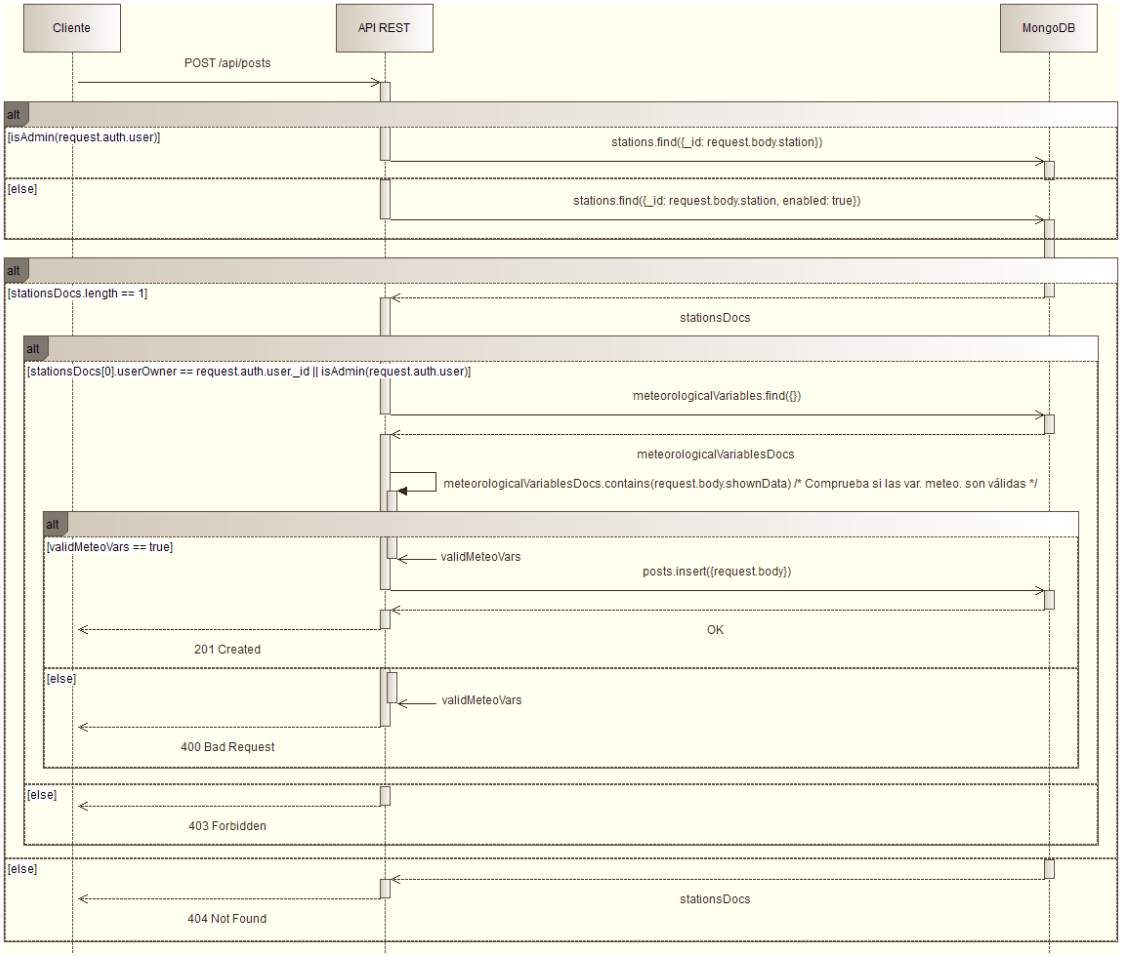

Figura 27: Diagrama de secuencia de creación de publicación de información meteorológica.

PUT /api/posts/:id

Modifica la información de la publicación de información meteorológica cuyo ID sea el indicado en la URI de la petición. La nueva información deberá viajar en el cuerpo de la petición. A continuación se muestra un ejemplo de petición y respuesta para esta operación:

Petición:

```
PUT /api/posts/58dfe651c9b64e04fc5657c9 HTTP/1.1
...

{
  "enabled":true,
  "name":"El Tiempo en la EINA",
  "station":"58b4476c23437204bc98cb24",
  "shownData":[
    "58ac8f480aa073210bda2451", "58ac8f480aa073210bda2453", "58ac8f480aa073210bda2455",
    "58ac8f480aa073210bda245a", "58ac8f480aa073210bda245e"
  ]
}
```

Respuesta:

```
HTTP/1.1 200 OK
...

{
  "error":false,
```

```
  "status":"updated",
  "uri":"https://ownmeteo.com/api/posts/58dfe651c9b64e04fc5657c9"
}
```

Esta operación tiene gran complejidad, por lo que no se ha representado en un diagrama de secuencia (debido a que sería confuso y no sería visible en este documento). Esta operación es bastante similar a la de creación de publicaciones, sin embargo, requiere de la ejecución de más comprobaciones para poder modificar una publicación satisfactoriamente. A continuación se enumeran estas comprobaciones adicionales:

- Comprobar que la publicación que se desea modificar existe. Además, si el usuario que lleva a cabo la operación no es administrador, se debe verificar que la publicación no esté desactivada (ya que si lo está, solo los usuarios administradores podrán modificarla).
- Si el usuario que lleva a cabo la operación es no administrador, se debe comprobar que la estación meteorológica vinculada a la publicación es de su propiedad (ya que si no podría modificar publicaciones ajenas a su usuario) y que dicha estación no está desactivada.
- Además, si el usuario es no administrador, se debe comprobar adicionalmente que la nueva estación de la publicación es también de su propiedad y que dicha estación no está desactivada.

Esta operación requiere que los clientes estén autenticados en la plataforma. Además los clientes autenticados como usuarios no administradores tendrán las restricciones mencionadas en la enumeración anterior.

DELETE /api/posts/:id

Elimina la publicación de información meteorológica cuyo ID sea el indicado en la URI de la petición. Esta operación solo puede ser invocada por clientes autenticados como usuarios administradores.

A.9. Accesos al API

GET /api/api-accesses

Obtiene el listado completo de accesos al API registrados en el sistema. A continuación se muestra el cuerpo de una respuesta de ejemplo para esta operación:

```
[
  {
    "_id":"58e0d390c9b64e04fc566704",
    "userAgent":"Mozilla/5.0 ... Safari/537.36",
    "ip":"212.97.169.61",
    "user":"58ac8f470aa073210bda2437",
    "responseStatus":304,
    "accessDate":"2017-04-02T10:33:52.537Z",
    "url":"GET /api/sessions/current",
    "__v":0,
    "uri":"https://ownmeteo.com/api/api-accesses/58e0d390c9b64e04fc566704"
  },
  {
    "_id":"58e0d38ec9b64e04fc566703",
    "userAgent":"Mozilla/5.0 ... Safari/537.36",
    "ip":"212.97.169.61",
    "user":"58ac8f470aa073210bda2437",
    "responseStatus":200,
    "accessDate":"2017-04-02T10:33:50.528Z",
```

```
    "url":"GET /api/users/58ac8f470aa073210bda2437",
    "__v":0,
    "uri":"https://ownmeteo.com/api/api-accesses/58e0d38ec9b64e04fc566703"
  }
]
```

Esta operación solo puede ser llevada a cabo por clientes autenticados como usuarios administradores.

Es importante destacar que no es muy recomendable utilizar esta operación frecuentemente, debido principalmente al gran número de accesos al API que pudiera haber registrados en el sistema. Es más recomendable utilizar la siguiente operación:

POST /api/api-accesses

Obtiene un listado de accesos al API en función de los parámetros enviados en el cuerpo de la petición. Así, el cuerpo de la petición deberá especificar:

- Límite de accesos al API devueltos por la operación (campo obligatorio).
- Criterio por el cual se ordenarán los accesos al API devueltos (campo obligatorio).
- Fecha mínima de los accesos al API devueltos (campo opcional).
- Fecha máxima de los accesos al API devueltos (campo opcional).

A continuación se muestra una petición de ejemplo y su respuesta para esta operación:

Petición:

```
POST /api/api-accesses HTTP/1.1
...

{
  "limit":2,
  "fromDate":"2017-01-31T23:00:00.000Z",
  "untilDate":"2017-02-28T23:00:00.000Z",
  "sort":{
    "accessDate":-1
  }
}
```

Respuesta:

```
HTTP/1.1 200 OK
...

[
  {
    "_id":"58b5ffcd029fd704ee801007",
    "ip":"::ffff:127.0.0.1",
    "user":"58ac8f470aa073210bda2437",
    "responseStatus":201,
    "accessDate":"2017-02-28T22:55:09.599Z",
    "url":"PUT /api/stations/58b4476c23437204bc98cb24/weather",
    "__v":0,
    "uri":"https://ownmeteo.com/api/api-accesses/58b5ffcd029fd704ee801007"
  },
  {
    "_id":"58b5ffcd029fd704ee801005",
    "ip":"::ffff:127.0.0.1",
    "user":"58ac8f470aa073210bda2437",
    "responseStatus":201,
    "accessDate":"2017-02-28T22:55:09.438Z",
    "url":"PUT /api/stations/58b448d323437204bc98cb80/weather",
    "__v":0,
    "uri":"https://ownmeteo.com/api/api-accesses/58b5ffcd029fd704ee801005"
```

```
  }
]
```

Esta operación requiere que el cliente esté autenticado como usuario administrador.

GET /api/api-accesses/:id

Obtiene la información del acceso al API cuyo ID sea el indicado en la URI de la petición. A continuación se muestra el cuerpo de una respuesta de ejemplo para esta operación:

```
{
  "_id":"58b5ffcd029fd704ee801007",
  "ip":"::ffff:127.0.0.1",
  "user":"58ac8f470aa073210bda2437",
  "responseStatus":201,
  "accessDate":"2017-02-28T22:55:09.599Z",
  "url":"PUT /api/stations/58b4476c23437204bc98cb24/weather",
  "__v":0,
  "uri":"https://ownmeteo.com/api/api-accesses"
}
```

Esta operación requiere que el cliente esté autenticado como usuario administrador.

A.10. Incidencias

GET /api/incidences

Obtiene el listado completo de incidencias registradas en el sistema. A continuación se muestra el cuerpo de una respuesta de ejemplo para esta operación:

```
[
  {
    "_id":"58dfd4d2c9b64e04fc565634",
    "incidenceDate":"2017-04-01T16:26:58.912Z",
    "description":"Error ... reading data. STATION ID: 58bfddce029fd704ee807dcd",
    "level":"error",
    "code":"davPar01_2",
    "__v":0,
    "uri":"https://ownmeteo.com/api/incidences/58dfd4d2c9b64e04fc565634"
  },
  {
    "_id":"58dfd4d2c9b64e04fc565630",
    "incidenceDate":"2017-04-01T16:26:58.898Z",
    "description":"Error ... reading data. STATION ID: 58db74b3b9fa0f04f838eceb",
    "level":"error",
    "code":"davPar01_2",
    "__v":0,
    "uri":"https://ownmeteo.com/api/incidences/58dfd4d2c9b64e04fc565630"
  }
]
```

Esta operación solo puede ser llevada a cabo por clientes autenticados como usuarios administradores.

Es importante destacar que no es muy recomendable utilizar esta operación frecuentemente, debido principalmente al gran número de incidencias que pudiera haber registrados en el sistema. Es más recomendable utilizar la siguiente operación:

POST /api/incidences

Obtiene un listado de incidencias en función de los parámetros enviados en el cuerpo de la petición. Así, el cuerpo de la petición deberá especificar:

- Límite de incidencias devueltas por la operación (campo obligatorio).
- Criterio por el cual se ordenarán las incidencias devueltas (campo obligatorio).
- Fecha mínima de las incidencias devueltas (campo opcional).
- Fecha máxima de las incidencias devueltas (campo opcional).

A continuación se muestra una petición de ejemplo y su respuesta para esta operación:

Petición:

```
POST /api/incidences HTTP/1.1
...

{
  "limit":2,
  "fromDate":"2017-01-31T23:00:00.000Z",
  "untilDate":"2017-02-28T23:00:00.000Z",
  "sort":{
    "incidenceDate":-1
  }
}
```

Respuesta:

```
[
  {
    "_id":"58dd76d6b9fa0f04f8390e49",
    "incidenceDate":"2017-03-30T21:21:26.440Z",
    "description":"Error ... station data. STATION ID: 58db74b3b9fa0f04f838eceb",
    "level":"error",
    "code":"davPar01_1",
    "__v":0,
    "uri":"https://ownmeteo.com/api/incidences/58dd76d6b9fa0f04f8390e49"
  },
  {
    "_id":"58dd76d6b9fa0f04f8390e47",
    "incidenceDate":"2017-03-30T21:21:26.433Z",
    "description":"Error ... station data. STATION ID: 58bfddce029fd704ee807dcd",
    "level":"error",
    "code":"davPar01_1",
    "__v":0,
    "uri":"https://ownmeteo.com/api/incidences/58dd76d6b9fa0f04f8390e47"
  }
]
```

Esta operación requiere que el cliente esté autenticado como usuario administrador.

PUT /api/incidences

Registra una nueva incidencia tomando los datos contenidos en el cuerpo de la petición. A continuación se muestra una petición de ejemplo y su respuesta para esta operación:

Petición:

```
PUT /api/incidences HTTP/1.1
...

{
  "code": "example_1",
  "description": "Incidence Example",
```

```
  "level": "warning"
}
```

Respuesta:

```
HTTP/1.1 201 Created
...

{
  "error": false,
  "status": "created",
  "uri": "https://ownmeteo.com/api/incidences/58e1147b0e576a42a0bcbdee"
}
```

Esta operación requiere que el cliente esté autenticado como usuario administrador.

GET /api/incidences/:id

Obtiene la información de la incidencia cuyo ID sea el indicado en la URI de la petición. A continuación se muestra el cuerpo de una respuesta de ejemplo para esta operación:

```
{
  "_id": "58e1147b0e576a42a0bcbdee",
  "incidenceDate": "2017-04-02T15:10:51.917Z",
  "description": "Incidence Example",
  "level": "warning",
  "code": "example_1",
  "__v": 0,
  "uri": "https://ownmeteo.com/api/incidences"
}
```

Esta operación requiere que el cliente esté autenticado como usuario administrador.

Anexo B - Mapas de Navegación

En este Anexo se presentan dos mapas de navegación; el mapa de navegación de la aplicación web si el usuario está autenticado como usuario no administrador y el mapa de navegación para usuarios administradores.

En el Anexo C se pueden encontrar descripciones detalladas de las vistas presentadas en los mapas de navegación de este Anexo.

B.1. Usuarios Autenticados como Usuarios no Administradores

En la figura 28 se puede apreciar el mapa de navegación para usuarios no administradores:

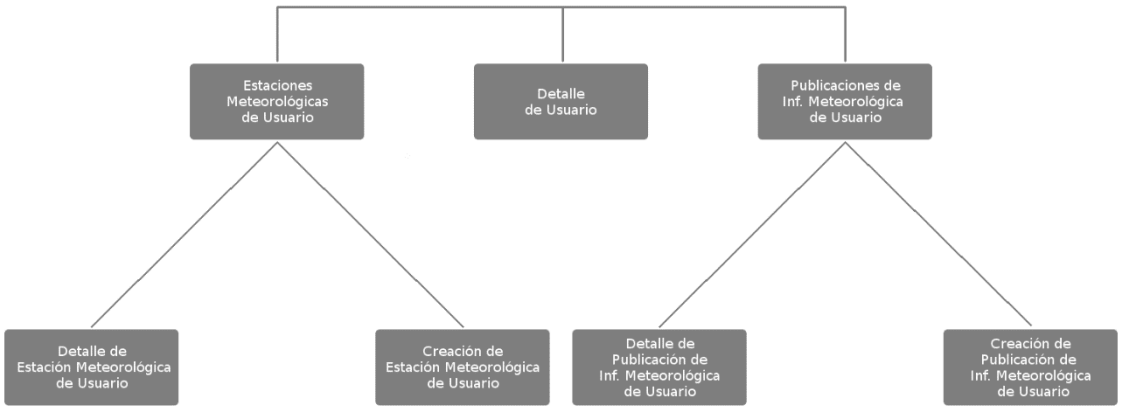

Figura 28: Mapa de navegación de usuarios autenticados como usuarios no administradores.

Así, una vez el usuario no administrador se autentique en la plataforma podrá navegar por las 7 vistas del mapa de navegación anterior. Es importante destacar que la cabecera de la aplicación web es común a todas las vistas, y que en dicha cabecera existen hipervínculos a las siguientes vistas: Estaciones Meteorológicas de Usuario, Detalle de Usuario y Publicaciones de Información Meteorológica de Usuario. Por lo tanto, estas 3 vistas son accesibles desde el resto de vistas, sin embargo, esto no se ha reflejado en el mapa de navegación para facilitar su comprensión.

B.2. Usuarios Autenticados como Usuarios Administradores

En la figura 29 se puede apreciar el mapa de navegación para usuarios administradores.

Una vez el administrador se autentique en la plataforma tendrá acceso a las 19 vistas presentadas en el mapa de navegación. Como en el caso anterior, la cabecera de la aplicación web es común a todas las vistas y en esta cabecera hay hipervínculos a las siguientes vistas: Panel de Administración, Usuarios, *Parsers*, Modelos de Estación Meteorológica, Estaciones Meteorológicas de Administrador, Publicaciones de Información Meteorológica de Administrador, Accesos al API e Incidencias. Así pues, estas vistas son accesibles desde el resto de vistas, sin embargo, como en el caso anterior, esto no se ha reflejado en el mapa de navegación con el objetivo de facilitar la comprensión del mismo.

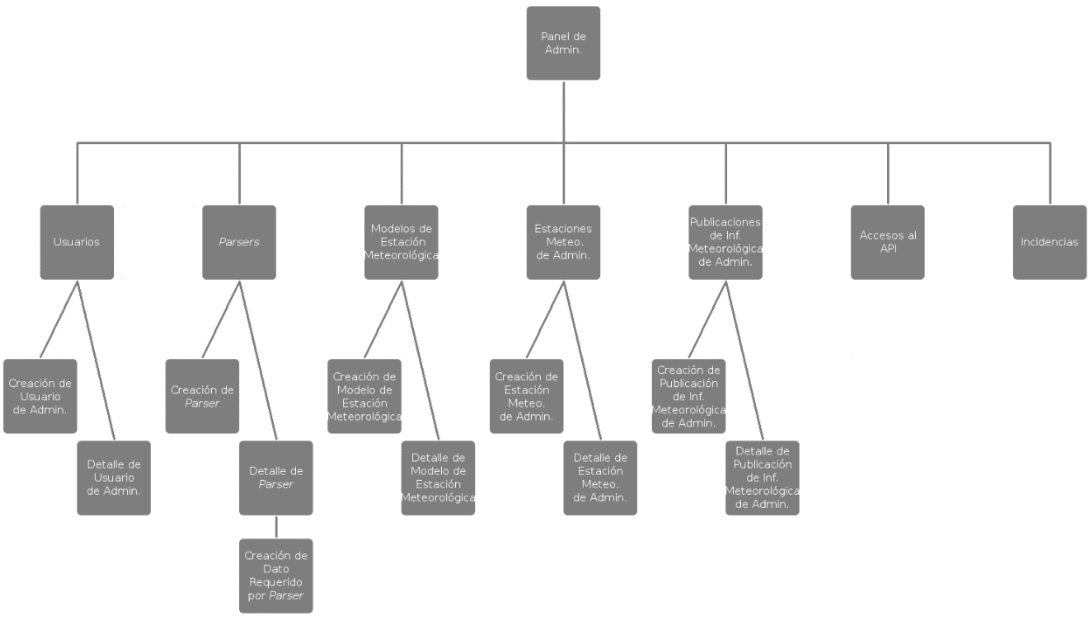

Figura 29: Mapa de navegación de usuarios autenticados como usuarios administradores.

Anexo C - Vistas y Controladores en el *front-end*

En este Anexo se describen las diferentes vistas desarrolladas para la aplicación web y los controladores AngularJS implementados para el correcto funcionamiento de las mismas.

A continuación se describen pues las vistas (y sus respectivos controladores), clasificadas según el nivel de autenticación del usuario que las vaya a utilizar: usuario no autenticado, usuario autenticado como no administrador y usuario autenticado como administrador.

C.1. Usuarios no Autenticados

En este apartado se describen las vistas que no requieren autenticación alguna, de forma que cualquier usuario con acceso a Internet podrá utilizar.

C.1.1. Portada del Sitio

Esta vista contiene información acerca de la plataforma *ownmeteo.com*: su objetivo, funcionalidades, etc. A continuación se muestra una captura de esta vista:

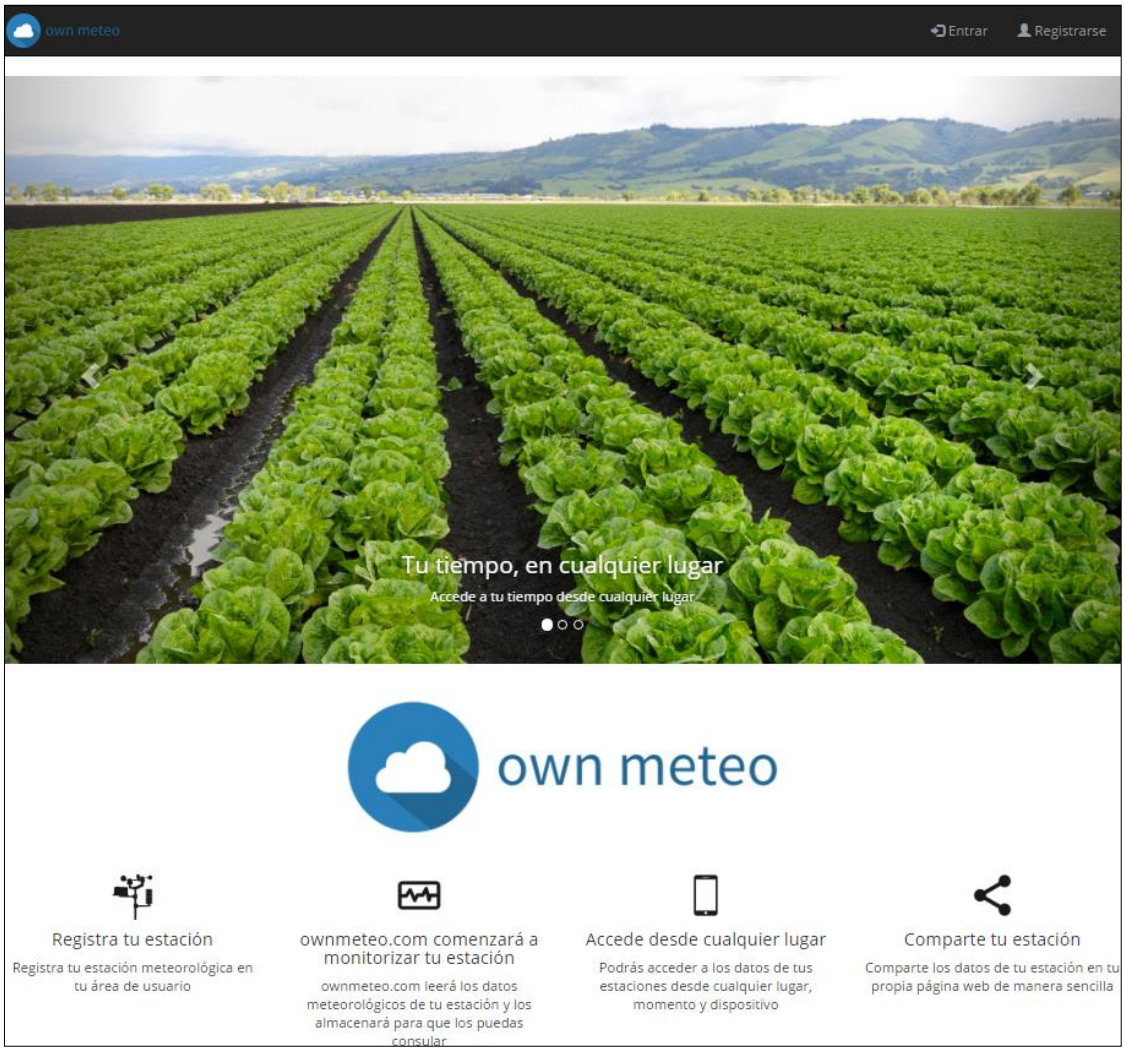

Figura 30: Captura de la portada del sitio.

Esta vista no hace uso de ningún controlador AngularJS (exceptuando el de la propia barra de navegación, común a todas las vistas del sitio web) debido a que no requiere ninguna funcionalidad concreta.

C.1.2. Vista de Registro

Esta vista contiene un formulario de registro donde los nuevos usuarios pueden registrarse en la plataforma. Si el usuario rellena el formulario correctamente y pulsa el botón *Registrarse* se creará un nuevo usuario con los datos introducidos. A continuación se muestra una captura de esta vista:

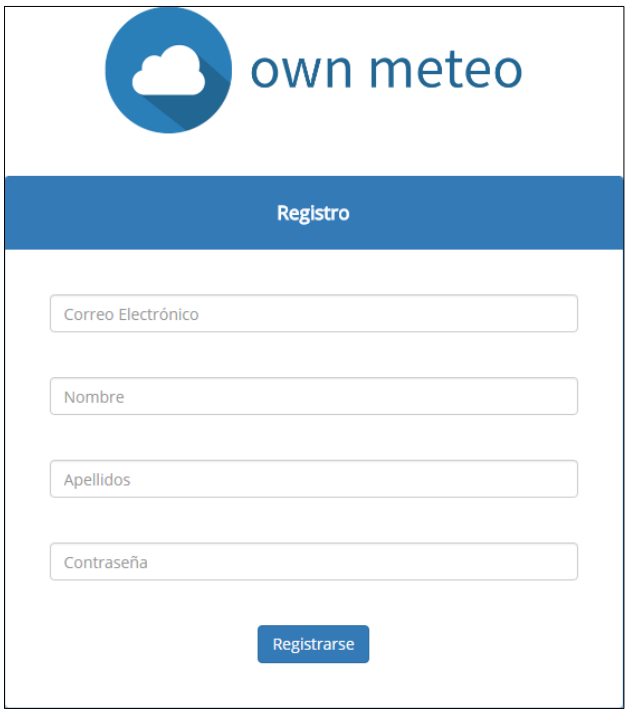

Figura 31: Captura de la vista de registro.

La vista de registro hace uso del controlador `register`, el cual se encarga de tomar los datos introducidos en el formulario y enviar la petición de creación de usuario al API cuando el usuario pulsa el botón *Registrarse*. Si el registro concluye satisfactoriamente, el controlador crea una nueva sesión para el usuario recién creado y redirige al usuario a la vista de estaciones meteorológicas de usuario, explicada más adelante en este Anexo.

Además de ello, si el controlador detecta que el usuario ya está autenticado en la plataforma, lo redirige a la portada del sitio (de forma que no vuelva a registrarse por error, por ejemplo).

C.1.3. Vista de Entrar/*Log In*

Esta vista contiene un formulario cuya finalidad es permitir al usuario introducir sus credenciales y autenticarse contra la plataforma. En la figura 32 se puede apreciar la captura de pantalla relativa a esta vista.

La vista hace uso del controlador `login`, el cual se encarga de tomar los credenciales del usuario y enviar la petición de creación de sesión al API cuando el usuario pulsa el botón *Entrar*. Si la sesión se crea correctamente, el controlador redirigirá al usuario la vista de estaciones meteorológicas de usuario.

Además de ello, se encarga de redirigir al usuario a la portada del sitio si detecta que ya está autenticado, de forma que no vuelva a autenticarse.

Figura 32: Captura de la vista de entrar.

C.1.4. Vista de Compartición de Información Meteorológica en Sitios Web de Terceros

Como ya se ha comentado anteriormente, se pueden crear publicaciones de información meteorológica y compartirlas en sitios web terceros, incrustando en ellos un elemento HTML proporcionado por la propia aplicación web de *ownmeteo.com*. En este apartado se describe la propia vista que será incrustada en los sitios web terceros.

Básicamente, esta vista contiene los datos meteorológicos actuales tomados por la estación meteorológica vinculada a la publicación de información meteorológica. A continuación se muestra una captura de pantalla donde se aprecia esta vista incrustada en un sitio web tercero:

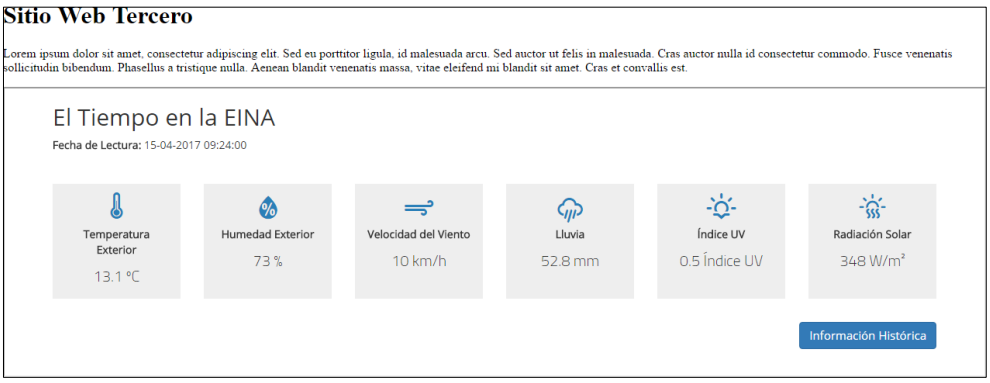

Figura 33: Captura de la vista de compartición de información meteorológica en sitios web de terceros.

Esta vista hace uso del controlador `embed`, el cual se encarga principalmente de:

- Obtener los datos de la publicación meteorológica (a través del API REST): nombre, datos meteorológicos mostrados…
- Obtener la lectura de información meteorológica más reciente para la estación vinculada a la publicación (haciendo uso del API REST).

C.1.5. Vista de Información Meteorológica Histórica

Si el usuario pulsa el botón *Información Histórica* en la vista explicada en el apartado anterior será redirigido a esta vista, la cual muestra los datos meteorológicos históricos recogidos por la estación meteorológica desde que fuera registrada en la plataforma. Así, esta vista muestra tantas gráficas de información meteorológica histórica como variables meteorológicas a mostrar tenga la publicación de información meteorológica. A continuación se muestra una captura de pantalla de esta vista:

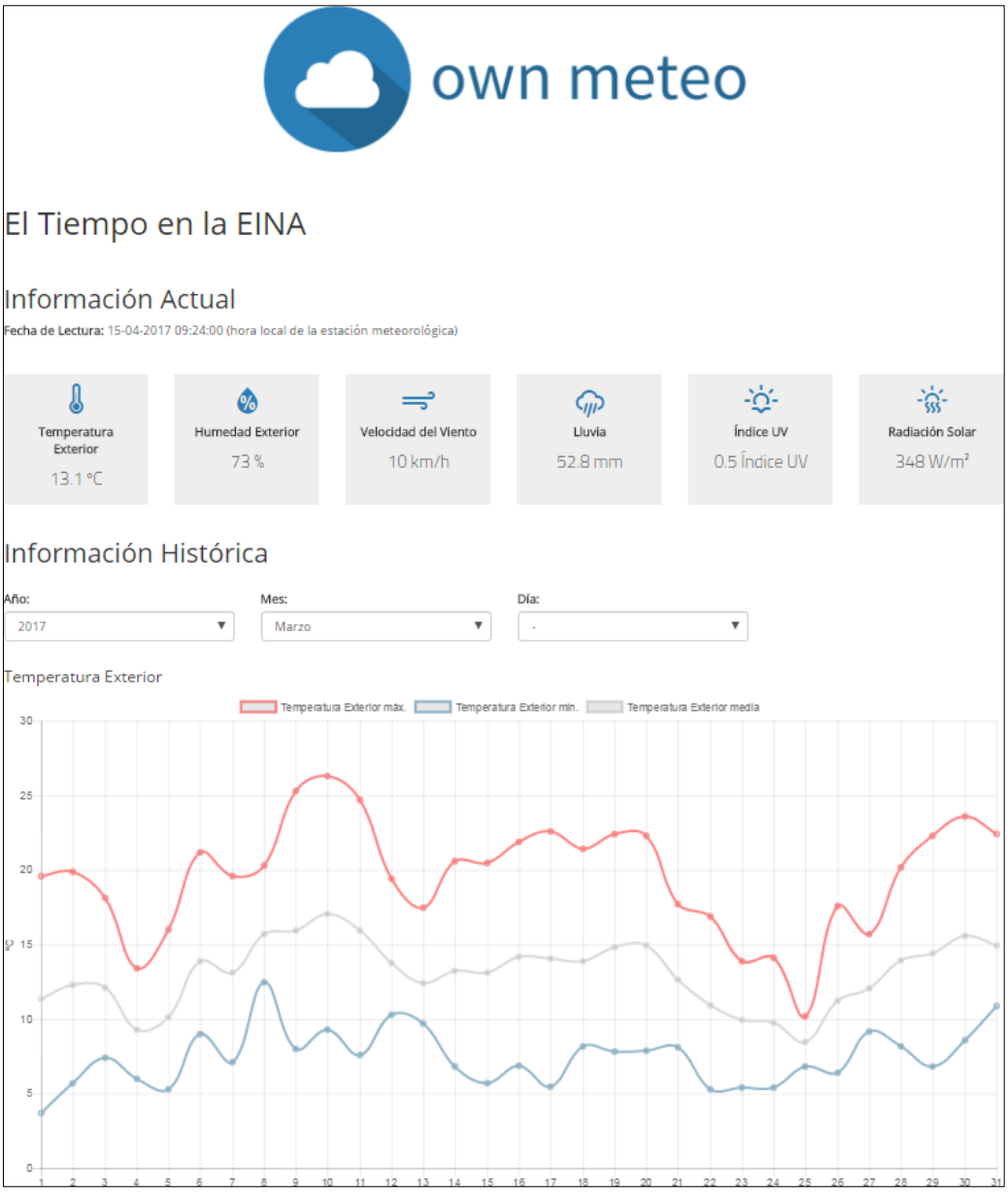

Figura 34: Captura de la vista de información meteorológica histórica.

Esta vista utiliza el controlador `weather`, el cual realiza las siguientes tareas:

- Obtener los datos de la publicación meteorológica: nombre, datos meteorológicos mostrados…
- Obtener la lectura de información meteorológica más reciente para la estación vinculada a la publicación.

- Obtener la información meteorológica histórica en función de los parámetros indicados por el usuario en los desplegables de *Año*, *Mes* y *Día*.

C.2. Usuarios Autenticados como Usuarios no Administradores

Ahora, se procede a describir las vistas que requieren que el usuario esté autenticado como usuario no administrador en el API.

C.2.1. Vista de Estaciones Meteorológicas de Usuario

Esta vista muestra al usuario las estaciones meteorológicas de su propiedad. Es decir, muestra un listado con las estaciones meteorológicas registradas por el propio usuario en la plataforma. A continuación se muestra una captura de esta vista:

Figura 35: Captura de la vista de estaciones meteorológicas de usuario.

La vista hace uso del controlador `userStations`, que principalmente se encarga de obtener el listado de estaciones meteorológicas del usuario.

C.2.2. Vista de Creación de Estación Meteorológica de Usuario

Esta vista contiene un formulario donde el usuario puede introducir los datos para registrar una nueva estación meteorológica en la plataforma. En la figura 36 se muestra una captura de pantalla de esta vista.

La vista utiliza el controlador `newStations`, encargado de:

- Obtener el listado de modelos de estación meteorológica.
- Esperar a que el usuario introduzca el modelo de la estación meteorológica que desea registrar. Una vez el usuario introduzca el modelo, el controlador descarga la información de dicho modelo y muestra al usuario nuevos campos en el formulario, a rellenar con la información requerida por dicho modelo (más concretamente por el *parser* asociado a dicho modelo).
- Registrar la nueva estación meteorológica enviando los datos introducidos por el usuario al API REST.
- Si la estación meteorológica se crea correctamente, redirigir al usuario a la vista de detalle de dicha estación meteorológica, descrita más adelante en este Anexo.

Figura 36: Captura de la vista de creación de estación meteorológica de usuario.

C.2.3. Vista de Detalle de Estación Meteorológica de Usuario

Principalmente, esta vista muestra al usuario los detalles de la estación meteorológica indicada. Además, esta vista permite al usuario:

- Editar los datos de su estación meteorológica pulsando el botón *Editar*.
- Eliminar (realmente desactivar, ya que las estaciones meteorológicas solo pueden ser eliminadas por usuarios administradores) la estación meteorológica pulsando el botón *Eliminar Estación*.
- Crear una nueva publicación de información meteorológica asociada a la propia estación pulsando el botón *Crear Publicación*.
- Ver las publicaciones de información meteorológica dependientes de la propia estación.

En la figura 37 se muestra una captura de esta vista.

La vista utiliza los controladores `stationDetail` y `stationPosts`. El primero de ellos es el encargado de:

- Obtener la información de la estación meteorológica.
- Obtener el listado de modelos de estación meteorológica.
- En el modo de edición, esperar a que el usuario introduzca un nuevo modelo de estación meteorológica. Si lo hace (es decir, cambia el modelo de su estación), el controlador descarga la información del nuevo modelo y muestra al usuario nuevos campos en el formulario, a rellenar con la información requerida por dicho modelo (más concretamente por el *parser* asociado a dicho modelo).
- En el modo de edición, guardar los cambios introducidos por el usuario si este así lo indica (pulsando el botón *Guardar*).
- Eliminar (realmente desactivar) la estación meteorológica si así se lo indica el usuario.

El controlador `stationPosts` únicamente se encarga de obtener el listado de publicaciones de información meteorológica asociados a la propia estación.

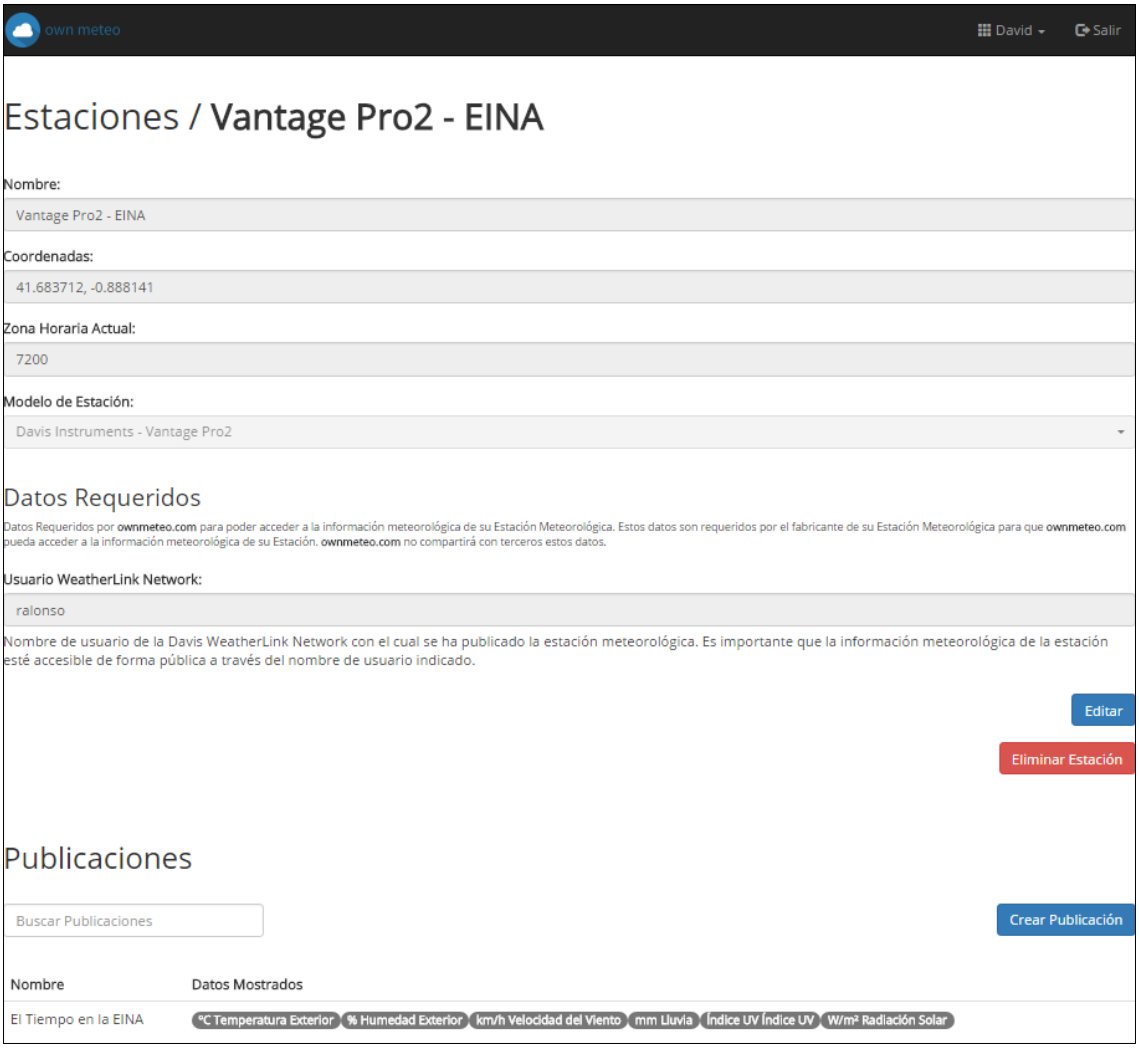

Figura 37: Captura de la vista de detalle de estación meteorológica de usuario.

C.2.4. Vista de Publicaciones de Información Meteorológica de Usuario

Esta vista muestra al usuario las publicaciones de información meteorológica vinculadas a sus estaciones. En la figura 38 puede apreciarse una captura de pantalla de la vista.

La vista utiliza el controlador `userPosts`, el cual se encarga principalmente de obtener (a través del API REST) el conjunto de publicaciones de información meteorológica vinculadas a las estaciones del usuario. Puesto que las publicaciones no están vinculadas al usuario, sino a las estaciones meteorológicas, el controlador debe obtener en primera instancia las estaciones meteorológicas del usuario y, a continuación, las publicaciones vinculadas a cada una de las estaciones del usuario.

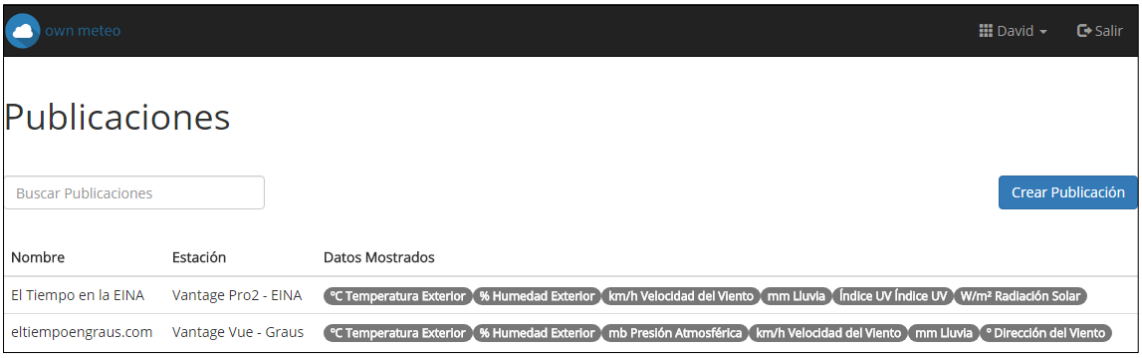

Figura 38: Captura de la vista de publicaciones de información meteorológica de usuario.

C.2.5. Vista de Creación de Publicación de Información Meteorológica de Usuario

Esta vista contiene un formulario donde el usuario puede introducir los datos para crear una publicación de información meteorológica. A continuación se muestra una captura de pantalla de esta vista:

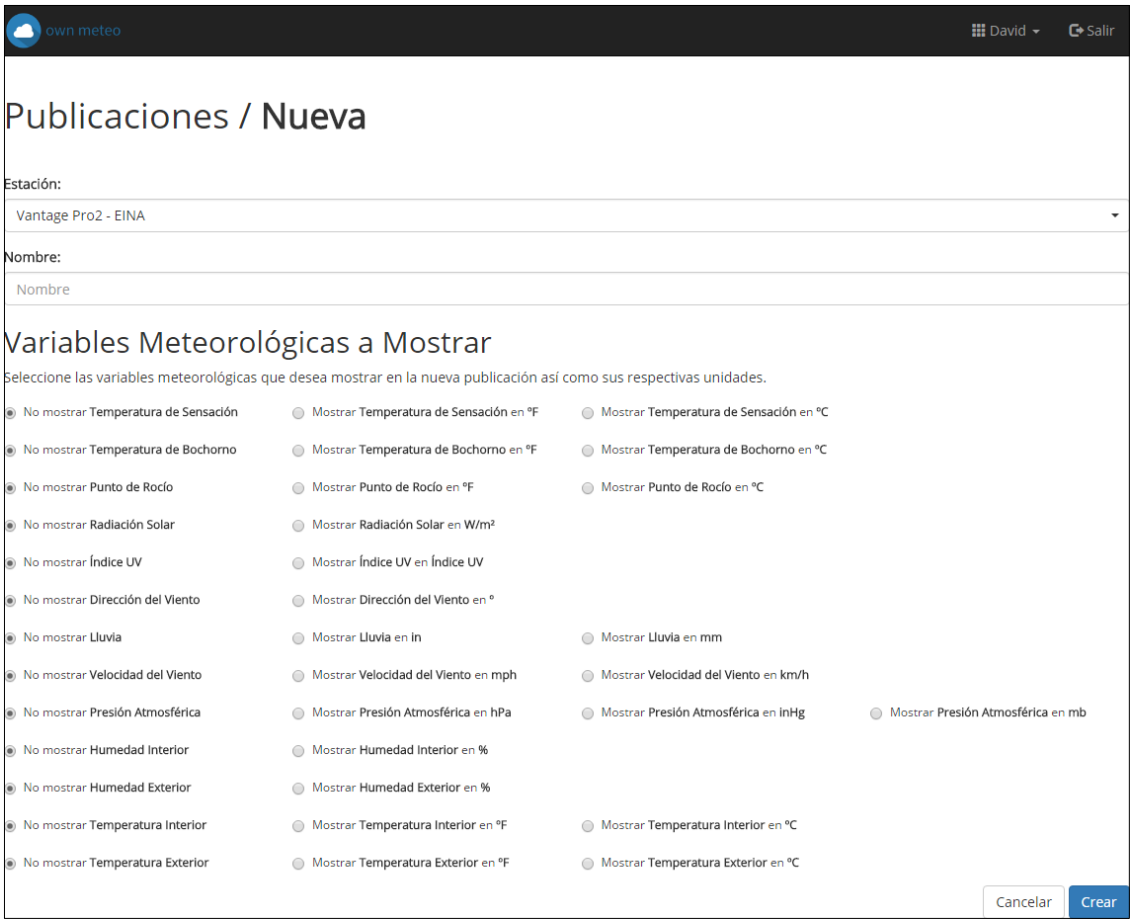

Figura 39: Captura de la vista de creación de publicación de información meteorológica de usuario.

La vista utiliza el controlador `newPosts`, encargado de:

- Obtener el listado de estaciones meteorológicas del usuario.
- Esperar a que el usuario seleccione la estación meteorológica a la cual estará vinculada la nueva publicación. Una vez seleccionada, el controlador consultará al API el listado de variables meteorológicas leídas por la estación seleccionada. Con este listado, el

controlador mostrará al usuario las posibles variables meteorológicas a mostrar por la nueva publicación.

- En el momento que el usuario pulse el botón *Crear*, el controlador se encargará de enviar los datos introducidos al API para crear la nueva publicación.
- Si la publicación se crea correctamente, el controlador redirigirá al usuario a la vista de detalle de publicación de información meteorológica, explicada en el siguiente apartado.

C.2.6. Vista de Detalle de Publicación de Información Meteorológica de Usuario

Principalmente, esta vista muestra al usuario los detalles de la publicación de información meteorológica indicada. Además, esta vista permite al usuario:

- Editar los datos de su publicación pulsando el botón *Editar*.
- Eliminar (realmente desactivar, ya que las publicaciones solo pueden ser eliminadas por usuarios administradores) la publicación pulsando el botón *Eliminar Publicación*.
- Obtener el elemento HTML a incrustar en sitios web terceros para compartir la publicación pulsando el botón *Insertar Publicación*.
- Ver las variables meteorológicas de la publicación pulsando el botón *Ver Publicación*.

A continuación se muestra una captura de esta vista:

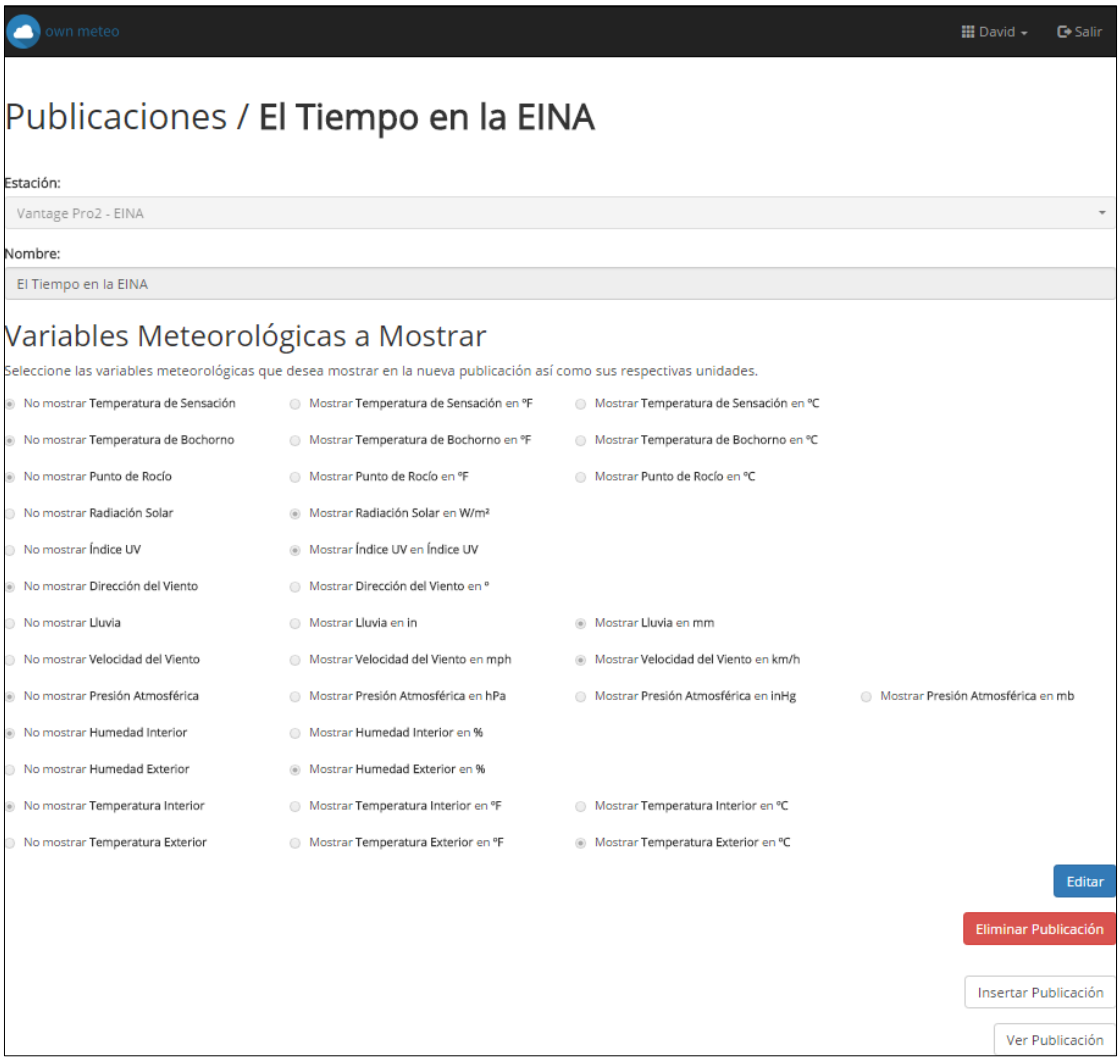

Figura 40: Captura de la vista de detalle de publicación de información meteorológica de usuario.

Esta vista utiliza el controlador `postDetail`, encargado de:

- Obtener la información de la publicación de información meteorológica.
- Obtener el listado de estaciones meteorológicas del usuario.
- En el modo de edición, si el usuario cambia la estación vinculada a la publicación, descargar el listado de variables meteorológicas leídas por la nueva estación.
- En el modo de edición, guardar los cambios introducidos por el usuario si este así lo indica (pulsando el botón *Guardar*).
- Eliminar (realmente desactivar) la publicación si así se lo indica el usuario.
- Generar el elemento HTML a incrustar en sitios web terceros para la compartición de la publicación.

C.2.7. Vista de Detalle de Usuario

Esta vista muestra al usuario sus datos personales y le ofrece la posibilidad de editarlos. Además, también permite eliminar (realmente desactivar, ya que un usuario no administrador no puede eliminarse a sí mismo) el usuario. A continuación se muestra una captura de esta vista:

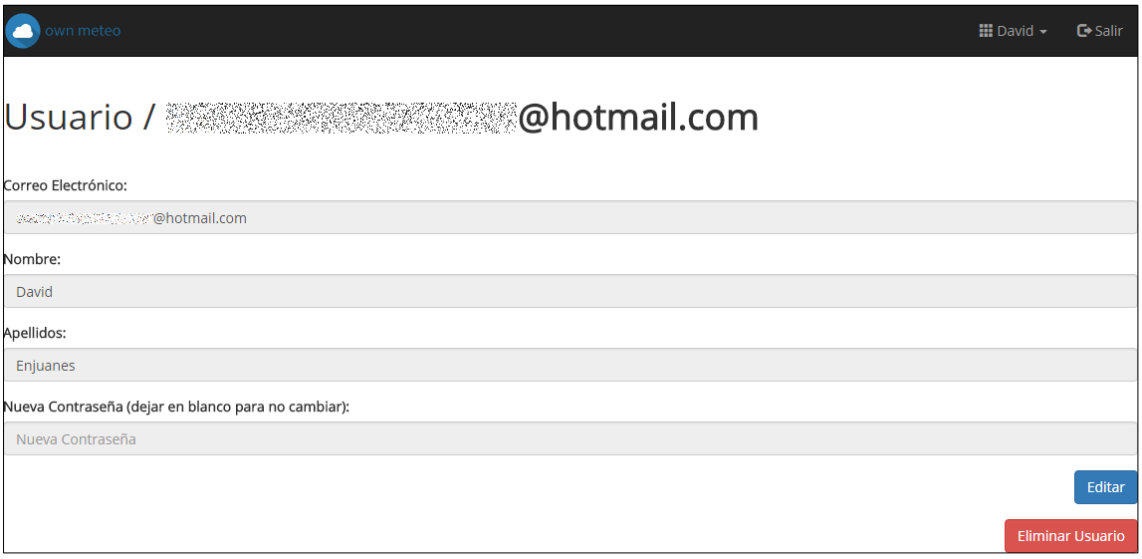

Figura 41: Captura de la vista de detalle de información de usuario.

El controlador utilizado por esta vista es el controlador `detailUser`. Este controlador se encarga de:

- Obtener los datos del usuario.
- Actualizar los datos del usuario si el usuario así lo indica en el modo de edición al pulsar el botón *Guardar*.
- Eliminar (desactivar) el usuario si el usuario así lo indica pulsando el botón *Eliminar Usuario*.

C.3. Usuarios Autenticados como Usuarios Administradores

A continuación se describen las vistas accesibles únicamente por usuarios autenticados como usuarios administradores en el API REST.

C.3.1. Panel de Administración

Esta vista contiene enlaces a las principales vistas accesibles por usuarios administradores. Además, contiene un listado con las incidencias más recientes acontecidas en la plataforma. A continuación se muestra una captura de esta vista:

Figura 42: Captura del panel de administración.

La vista utiliza el controlador `adminPanel` encargado de obtener el listado de incidencias más recientes.

C.3.2. Vista de Usuarios

Esta vista contiene el listado de usuarios registrados en la plataforma, tanto activos como no activos. En la figura 43 se muestra una captura de la vista.

La vista utiliza el controlador `adminUsers`, el cual es el encargado de obtener el listado de usuarios y clasificarlos según su estado: activo o no.

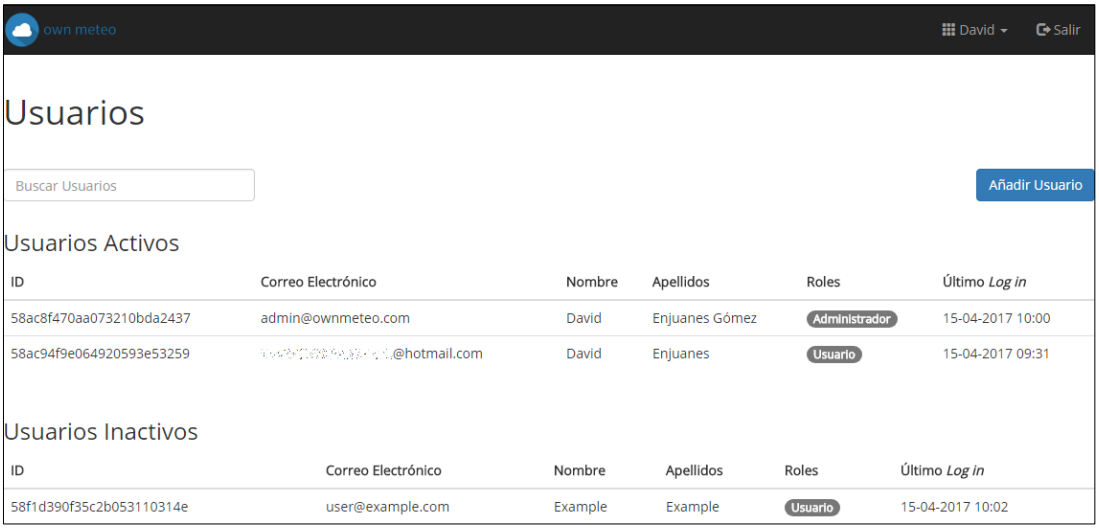

Figura 43: Captura de la vista de usuarios.

C.3.3. Vista de Creación de Usuario de Administrador

Esta vista permite al usuario administrador crear nuevos usuarios, tanto administradores como no administradores. A continuación se muestra una captura de la vista:

Figura 44: Captura de la vista de creación de usuario de administrador.

La vista hace uso del controlador `adminNewUser`, que se encarga de crear el nuevo usuario a partir de los datos introducidos por el administrador en el formulario de la vista.

C.3.4. Vista de Detalle de Usuario de Administrador

Esta vista muestra los detalles del usuario indicado. En la figura 45 se muestra una captura de esta vista.

La vista utiliza el controlador `detailUser`, utilizado también en la vista de detalle de usuario (vista en la cual los usuarios no administradores podían modificar sus propios datos). El

controlador realiza las mismas tareas que en dicha vista, pero, adicionalmente en esta vista permite:

- Cambiar el rol del usuario.
- Eliminar definitivamente al usuario.

Figura 45: Captura de la vista de detalle de usuario de administrador.

C.3.5. Vista de *Parsers*

Esta vista muestra el listado de *parsers* del sistema. A continuación se muestra una captura de la vista:

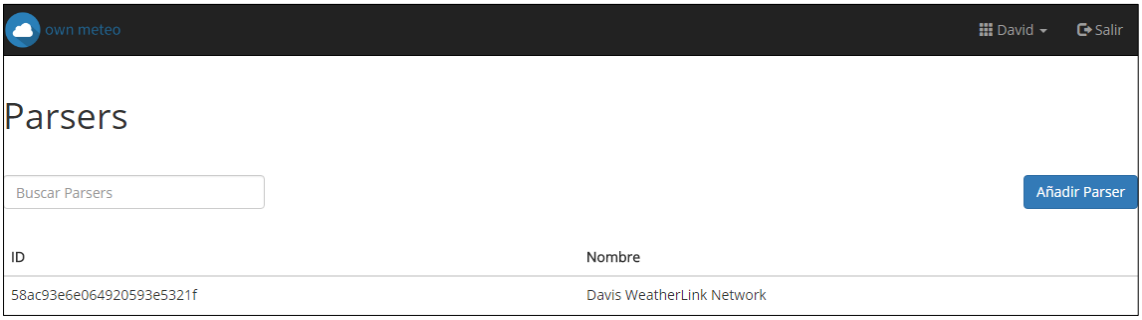

Figura 46: Captura de la vista de *parsers*.

La vista utiliza el controlador `adminParsers`, encargado de obtener el listado de *parsers* del sistema a través del API REST.

C.3.6. Vista de Creación de *Parser*

En esta vista se permite al usuario administrador crear un nuevo *parser*. En la figura 47 se muestra una captura de la vista.

La vista utiliza el controlador `adminNewParser`, encargado de tomar los datos del *parser* introducidos por el administrador (tanto datos como código fuente) y enviarlos al API para crear el nuevo *parser*. Es importante destacar que el controlador debe asegurarse que el administrador ha subido un fichero ZIP en el apartado de código fuente del *parser*. Dicho ZIP deberá ser convertido por el controlador a una cadena de texto, cuyo contenido sea el ZIP codificado en Base 64, de forma que pueda ser enviado al API.

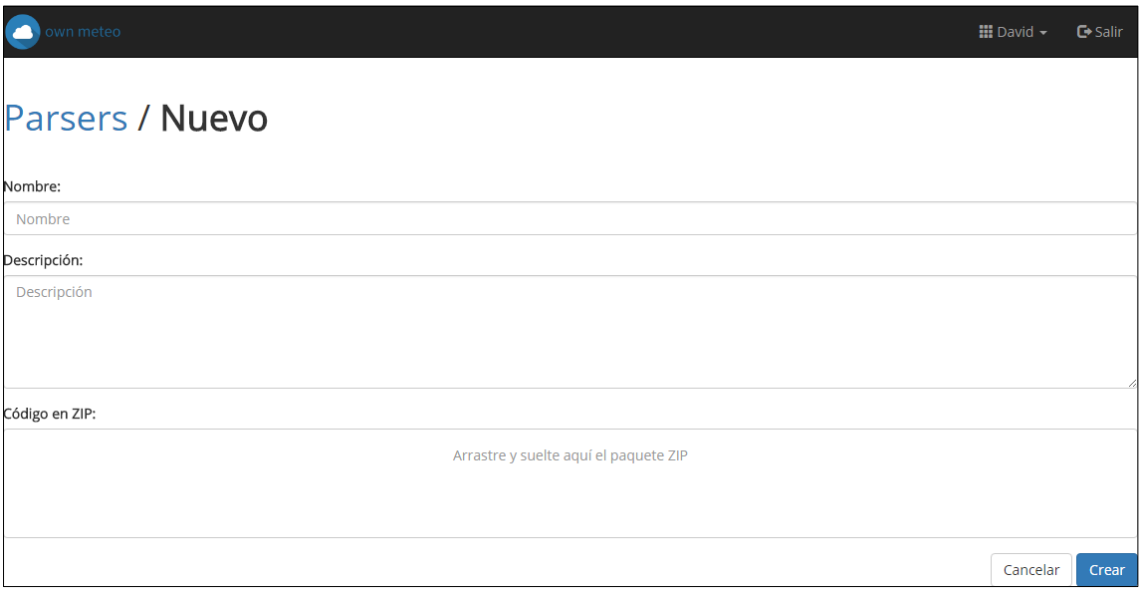

Figura 47: Captura de la vista de creación de *parser*.

C.3.7. Vista de Detalle de *Parser*

Esta vista muestra al administrador los detalles del *parser* indicado. La vista también permite:

- Descargar el código fuente del *parser*.
- Modificar los datos del *parser* (tanto datos como código fuente).
- Ver los datos requeridos por el *parser*.
- Eliminar datos requeridos por el *parser*.

En la figura 48 se muestra una captura de esta vista.

Esta vista hace uso del controlador `adminParserDetail`, encargado de:

- Obtener los datos del *parser*.
- Descargar el código fuente del *parser* si así lo solicita el administrador.
- En el modo de edición, si el administrador pulsa el botón *Guardar*, enviar los nuevos datos del *parser* introducidos por el administrador (tanto datos como código).
- Obtener el listado de datos requeridos por el *parser*.
- Si el administrador así lo indica pulsando el icono de papelera junto al dato requerido deseado, eliminar dicho dato requerido.

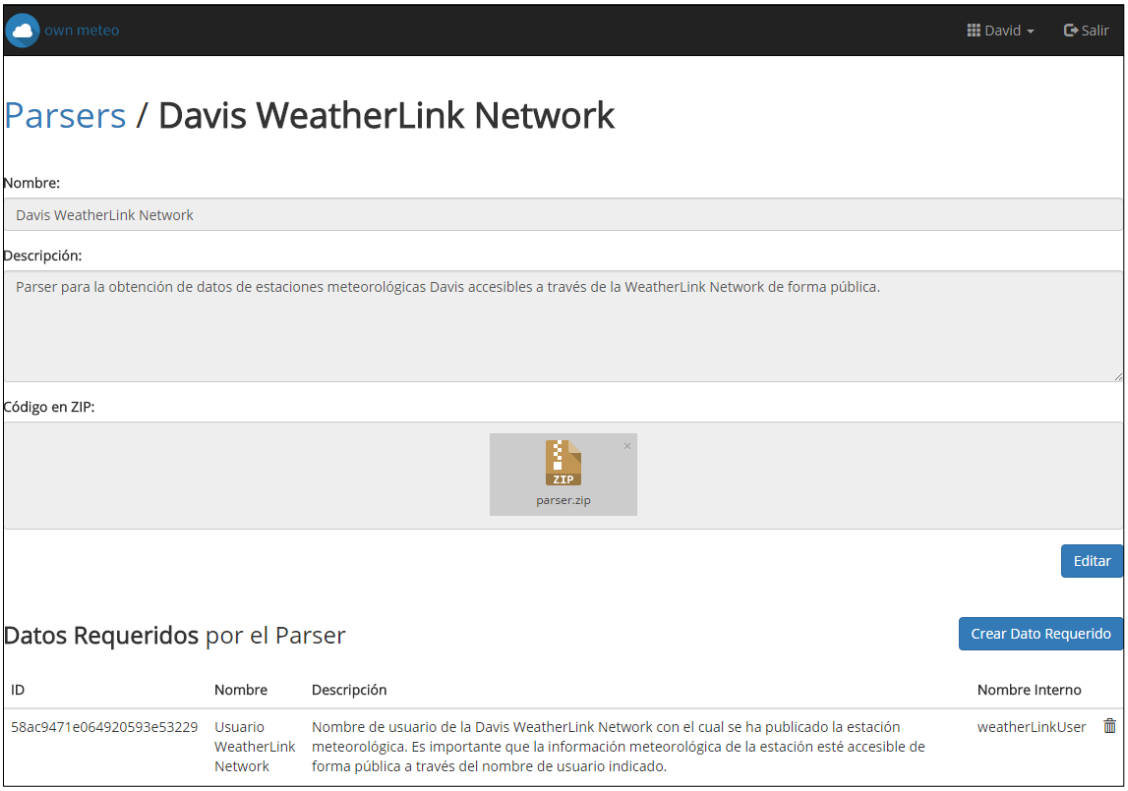

Figura 48: Captura de la vista de detalle de *parser*.

C.3.8. Vista de Creación de Dato Requerido por *Parser*

Esta vista permite al administrador crear un nuevo dato requerido por *parser*. A continuación se muestra una captura de la vista:

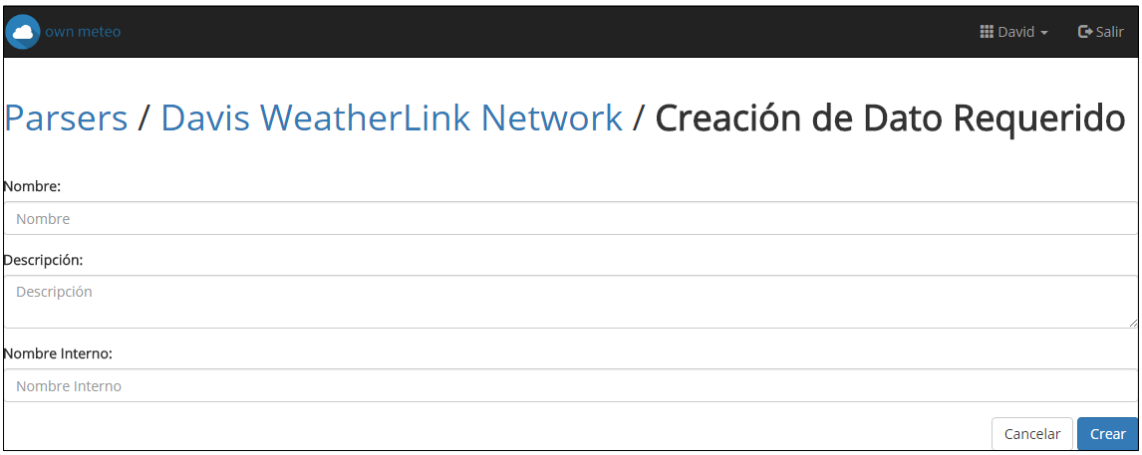

Figura 49: Captura de la vista de creación de dato requerido por parser.

La vista hace uso del controlador `adminNewReqParser`, encargado de crear el dato requerido por *parser* con los datos introducidos por el administrador.

C.3.9. Vista de Modelos de Estación Meteorológica

Esta vista muestra al administrador el listado de modelos de estación reconocidos por la plataforma. A continuación se muestra una captura de la vista:

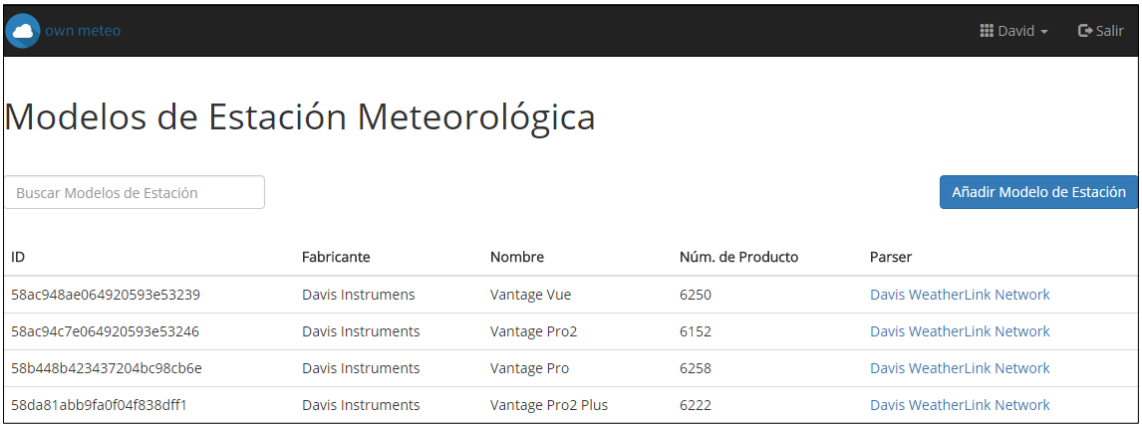

Figura 50: Captura de la vista de modelos de estación meteorológica.

La vista hace uso del controlador `adminStationsModel`, encargado de obtener el listado de modelos de estación meteorológica.

C.3.10. Vista de Creación de Modelo de Estación Meteorológica

Esta vista permite al administrador crear un nuevo modelo de estación meteorológica. En la figura 51 se muestra una captura de esta vista.

La vista utiliza el controlador `adminNewStationModel`, encargado de:

- Descargar el listado de *parsers* de forma que el administrador pueda seleccionar el *parser* asociado al nuevo modelo de estación.
- En el momento que el usuario pulse el botón *Crear*, enviar los datos introducidos por el administrador al API para crear el nuevo modelo de estación.

Figura 51: Captura de la vista de creación de modelo de estación meteorológica.

C.3.11. Vista de Detalle de Modelo de Estación Meteorológica

Esta vista permite al administrador ver y editar los detalles del modelo de estación seleccionado. A continuación se muestra una captura de la vista:

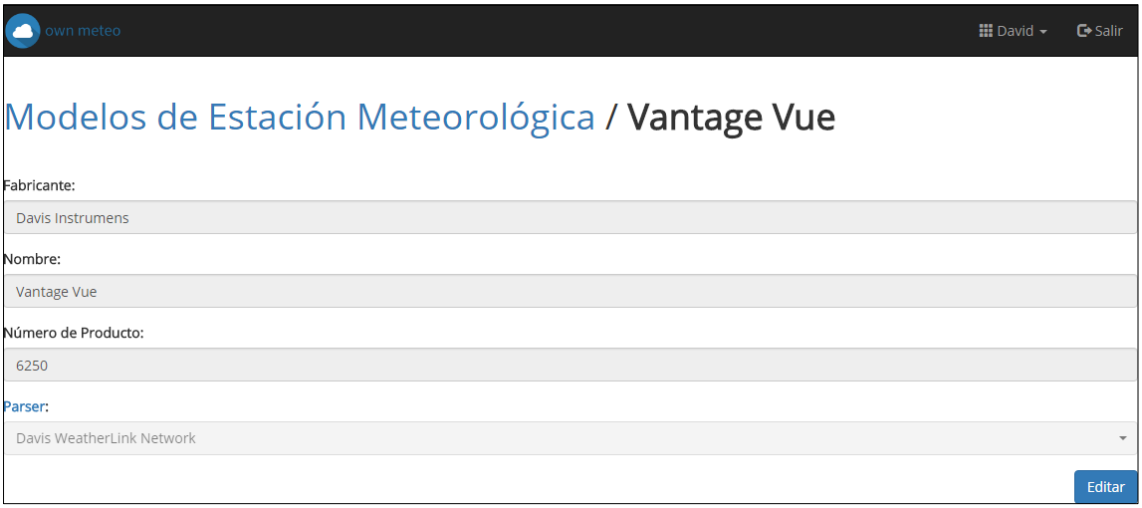

Figura 52: Captura de la vista de detalle de modelo de estación meteorológica.

La vista hace uso del controlador `adminStationModelsDetail`, encargado de guardar los nuevos datos del modelo si el administrador los editase.

C.3.12. Vista de Estaciones Meteorológicas de Administrador

Esta vista muestra al administrador el conjunto total de estaciones meteorológicas registradas en la plataforma (por todos sus usuarios). A continuación se muestra una captura de esta vista:

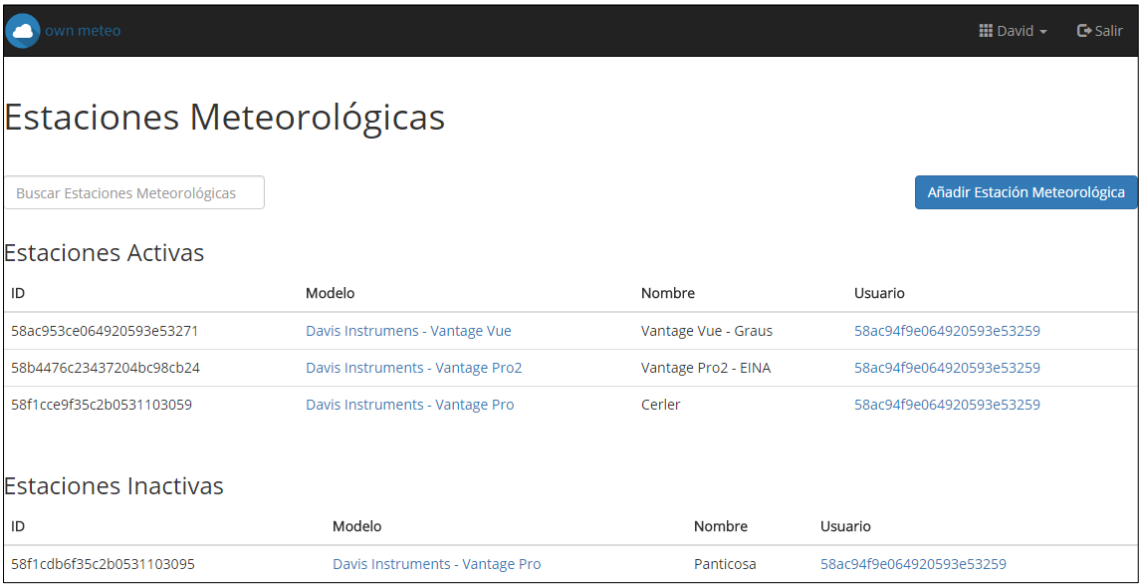

Figura 53: Captura de la vista de estaciones meteorológicas de administrador.

El controlador utilizado por esta vista es el controlador `adminStations`, cuyo objetivo es el de descargar el listado completo de estaciones meteorológicas de la plataforma.

C.3.13. Vista de Creación de Estación Meteorológica de Administrador

Esta vista permite al administrador crear una nueva estación meteorológica. A continuación puede apreciarse una captura de la vista:

Figura 54: Captura de la vista de creación de estación meteorológica de administrador.

La vista hace uso del controlador `newStations`, utilizado también en la vista de creación de estación meteorológica de usuario (descrita previamente). De hecho, la vista es igual que la vista de creación de estación meteorológica de usuario con la diferencia de que en este caso se le permite al administrador introducir un ID de usuario, que será el propietario de la estación creada.

C.3.14. Vista de Detalle de Estación Meteorológica de Administrador

Esta vista tiene la misma funcionalidad que la vista de detalle de estación meteorológica de usuario. La principal diferencia radica en que el usuario administrador puede ver estaciones meteorológicas de cualquier usuario, por lo que en esta vista aparece el ID del usuario propietario de la estación. Además, esta vista permite al administrador eliminar definitivamente la estación meteorológica. En la figura 55 se puede apreciar una captura de esta vista.

Al igual que la vista de detalle de estación meteorológica de usuario, esta vista utiliza los controladores `stationDetail` y `stationPosts`.

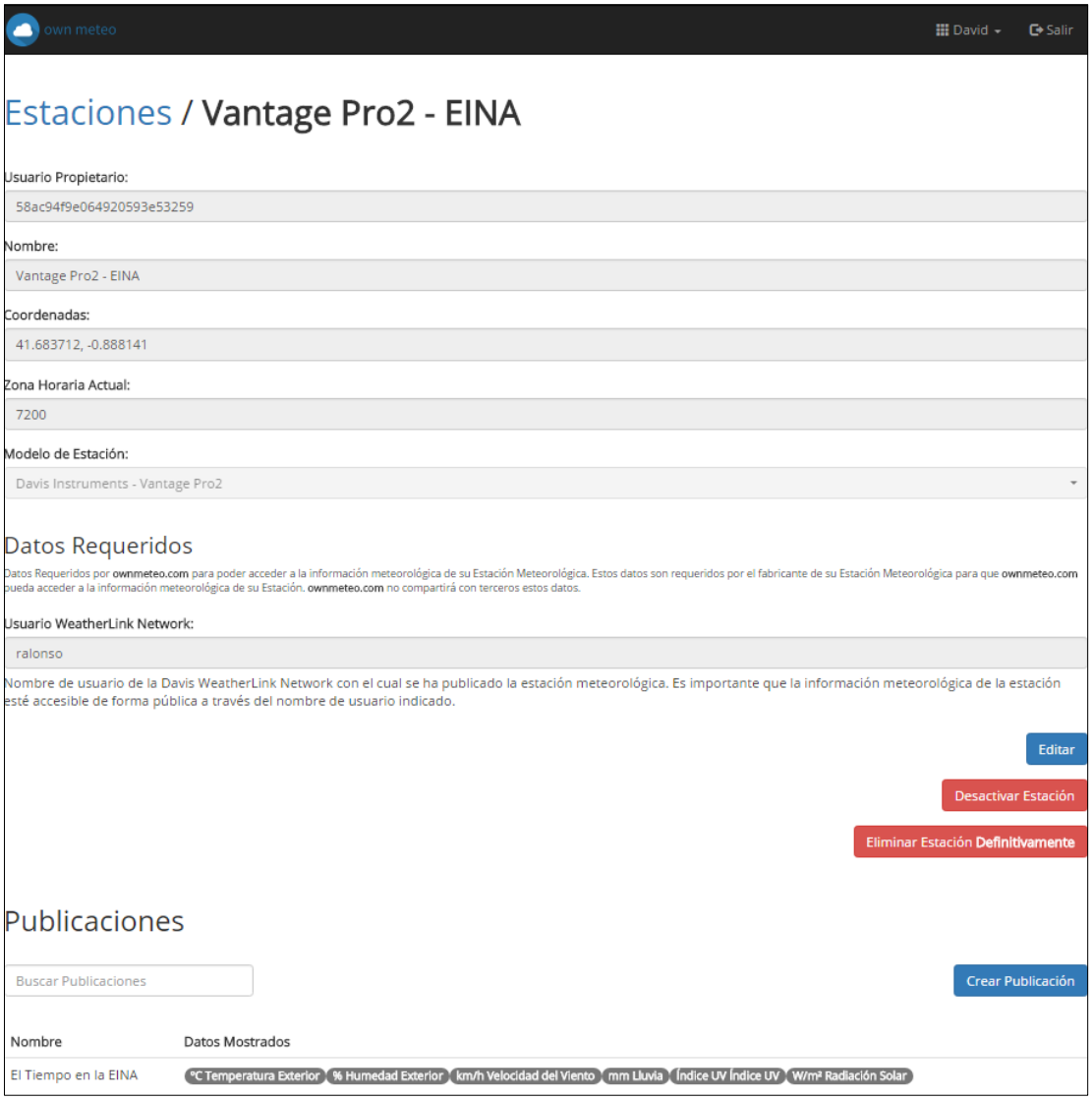

Figura 55: Captura de la vista de detalle de estación meteorológica de administrador.

C.3.15. Vista de Publicaciones de Información Meteorológica de Administrador

Esta vista muestra al administrador el conjunto total de publicaciones de información meteorológica existentes en la plataforma. A continuación se muestra una captura de esta vista:

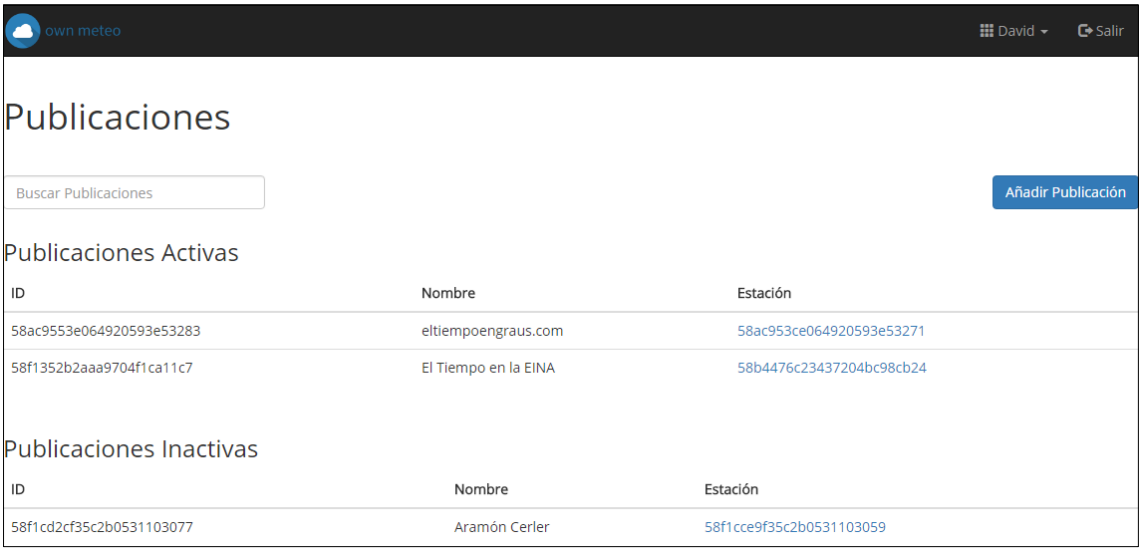

Figura 56: Captura de la vista de publicaciones de información meteorológica de administrador.

El controlador utilizado por esta vista es el controlador `adminPosts`, cuyo objetivo es el de descargar el listado completo de publicaciones de información meteorológica de la plataforma.

C.3.16. Vista de Creación de Publicación de Inf. Meteorológica de Administrador

Esta vista permite al administrador crear una nueva publicación de información meteorológica. A continuación puede apreciarse una captura de la vista:

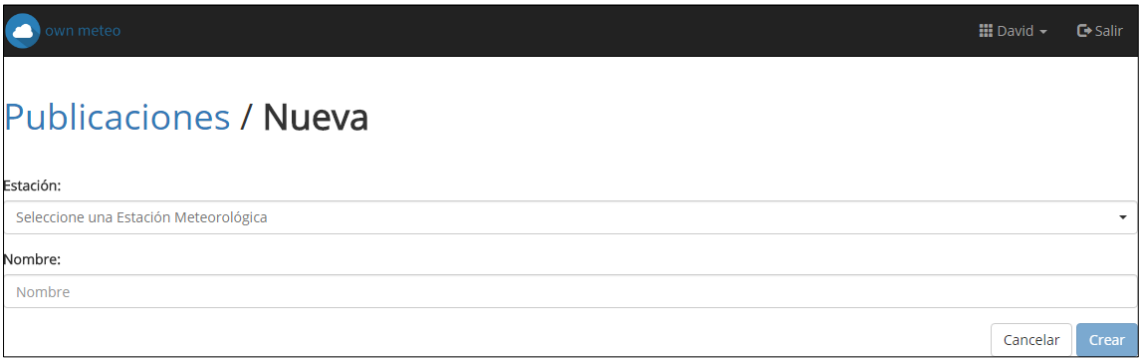

Figura 57: Captura de la vista de creación de publicación de información meteorológica de administrador.

La vista hace uso del controlador `newPosts`, utilizado también en la vista de creación de publicación de información meteorológica de usuario (descrita previamente). La única diferencia radica en que, en esta vista, el controlador descargará la lista completa de estaciones meteorológicas de la plataforma, permitiendo así al administrador crear una publicación asociada a cualquier estación meteorológica (de cualquier usuario).

C.3.17. Vista de Detalle de Publicación de Inf. Meteorológica de Administrador

Esta vista tiene la misma funcionalidad que la vista de detalle de publicación de información meteorológica de usuario. La principal diferencia radica en que el usuario administrador puede vincular la publicación a cualquier estación de la plataforma. Además, esta vista permite al administrador eliminar definitivamente la publicación. A continuación se muestra una captura de la vista:

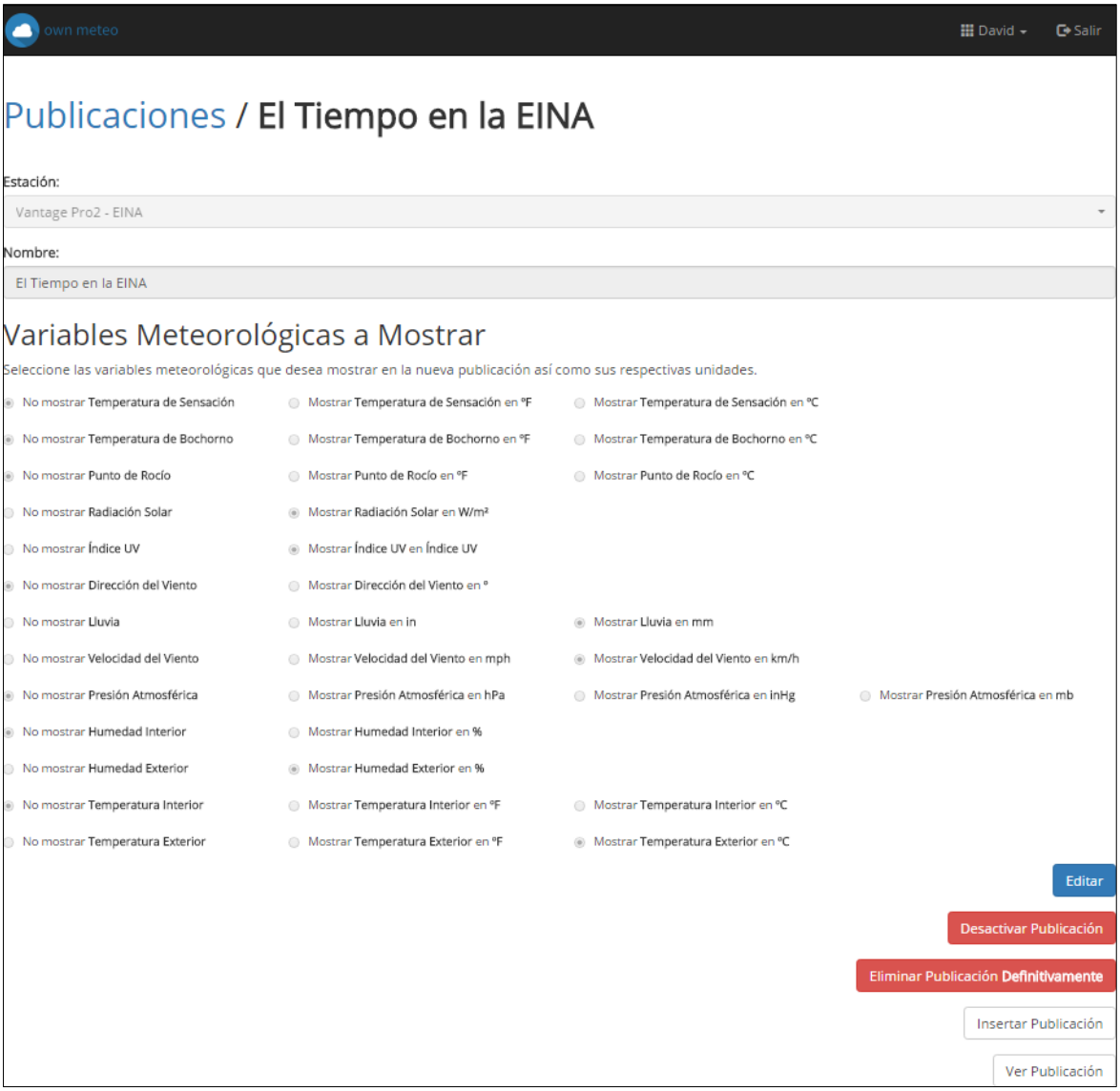

Figura 58: Captura de la vista de detalle de publicación de información meteorológica de administrador.

Al igual que la vista de detalle de publicación de información meteorológica de usuario, esta vista utiliza el controlador `postDetail`.

C.3.18. Vista de Accesos al API

Muestra al administrador los accesos al API REST. Los accesos son mostrados de más a menos reciente, y pueden ser filtrados por fecha mínima y máxima. En la figura 59 se puede apreciar una captura de esta vista.

La vista hace uso del controlador `adminApiAccesses`, encargado de obtener el listado de accesos al API filtrado por los parámetros introducidos por el administrador.

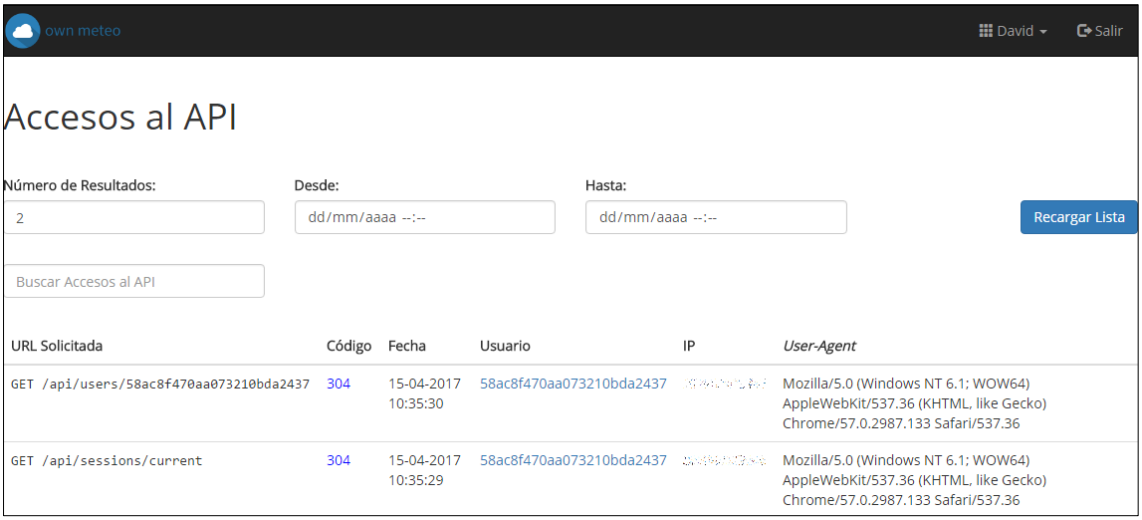

Figura 59: Captura de la vista de accesos al API.

C.3.19. Vista de Incidencias

Esta vista muestra al administrador las incidencias acontecidas en el sistema. Al igual que en la vista anterior, las incidencias pueden ser filtradas según su fecha. A continuación se muestra una captura de la vista:

Figura 60: Captura de la vista de incidencias.

Esta vista hace uso del controlador `adminIncidences`, encargado de obtener el listado de incidencias según los filtros indicados por el administrador.

Anexo D - Juego de Pruebas

En este Anexo se describen todas las pruebas contenidas en el juego de pruebas desarrollado para la plataforma. Como ya se ha comentado previamente, estas pruebas verifican el correcto funcionamiento de las vistas de la aplicación web (y en consecuencia, del API REST). Es por esta razón que las pruebas son presentadas a continuación clasificadas según la vista donde se ejecutan:

D.1. Vistas sin Autenticación

En este apartado se describen las pruebas realizadas sobre las vistas que no requieren ningún tipo autenticación.

D.1.1. Vista de Registro

A continuación se muestra una tabla donde se pueden apreciar las pruebas desarrolladas para esta vista así como el resultado previsto para las mismas:

Prueba	Resultado previsto
Registro de un nuevo usuario utilizando un correo electrónico ya en uso por otro usuario.	No se registra el nuevo usuario y se muestra un mensaje de error explicando lo acontecido.
Registro de un usuario dejando campos en blanco.	No se registra el nuevo usuario y se muestra un mensaje de error explicando que todos los campos deben ser rellenados.
Registro válido.	Se registra el nuevo usuario y el navegador muestra la vista de estaciones meteorológicas del nuevo usuario.

Tabla 7: Pruebas para la vista de registro.

D.1.2. Vista de Entrar/*Log in*

A continuación se muestra una tabla donde se pueden apreciar las pruebas desarrolladas para esta vista así como el resultado previsto para las mismas:

Prueba	Resultado previsto
Log In de un usuario administrador válido.	Se crea la nueva sesión y el navegador muestra el panel de administración.
Log In de un usuario no administrador válido.	Se crea la nueva sesión y el navegador muestra la vista de estaciones meteorológicas del usuario.
Log In con credenciales inválidos.	No se crea la sesión y se muestra un mensaje de error explicando lo acontecido.
Log In dejando campos en blanco.	No se crea la sesión y se muestra un mensaje de error explicando que todos los campos deben ser rellenados.

Tabla 8: Pruebas para la vista de entrar/log in.

D.2. Vistas de Usuarios Autenticados como Usuarios no Administradores

En este apartado se describen las pruebas realizadas sobre las vistas que requieren autenticación como usuario no administrador:

D.2.1. Vista de Creación de Estación Meteorológica de Usuario

A continuación se muestra una tabla donde se pueden apreciar las pruebas desarrolladas para esta vista así como el resultado previsto para las mismas:

Prueba	Resultado previsto
Creación de estación meteorológica dejando campos en blanco.	No se crea la estación meteorológica y se muestra un mensaje de error explicando que todos los campos deben ser rellenados.
Creación de estación meteorológica introduciendo datos requeridos (por el *parser*) inválidos.	No se crea la estación meteorológica y se muestra un mensaje de error explicando lo acontecido.
Creación de estación meteorológica válida.	Se crea la nueva estación y el navegador muestra la vista de detalle de la nueva estación meteorológica.

Tabla 9: Pruebas para la vista de creación de estación meteorológica de usuario.

D.2.2. Vista de Creación de Publicación de Información Meteorológica de Usuario

A continuación se muestra una tabla donde se pueden apreciar las pruebas desarrolladas para esta vista así como el resultado previsto para las mismas:

Prueba	Resultado previsto
Creación de publicación dejando campos en blanco.	No se crea la publicación y se muestra un mensaje de error explicando que todos los campos deben ser rellenados.
Creación de publicación válida.	Se crea la nueva publicación y el navegador muestra la vista de detalle de la nueva publicación.

Tabla 10: Pruebas para la vista de creación de publicación de información meteorológica de usuario.

D.2.3. Vista de Detalle de Usuario

A continuación se muestra una tabla donde se pueden apreciar las pruebas desarrolladas para esta vista así como el resultado previsto para las mismas:

Prueba	Resultado previsto
Modificación del correo electrónico del usuario por uno utilizado ya por otro usuario.	No se lleva a cabo la modificación y se muestra un mensaje de error explicando lo acontecido.
Modificación de los datos del usuario dejando campos en blanco (diferentes al de contraseña, ya que si esta no se desea cambiar se deja en blanco).	No se lleva a cabo la modificación y se muestra un mensaje de error explicando que todos los campos deben ser rellenados excepto el de contraseña si esta no se desea cambiar.

Modificación de los datos del usuario válida dejando el campo de contraseña en blanco.	La modificación se lleva a cabo y se muestra un mensaje de éxito.
Modificación de los datos del usuario válida sin dejar en blanco el campo de contraseña.	La modificación se lleva a cabo y se muestra un mensaje de éxito.

Tabla 11: Pruebas para la vista de detalle de usuario.

D.2.4. Vista de Detalle de Publicación de Información Meteorológica de Usuario

A continuación se muestra una tabla donde se pueden apreciar las pruebas desarrolladas para esta vista así como el resultado previsto para las mismas:

Prueba	Resultado previsto
Modificación de la publicación dejando campos en blanco.	No se lleva a cabo la modificación y se muestra un mensaje de error explicando que todos los campos deben ser rellenados.
Modificación de la publicación válida.	La modificación se lleva a cabo y se muestra un mensaje de éxito.
Desactivación de la publicación.	La desactivación se lleva a cabo y se muestra un mensaje de éxito. Tras ello el navegador redirige al usuario a su panel de publicaciones.

Tabla 12: Pruebas para la vista de detalle de publicación de información meteorológica de usuario.

D.2.5. Vista de Detalle de Estación Meteorológica de Usuario

A continuación se muestra una tabla donde se pueden apreciar las pruebas desarrolladas para esta vista así como el resultado previsto para las mismas:

Prueba	Resultado previsto
Modificación de la estación dejando campos en blanco.	No se lleva a cabo la modificación y se muestra un mensaje de error explicando que todos los campos deben ser rellenados.
Modificación de la estación válida.	La modificación se lleva a cabo y se muestra un mensaje de éxito.
Desactivación de la estación.	La desactivación se lleva a cabo y se muestra un mensaje de éxito. Tras ello el navegador redirige al usuario a su panel de estaciones.

Tabla 13: Pruebas para la vista de detalle de estación meteorológica de usuario.

D.3. Vistas de Usuarios Autenticados como Usuarios Administradores

En este apartado se describen las pruebas realizadas sobre las vistas que requieren autenticación como usuario administrador:

D.3.1. Vista de Detalle de Usuario de Administrador

Como ya se ha comentado en el Anexo C, esta vista permite modificar los datos del usuario indicado. Puesto que esta vista comparte controlador con la vista de detalle de usuario (sobre la cual ya se han propuesto pruebas), las pruebas descritas en este apartado complementan a las propuestas para dicha vista.

A continuación se muestra una tabla donde se pueden apreciar las pruebas desarrolladas para esta vista así como el resultado previsto para las mismas:

Prueba	Resultado previsto
Desactivación del usuario.	La desactivación se lleva a cabo, se muestra un mensaje de éxito, se oculta el botón de desactivación y se muestra el botón de activación de usuario.
Activación del usuario.	La activación se lleva a cabo, se muestra un mensaje de éxito, se oculta el botón de activación y se muestra el botón de desactivación de usuario.
Modificación del rol de usuario a usuario administrador.	La modificación se lleva a cabo y se muestra un mensaje de éxito.
Modificación del rol de usuario a usuario no administrador.	La modificación se lleva a cabo y se muestra un mensaje de éxito.
Eliminación definitiva del usuario.	La eliminación se lleva a cabo y se muestra un mensaje de éxito. Tras ello el navegador redirige al administrador a la vista de usuarios (listado de usuarios registrados).

Tabla 14: Pruebas para la vista de detalle de usuario de administrador.

D.3.2. Vista de Creación de Modelo de Estación Meteorológica

A continuación se muestra una tabla donde se pueden apreciar las pruebas desarrolladas para esta vista así como el resultado previsto para las mismas:

Prueba	Resultado previsto
Creación de modelo de estación meteorológica dejando campos en blanco.	No se crea el modelo y se muestra un mensaje de error explicando que todos los campos deben ser rellenados.
Creación de modelo de estación meteorológica válida.	Se crea el nuevo modelo y el navegador muestra la vista de detalle del nuevo modelo de estación meteorológica.

Tabla 15: Pruebas para la vista de creación de modelo de estación meteorológica.

D.3.3. Vista de Detalle de Modelo de Estación Meteorológica

A continuación se muestra una tabla donde se pueden apreciar las pruebas desarrolladas para esta vista así como el resultado previsto para las mismas:

Prueba	Resultado previsto
Modificación del modelo de estación meteorológica dejando campos en blanco.	No se lleva a cabo la modificación y se muestra un mensaje de error explicando que todos los campos deben ser rellenados.
Modificación del modelo de estación meteorológica válida.	La modificación se lleva a cabo y se muestra un mensaje de éxito.

Tabla 16: Pruebas para la vista de detalle de modelo de estación meteorológica.

D.3.4. Vista de Detalle de Publicación de Inf. Meteorológica de Administrador

Como ya se ha comentado en el Anexo C, esta vista permite modificar los datos de la publicación de información meteorológica indicada. Puesto que esta vista comparte controlador con la vista de detalle de publicación de información meteorológica de usuario (sobre la cual ya se han propuesto pruebas), las pruebas descritas en este apartado complementan a las propuestas para dicha vista.

A continuación se muestra una tabla donde se pueden apreciar las pruebas desarrolladas para esta vista así como el resultado previsto para las mismas:

Prueba	Resultado previsto
Activación de la publicación de información meteorológica.	La activación se lleva a cabo, se muestra un mensaje de éxito, se oculta el botón de activación y se muestra el botón de desactivación de publicación.
Eliminación definitiva de la publicación de información meteorológica.	La eliminación se lleva a cabo y se muestra un mensaje de éxito. Tras ello el navegador redirige al administrador a la vista de publicaciones (listado de publicaciones).

Tabla 17: Pruebas para la vista de detalle de publicación de información meteorológica de administrador.

D.3.5. Vista de Detalle de Estación Meteorológica de Administrador

Como ya se ha comentado en el Anexo C, esta vista permite modificar los datos de la estación meteorológica indicada. Puesto que esta vista comparte controlador con la vista de detalle de estación meteorológica de usuario (sobre la cual ya se han propuesto pruebas), las pruebas descritas en este apartado complementan a las propuestas para dicha vista.

A continuación se muestra una tabla donde se pueden apreciar las pruebas desarrolladas para esta vista así como el resultado previsto para las mismas:

Prueba	Resultado previsto
Activación de la estación meteorológica.	La activación se lleva a cabo, se muestra un mensaje de éxito, se oculta el botón de activación y se muestra el botón de desactivación de estación.
Eliminación definitiva de la estación meteorológica.	La eliminación se lleva a cabo y se muestra un mensaje de éxito. Tras ello el navegador redirige al administrador a la vista de estaciones (listado de estaciones).

Tabla 18: Pruebas para la vista de detalle de estación meteorológica de administrador.

Anexo E - Códigos de Error en la Aplicación Web

Con el objetivo de identificar errores y proceder a resolverlos, se han numerado los posibles errores que pueden darse en la aplicación web. Así, si eventualmente se da un error, el usuario podrá identificar que está ocurriendo.

E.1. Estructura de los Códigos de Error

Los códigos de error tienen la siguiente estructura:

Versión	-	Vista	Tipo	Disparador	Causa

Tabla 19: Estructura de los códigos de error.

A continuación se describe cada componente de los códigos de error:

- Versión: versión del código de error. Este campo ha sido añadido con el objetivo de poder extender la estructura de códigos de error si fuera necesario en el futuro. Actualmente, la versión utilizada es la versión 01.
- Vista: número de vista donde se da el error. En la versión 1 consiste en un número de dos dígitos.
- Tipo: tipo del error acontecido. En la versión 1 consiste en un número de dos dígitos.
- Disparador: acción que ha disparado el error. En la versión 1 consiste en un número de dos dígitos.
- Causa: causa del error. En la versión 1 consiste en un número de dos dígitos.

Así, un posible código de error podría ser el siguiente: 01-13033202.

A continuación se muestran los posibles valores para cada componente de los códigos de error:

E.2. Versión

Actualmente, solo existe una posible versión, identificada por dos dígitos:

01	Versión 1

Tabla 20: Posibles valores para el componente versión.

E.3. Vista

Las vistas están identificadas por un número de dos dígitos. A continuación se muestra una tabla donde aparecen las vistas con su correspondiente número de vista:

01	Vista de Entrar/*Log In*
02	Vista de Registro
03	Vista de Estaciones Meteorológicas de Usuario
04	Vista de Creación de Estación Meteorológica de Usuario/Vista de Creación de Estación Meteorológica de Administrador
05	Vista de Detalle de Estación Meteorológica de Usuario/Vista de Detalle de Estación Meteorológica de Administrador
06	Vista de Publicaciones de Información Meteorológica de Usuario

07	Vista de Creación de Publicación de Información Meteorológica de Usuario/Vista de Creación de Publicación de Información Meteorológica de Administrador
08	Vista de Detalle de Publicación de Información Meteorológica de Usuario/Vista de Detalle de Publicación de Inf. Meteorológica de Administrador
09	Vista de Detalle de Usuario/Vista de Detalle de Usuario de Administrador
10	Panel de Administración
11	Vista de Usuarios
12	Vista de *Parsers*
13	Vista de Creación de *Parser*
14	Vista de Detalle de *Parser*
15	Vista de Creación de Dato Requerido por *Parser*
16	Vista de Modelos de Estación Meteorológica
17	Vista de Creación de Modelo de Estación Meteorológica
18	Vista de Detalle de Modelo de Estación Meteorológica
19	Vista de Estaciones Meteorológicas de Administrador
20	Vista de Creación de Usuario de Administrador
21	Vista de Publicaciones de Información Meteorológica de Administrador
22	Vista de Accesos al API
23	Vista de Incidencias

Tabla 21: Posibles valores para el componente vista.

Como se ha comentado en el Anexo C, hay vistas que comparten controlador. Es por esta razón que algunas vistas de la tabla anterior comparten su número de vista.

E.4. Tipos

Existen 3 tipos de errores, todos ellos identificados por un número de dos dígitos:

01	Error de API
02	Error de navegador
03	Error de usuario

Tabla 22: Posibles valores para el componente tipo.

E.5. Disparadores

A continuación se muestran los posibles disparadores de error, identificados también por un número de dos dígitos:

01	Creación de sesión
02	Creación de usuario
03	Obtención de la sesión actual
04	Obtención del listado de estaciones meteorológicas del usuario indicado
05	Obtención del listado de modelos de estación meteorológica
06	Obtención del modelo de estación meteorológica indicado
07	Obtención del dato requerido por *parser* indicado
08	Creación de estación meteorológica
09	Obtención de la estación meteorológica indicada
10	Búsqueda del modelo de estación meteorológica indicado en el listado de modelos
11	Modificación de la estación meteorológica indicada
12	Desactivación de la estación meteorológica indicada

13	Activación de la estación meteorológica indicada
14	Eliminación de la estación meteorológica indicada
15	Obtención del listado de publicaciones de la estación meteorológica indicada
16	Obtención del listado variables meteorológicas reconocidas
17	Obtención de la publicación indicada
18	Obtención del usuario indicado
19	Obtención del listado de estaciones meteorológicas
20	Obtención del listado de variables meteorológicas disponibles para la estación meteorológica indicada
21	Creación de publicación
22	Modificación de la publicación indicada
23	Desactivación de la publicación indicada
24	Activación de la publicación indicada
25	Eliminación de la publicación indicada
26	Modificación del usuario indicado
27	Desactivación del usuario indicado
28	Activación del usuario indicado
29	Eliminación del usuario indicado
30	Obtención del listado de usuarios
31	Obtención del listado de *parsers*
32	Creación de *parser*
33	Obtención de datos requeridos por *parser* del *parser* indicado
34	Obtención del *parser* indicado
35	Modificación del *parser* indicado
36	Eliminación del dato requerido por *parser* indicado
37	Creación de dato requerido por *parser*
38	Creación de modelo de estación meteorológica
39	Modificación del modelo de estación meteorológica indicado
40	Obtención del listado de publicaciones
41	Obtención del listado de incidencias filtrado
42	Obtención del listado de accesos al API filtrado

Tabla 23: Posibles valores para el componente disparador.

E.6. Causas

Las causas de los errores también están definidas por un número de dos dígitos. A continuación se muestran las posibles causas de error y sus números identificativos:

01	Desconocida
02	No se han rellenado todos los campos requeridos del formulario
03	Credenciales inválidos
04	Correo electrónico ya en uso
05	Error leyendo el fichero
06	Dato requerido por *parser* inválido

Tabla 24: Posibles valores para el componente causas.

Anexo F - Puesta en Producción

Como ya se ha comentado anteriormente, una de las fases del proyecto fue la de puesta en producción de la plataforma, de forma que esta quedase accesible en Internet a través del nombre DNS ownmeteo.com. En este Anexo se describe el proceso llevado a cabo para poner en producción la plataforma *ownmeteo.com*.

En primera instancia se habilitó una máquina con sistema operativo Ubuntu Server 14.04. A continuación se configuraron los ficheros de zona DNS del dominio ownmeteo.com para apuntar a la dirección IP de la máquina habilitada. También se obtuvo un certificado SSL para poder realizar las comunicaciones a través de HTTPS. A continuación, se siguieron los siguientes pasos:

1. Instalación de Node.js 6:

Para instalar Node.js 6 en el servidor se ejecutaron los siguientes comandos:

```
$> curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash -
$> sudo apt-get install -y nodejs
```

2. Instalación de MongoDB Community Edition 3:

Para instalar MongoDB en el servidor se ejecutaron los siguientes comandos:

```
$> sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv \
0C49F3730359A14518585931BC711F9BA15703C6
$> echo "deb [ arch=amd64 ] http://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.4 \
multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.4.list
$> sudo apt-get update
$> sudo apt-get install -y mongodb-org
```

3. Instalación de NGINX:

Para instalar NGINX se ejecutaron los siguientes comandos:

```
$> sudo apt-get update
$> sudo apt-get install -y nginx
```

4. Desempaquetado del proyecto:

Una vez instalados los principales componentes se procedió a desempaquetar el código de la plataforma *ownmeteo.com*. Para ello se ejecutaron los siguientes comandos:

```
$> tar -zxvf TFG.tar.gz      # TFG.tar.gz contiene el código de ownmeteo.com
```

5. Copiado del servidor del API REST y del servidor de vistas a /srv:

El código de ambos servidores fue copiado a /srv con los siguientes comandos:

```
$> sudo mkdir /srv/ownmeteo /srv/ownmeteo/api /srv/ownmeteo/view
$> sudo cp -r TFG/API_v1/* /srv/ownmeteo/api
$> sudo cp -r TFG/VIEW_v1/* /srv/ownmeteo/view
```

6. Preparación de la plataforma:

Una vez copiado el código de los servidores se debían inicializar ciertos parámetros de la plataforma (por ejemplo: la creación de un usuario administrador). Para ello, en el servidor del API REST hay un fragmento de código comentado, el cual debe ser descomentado únicamente en la primera ejecución de la plataforma, de forma que se inicialicen los parámetros pertinentes.

Fichero /srv/ownmeteo/api/app.js antes de descomentar:

```
...
/* PRUEBAS */
//console.log("[DEBUG] Ejecutando código pruebas");
//app.use("/", express.static(__dirname + "/tmp"));
//require("./modules/deploy")(models, crypto);
/* FIN PRUEBAS */
...
```

Fichero /srv/ownmeteo/api/app.js después de descomentar:

```
...
/* PRUEBAS */
console.log("[DEBUG] Ejecutando código pruebas");
app.use("/", express.static(__dirname + "/tmp"));
require("./modules/deploy")(models, crypto);
/* FIN PRUEBAS */
...
```

Tras descomentar el fragmento de código anterior se ejecutó el API REST con el siguiente comando:

```
$> sudo node /srv/ownmeteo/api/app.js
```

Una vez se habían creado el usuario administrador, las unidades de medida y las variables meteorológicas (esta información aparece por consola), se detuvo el servidor mediante Ctrl+C. Finalmente se volvió a comentar el fragmento de código descomentado previamente.

7. Configuración de NGINX:

En primera instancia se configuró NGINX para:

- Redirigir las peticiones a su correspondiente servidor (al de vistas o al del API REST) en función de la URI solicitada.
- Utilizar el certificado SSL obtenido anteriormente.
- Redirigir las peticiones inseguras HTTP al puerto seguro HTTPS 443.
- Redirigir las peticiones a www.ownmeteo.com a ownmeteo.com.

Para ello se modificó el fichero /etc/nginx/sites-available/default de forma que tuviera el siguiente contenido:

```
server {

        listen 443 ssl;
        server_name ownmeteo.com;
        ssl_certificate /etc/letsencrypt/live/ownmeteo.com/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/ownmeteo.com/privkey.pem;

        location / {
                proxy_pass http://127.0.0.1:8081;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection 'upgrade';
                proxy_set_header Host $host;
                proxy_cache_bypass $http_upgrade;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        }

        location /api {
                proxy_pass http://127.0.0.1:8080;
                proxy_http_version 1.1;
```

```
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection 'upgrade';
                proxy_set_header Host $host;
                proxy_cache_bypass $http_upgrade;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        }
}

server {
        listen 443 ssl;
        server_name www.ownmeteo.com;
        return 301 https://ownmeteo.com$request_uri;
}

server {
        listen 80;
        server_name ownmeteo.com www.ownmeteo.com;
        return 301 https://ownmeteo.com$request_uri;
}
```

Tras ello se aumentó el tamaño máximo de petición aceptado por NGINX a 50 MB, de forma que los nuevos *parsers* puedan ser subidos sin problemas. Para ello se añadió la siguiente línea en la cláusula `http` del fichero /etc/nginx/nginx.conf:

```
...
http {
        ...
        client_max_body_size 50M;
        ...
}
...
```

8. Creación de los ficheros de servicios para los servidores Node.js:

La siguiente tarea realizada fue la de hacer que los servidores Node.js (servidor del API REST y servidor de vistas) fueran ejecutados como servicios por el sistema operativo. Así, si estos servidores se cierran repentinamente a causa de un error o el sistema se reinicia, el sistema operativo se encargará de volverlos a poner en funcionamiento.

Para ello se crearon los ficheros /etc/init/ownmeteo-api.conf y /etc/init/ownmeteo-view.conf.

Contenido final del fichero /etc/init/ownmeteo-api.conf:

```
description "ownmeteo.com - API REST"
author      "David Enjuanes"

start on runlevel [2345]
stop on shutdown

respawn
respawn limit 10 5

script
    export HOME="/srv/ownmeteo/api"
    echo $$ > /var/run/ownmeteo-api.pid
    exec /usr/bin/nodejs /srv/ownmeteo/api/app.js
end script

pre-start script
    echo "[`date`] ownmeteo.com API REST Starting" >> /var/log/ownmeteo-api.log
end script

pre-stop script
    rm /var/run/ownmeteo-api.pid
    echo "[`date`] ownmeteo.com API REST Stopping" >> /var/log/ownmeteo-api.log
end script
```

Contenido final del fichero /etc/init/ownmeteo-view.conf:

```
description "ownmeteo.com - Vistas"
author      "David Enjuanes"

start on runlevel [2345]
stop on shutdown

respawn
respawn limit 10 5

script
    export HOME="/srv/ownmeteo/view"
    echo $$ > /var/run/ownmeteo-view.pid
    exec /usr/bin/nodejs /srv/ownmeteo/view/app.js
end script

pre-start script
    echo "[`date`] ownmeteo.com Views Server Starting" >> /var/log/ownmeteo-view.log
end script

pre-stop script
    rm /var/run/ownmeteo-view.pid
    echo "[`date`] ownmeteo.com Views Server Stopping" >> /var/log/ownmeteo-view.log
end script
```

9. Habilitación del *firewall*:

El servidor del API REST y el servidor de vistas escuchan las peticiones HTTP en los puertos TCP 8080 y 8081 respectivamente. Sin embargo, estos puertos solo deben ser accedidos por NGINX, puesto que los clientes no deben acceder a los servidores Node.js directamente, sino que lo deben hacer a través de NGINX.

Así, por seguridad y para evitar lo descrito en el párrafo anterior, se habilitó el *firewall* del sistema de forma que solo quedasen abiertos los siguientes puertos: 22 (SSH), 80 (HTTP) y 443 (HTTPS).

Para habilitar el *firewall* con las reglas anteriores se ejecutaron los siguientes comandos:

```
sudo ufw disable
sudo ufw default deny incoming
sudo ufw default allow outgoing
sudo ufw allow ssh
sudo ufw allow http
sudo ufw allow https
sudo ufw enable
```

10. Habilitación del Lector de Estaciones Meteorológicas:

A través de la herramienta del sistema crontab se configuró que el Lector de Estaciones Meteorológicas se ejecutase periódicamente, de forma que la información meteorológica de las estaciones de la plataforma se registrase cada cierto tiempo.

Para ello, mediante la ejecución del comando `crontab -e`, se añadió la siguiente línea de código al fichero crontab (se asume que el proyecto ha sido desempaquetado -paso 4- en el directorio /home/david):

```
*/5 * * * * /usr/bin/nodejs /home/david/TFG/stationReader/app.js
```

Esta configuración hace que el Lector de Estaciones Meteorológicas sea ejecutado cada 5 minutos.

11. Reinicio del servidor:

Para que todos los cambios anteriores surtieran efecto se reinició el servidor con el siguiente comando:

```
$> sudo reboot
```

12. Comprobación del correcto funcionamiento:

Finalmente se comprobó el correcto funcionamiento de la plataforma accediendo a la misma a través de la URL https://www.ownmeteo.com.