Departamento de Ingeniería y Ciencia de los Computadores

Máster Universitario en Sistemas Inteligentes

Desarrollo de una API para la descripción y gestión de Servicios Web REST

SIU043 - Trabajo fin de máster

Autor: José Manuel Puerta González

Dirigido por: Ismael Sanz Blasco

Curso 2014/2015

Resumen

En esta memoria se presenta el trabajo de fin de máster que consiste en el desarrollo de una API para la gestión de descripciones de servicios y otra para la monitorización y gestión de servicios. Además, también se ha implementado una interfaz web para utilizar la API de monitorización y gestión de forma amigable.

En la memoria se describen los pasos realizados para llevar a cabo el proyecto. En primer lugar se comenta la motivación, se establecen los objetivos, se muestra la planificación y se describen las tecnologías utilizadas. A continuación se realiza un estado del arte sobre la descripción y gestión de servicios web REST. Después, se definen los requisitos que deben cumplir las API y la interfaz y se describe su especificación y diseño. Seguidamente, se explican los detalles de implementación y las pruebas realizadas. Finalmente, se presentan las conclusiones y el trabajo futuro.

Índice general

Índice de tablas

Índice de figuras

1 Introducción

En este primer capítulo se describe la motivación del proyecto y sus principales objetivos. A continuación, se presenta la planificación temporal, las tecnologías utilizadas y la estructura del resto de la memoria.

1.1 Motivación

Actualmente el número de servicios web ha crecido enormemente, lo que implica la necesidad de usar mecanismos que permitan gestionarlos y monitorizarlos, ya que resulta difícil saber qué servicios hay implementados y en qué estado se encuentran. Para poder monitorizar y gestionar los servicios hace falta describirlos para saber qué hacen y cómo invocarlos. Sin embargo, las tareas de gestión y monitorización son complejas y no se pueden realizar manualmente, por lo que se necesita alguna herramienta que las facilite.

Además, el uso de servicios web basados en el estilo arquitectónico REST [11] está aumentando porque se trata de una arquitectura más simple y ligera de utilizar que las que se han venido utilizando hasta ahora.

Una organización que presta servicios a través de la web necesita organizarlos de forma que sean fácilmente localizables y accesibles. Para ello es necesario tener un repositorio de servicios. Su mantenimiento requiere de una herramienta de monitorización capaz de controlar el estado de los servicios y obtener información relevante para la toma de decisiones.

La motivación de este proyecto surge de la necesidad de desarrollar una herramienta que permita describir los servicios de manera que se puedan gestionar y monitorizar. Además, la tendencia de uso de servicios REST está creciendo, por lo que la herramienta que se quiere desarrollar se basa en esta tecnología.

1.2 Objetivos

Los objetivos de este trabajo final de máster son:

- Diseñar e implementar una API REST para la gestión de descripciones de servicios.
- Diseñar e implementar una API REST para la monitorización y gestión de servicios. La monitorización de los servicios permitirá:
    - Conocer el estado en el que se encuentra un servicio.
    - Medir el tiempo que ha permanecido activo o inactivo.
    - Medir el tiempo de respuesta del servicio.
    - Enviar alertas a los usuarios.

  La gestión de los servicios permitirá ejecutar y parar aquellos servicios que sean locales.

- Diseñar e implementar una interfaz web para utilizar de forma amigable la API de monitorización y gestión.

Estos objetivos se explicarán en detalle en el Análisis (capítulo 3).

1.3 Planificación

El trabajo final de máster es una asignatura de 12 créditos lo que implica una dedicación de 300 horas. En este apartado se muestra una tabla con la planificación temporal de las tareas que se han de llevar a cabo para conseguir los objetivos del proyecto.

Tarea	**Horas planificadas**
Documentación sobre tecnologías empleadas	20
Análisis de la API para la descripción de servicios	20
Diseño de la API para la descripción de servicios	20
Implementación de la API para la descripción de servicios	50
Análisis de la monitorización y la gestión de servicios	20
Diseño de la monitorización y la gestión de servicios	10
Implementación de la API para la monitorización y la gestión de servicios	60
Implementación de una interfaz para la monitorización y la gestión de servicios	35
Pruebas	20
Redacción de la memoria	35
Preparación de la presentación	10

Tabla 1: Planificación temporal del proyecto

Esta planificación ha sufrido algunas desviaciones ya que se han tenido que dedicar más horas a la implementación debido a que se han utilizado librerías que no se conocían y había que aprender a usarlas.

1.4 Tecnologías empleadas

En este apartado se describen las herramientas y tecnologías que se han usado para desarrollar el proyecto.

- **Python:** Python [1] es un lenguaje de scripting independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad.

Se ha elegido este lenguaje por su simplicidad y por la enorme disponibilidad y variedad de librerías que facilitan el desarrollo de aplicaciones. Además cuenta con una amplia documentación y una gran comunidad de desarrolladores que resuelven multitud de problemas en los foros.

- **JSON:** Acrónimo de JavaScript Object Notation [2], es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.JSON [3] es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

- **HTTP:** Hypertext Transfer Protocol [4] (en español protocolo de transferencia de hipertexto) es el protocolo usado en cada transacción de la World Wide Web. HTTP define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor.

- **Bootstrap:** Bootstrap[5]es un framework o conjunto de herramientas de software libre para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como, extensiones de JavaScript opcionales adicionales.

- **Javascript:** JavaScript [6] (abreviado comúnmente "JS") es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

  Se utiliza principalmente del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas.

- **Ajax:** Acrónimo de Asynchronous JavaScript And XML [7] (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones. El acceso a los datos se realiza mediante XMLHttpRequest, objeto disponible en los navegadores actuales.

- **Bottle:** Bottle [8] es un rápido, simple y ligero micro web-framework WSGI (Web Server Gateway Interface) para Python. WSGI [9] es una interface simple y universal entre los servidores web y las aplicaciones web o frameworks.

1.5 Estructura

El resto de esta memoria se organiza en los siguientes capítulos:

En el capítulo 2 se presenta el estado del arte de las descripción y gestión de servicios. También se habla sobre los servicios web, sobre SOAP y el estilo arquitectónico REST.

En el capítulo 3 se analizan los requisitos que tiene que cumplir la API de gestión de descripciones de servicios, la API de monitorización y gestión de servicios y la interfaz web.

El capítulo 4 consiste en el diseño de las API y la interfaz. Por una parte, se explica la especificación de cada una de las API y por la otra el diseño de la interfaz.

El capítulo 5 se detallan las pruebas realizadas para comprobar el correcto funcionamiento de las API y de la interfaz.

En el capítulo 6 se describen los detalles de implementación.

En el capítulo 7 se presentan las conclusiones y el trabajo futuro.

Por último, en los anexos se explican los ficheros de configuración de los servicios.

2 Estado del arte

En este capítulo se presenta el estado del arte de la descripción y la gestión de servicios web REST.

En primer lugar se definen los servicios web y el protocolo *SOAP* (Simple Object Access Protocol), a continuación se explica la arquitectura REST y finalmente se comentan los problemas que surgen en la descripción y la gestión de los servicios web.

2.1 Los Servicios Web

Existen múltiples definiciones para el término Servicios Web [10], lo que muestra su complejidad a la hora de dar una adecuada definición que englobe todo lo que son e implican. Una posible sería hablar de ellos como un conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer unos servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web.

2.2 SOAP

Según la Wikipedia [13], SOAP (siglas de Simple Object Access Protocol) es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse a través del intercambio de datos XML. Este protocolo deriva de un protocolo creado por Dave Winer en 1998, llamado XML-RPC. SOAP fue creado por Microsoft, IBM y otros. Está actualmente bajo el auspicio de la W3C (World Wide Web Consortium, abreviado W3C, que es un consorcio internacional que produce recomendaciones para la World Wide Web). Es uno de los protocolos utilizados en los servicios Web.

Según [14] muchos diseñadores de servicios web están llegando a la conclusión de que SOAP es demasiado complicado. Por tanto, están comenzando a utilizar servicios web basados en REST para mostrar grandes cantidades de datos. Este es el caso de grandes empresas como eBay y Google.

El estilo arquitectónico REST es más ligero, simple y es capaz de aprovechar mucho mejor las tecnologías de la web que le han hecho ser escalable hasta el tamaño de internet.

En el siguiente apartado se explica en qué consiste el estilo arquitectónico REST.

2.3 Arquitectura REST

Esta sección tiene dos apartados. En primer lugar se define la arquitectura REST (Representational State Transfer) y a continuación se describen sus restricciones.

2.3.1 Definición

Según la Wikipedia [11], REST es un modelo de arquitectura de software que consta de un conjunto coordinado de restricciones arquitectónicas aplicadas a los componentes, conectores y datos, dentro de un sistema hipermedia distribuido. REST ignora los detalles de la implementación de los componentes y la sintaxis de protocolo con el fin de centrarse en las funciones de los componentes, las restricciones sobre su interacción con otros componentes, y la interpretación de los datos.

2.3.2 Restricciones

Según [12] las características arquitectónicas de REST se pueden resumir de la siguiente manera: (1) la interacción sin estado, (2) interfaz uniforme, (3) la identificación de los recursos, (4) manipulación de los recursos a través de representaciones, (5) los mensajes autodescriptivos, y (6) hipermedia como el motor del estado de la aplicación.

La interacción **sin estado** significa que todo el estado de sesión se mantiene en el cliente y que cada solicitud desde el cliente al servidor debe contener toda la información necesaria para que el servidor pueda comprender la solicitud; esto hace que cada interacción con el servidor sea independiente y por lo tanto desacopla el cliente del servidor.

Todas las interacciones en un sistema REST se llevan a cabo a través de una **interfaz uniforme** que desacopla las implementaciones de los servicios que se prestan. Para obtener este tipo de interfaz uniforme, todos los recursos deben ser accesibles a través de una representación y tienen que tener un **identificador**. Todas las representaciones de recursos deben ser autodescriptivas, es decir, que de alguna manera están etiquetados con un tipo que especifica la forma en que se han de interpretar.

Finalmente, la restricción **hipermedia como el motor del estado de la aplicación** (HATEOAS, del inglés Hypermedia As The Engine Of Application State) se refiere a la utilización de hipervínculos en las representaciones de los recursos como una forma de navegar por el estado de la aplicación. Por ejemplo, cuando se usan hipervínculos en un hipertexto.

La arquitectura REST se basa en estándares de la Web. Esto significa que debe utilizar identificadores de recursos uniformes (URI) para la identificación de los recursos, el Protocolo de transferencia de hipertexto (HTTP) para acceder y modificar las representaciones de los recursos, y el Marco de Descripción de Recursos (RDF) como el modelo de datos unificado para la descripción de recursos.

La clave del éxito de REST es el aprovechamiento de los estándares de la Web y su simplicidad a la hora de desarrollar servicios web. Por ello, la tendencia en el uso de REST crece a medida que baja la de SOAP.

Una descripción más detallada se encuentra en la tesis doctoral de Roy Fielding [15] que fue quien introdujo el término REST en el año 2000. En su capítulo cinco especifica con detalle los elementos de la arquitectura: componentes, conectores y datos. Los objetivos que se persiguen: escalabilidad, generalidad de las interfaces, independencia en el desarrollo de

componentes y sistemas intermedios para reducir el tiempo de interacción, mejorar la seguridad y encapsular los sistemas heredados.

El cumplimiento de estas restricciones permite a cualquier tipo de sistema de hipermedios distribuidos tener propiedades deseables como rendimiento, escalabilidad, sencillez, modificabilidad, visibilidad, portabilidad y fiabilidad.

Para aprovechar todas las características de los servicios web basados en la arquitectura REST es necesario realizar descripciones de los servicios y aplicar anotaciones semánticas (añadir metadatos y datos ontológicos), que permitan automatizar la búsqueda, el descubrimiento y la composición de los servicios.

2.4 Descripción de Servicios

Siguiendo a María Jesús Lamarca Lapuente [16] la World Wide Web no es sólo un espacio de información, también es un espacio de interacción. Utilizando la Web como plataforma, los usuarios, de forma remota, pueden solicitar un servicio que algún proveedor ofrezca en la red. Pero, para que esta interacción funcione, deben existir unos mecanismos de comunicación estándar entre diferentes aplicaciones. Estos mecanismos deben poder interactuar entre sí para presentar la información de forma dinámica al usuario. Por lo tanto se precisa una arquitectura de referencia estándar que haga posible la interoperabilidad y extensibilidad entre las distintas aplicaciones y que permita su combinación para realizar operaciones complejas.

Según la W3C [17] la descripción de servicios juega un papel importante en el acceso a un servicio. La mayoría de lenguajes de descripción de servicios están diseñados para ser consumidos por máquinas, lo que permite la creación automática de esqueletos de código, búsquedas más fáciles y la clasificación de servicios.

Siguiendo la W3C [18] los servicios web proporcionan los cimientos basados en estándares para el intercambio de información entre sistemas software distribuidos. La recomendación del W3C Web Services Description Language (WSDL) especifica un método estándar para describir las interfaces de un servicio Web a nivel sintáctico y cómo invocarlo. Si bien las descripciones sintácticas proporcionan información sobre la estructura de los mensajes de entrada y salida de una interfaz y sobre cómo invocar el servicio, se necesita una semántica para describir lo que hace realmente un servicio web. Esta semántica, cuando se expresa en lenguajes formales, elimina la ambigüedad de la descripción de las interfaces de los servicios Web, allanando el camino para el descubrimiento automático, la composición y la integración de componentes de software. WSDL no prevé explícitamente mecanismos para especificar la semántica de un servicio Web. La Anotación Semántica para WSDL y XML Schema (SAWSDL) define los mecanismos por los cuales las anotaciones semánticas pueden añadirse a los componentes de WSDL.

Según [12] una documentación legible por máquinas de la interfaz de un servicio y sus datos es un primer paso hacia su (semi) integración automática. Un lenguaje de descripción semántica para servicios REST debe cumplir con las restricciones del estilo arquitectónico REST.

Aunque estas restricciones son importantes en el diseño de un servicio REST, los aspectos más importantes para un lenguaje de descripción semántica son cómo se puede acceder a los recursos, cómo se representan, y cómo están relacionados entre sí. El lenguaje de descripción debe ser lo suficientemente expresivo para describir cómo la representación de los recursos pueden ser recuperados y manipulados, y cuál es el significado de esas representaciones. Para integrar el servicio en la Web Semántica, el lenguaje de descripción también debe proporcionar los medios para transformar las representaciones en triples RDF (Resource Description Framework, son declaraciones sobre recursos web en forma de expresiones sujeto-predicado-objeto).

El principal problema de la descripción semántica de Servicios REST o API Web es que no existe un formato de descripción estándar legible por máquinas.

2.5 Gestión de Servicios

La gestión de servicios Web [19] se realiza a través de un conjunto de capacidades de gestión que permiten la monitorización, el control y la generación de informes sobre la calidad y el uso del servicio. En la calidad del servicio influyen cualidades como la disponibilidad (presencia y número de instancias del servicio) y el rendimiento (por ejemplo, la latencia de acceso y la tasa de fallos), y también la accesibilidad (de los endpoints).

Un servicio web se puede gestionar cuando dispone de un conjunto de operaciones de gestión que permiten manejar las capacidades de gestión. Estas capacidades de gestión realizan sus funciones de monitorización, control y presentación de informes con la ayuda de un modelo de información de gestión que modela los distintos tipos de uso del servicio y la información de la calidad de servicio asociados a la gestión del servicio Web. Los tipos de información típicas incluyen el total de solicitudes y respuestas, los tiempos de inicio y finalización, los estados del ciclo de vida, los identificadores de la entidad (por ejemplo, de los remitentes, receptores, contextos, mensajes, etc.)

Aunque las capacidades de gestión permiten a un servicio web convertirse en gestionable, la extensión y el grado de gestión admisible se definen en las políticas de gestión que están asociadas con el servicio web. Las políticas de gestión, por lo tanto se utilizan para definir las obligaciones y los permisos necesarios para la gestión del servicio Web.

La gestión del servicio web tiene que tener una semántica de servicios común que pueda ser entendida por el solicitante y proveedor.

2.5.1 Definición del problema

Según [20] la Arquitectura Orientada a Servicios (SOA) se está utilizando en gran medida en las organizaciones. Cada vez se demanda más flexibilidad y dinamismo en la integración de procesos. Como consecuencia han surgido nuevos problemas en las organizaciones a la hora de gestionar servicios web.

El primer problema es cómo gestionar y controlar los servicios con la misma eficacia con la que se gestionan y controlan el resto de los recursos de la organización. En la actualidad, la monitorización y la gestión de servicios web son tareas complicadas. Cuando se manejan grandes cantidades de servicios puede pasar que nadie sepa cuántos servicios se han implementado, dónde se encuentran y de qué se ocupan. La organización carece de la información sobre el uso y el estado de los servicios. Además, no existe ninguna forma de visualizar qué servicios se están ejecutando. Por lo tanto, se pueden duplicar servicios. Es importante disponer de herramientas que permitan identificar los servicios web duplicados. Para ello, es necesario obtener y analizar datos estadísticos sobre el uso de los servicios.

El segundo problema es cómo monitorizar y utilizar la Calidad de Servicio (QoS) cuando las empresas necesitan cobrar por el uso de sus servicios. En este caso, los servicios deben seleccionarse en tiempo de ejecución y elegir de entre un conjunto de servicios en base a la QoS. Sólo se pueden usar los servicios web disponibles en dicho momento y en ningún caso llamar a ningún servicio web offline. También se debe descubrir los endpoints de los servicios que muestran un bajo rendimiento y hacer que no estén disponibles si no cumplen con los umbrales configurados.

El tercer problema es cómo informar sobre las alertas y la integridad de los servicios de manera dinámica. Esta parte es crucial para una empresa que desea facturar por los servicios prestados, entonces necesita monitorizar las métricas de integridad de los servicios a nivel de negocio para identificar los incumplimientos de los Acuerdos de Nivel de Servicio (o SLA) y las interrupciones temporales de los servicios. Además, también necesitan encontrar una forma de visualizar estas métricas. Las alertas se deben enviar automáticamente cuando se detectan fallos para así poder tomar todas la medidas oportunas.

El cuarto problema es cómo solucionar los problemas de seguridad. Existen muchas amenazas de seguridad provocadas por el acceso externo. No se pueden ofrecer servicios a clientes, socios y proveedores debido a la falta de seguridad. El uso de servicios resulta engorroso debido a la cantidad de sistemas de autenticación y autorización que son necesarios para dar acceso a los socios. Por lo tanto, el problema de la seguridad de los servicios también resulta muy importante y se debe solucionar.

Como se puede ver, buena parte de los problemas (seguridad, SLA, QoS) surgen cuando una empresa necesita controlar el uso de unos servicios que serán facturados a un cliente. La gestión de servicios para una organización sin ánimo de lucro o de pequeño tamaño es mucho más sencilla.

2.6 Trabajo relacionado

En este apartado vamos a comentar tres proyectos que están relacionados con la descripción y gestión de servicios:

- **ProgrammableWeb**: ProgrammableWeb [21] es una web que permite a los usuarios compartir las API que han desarrollado. Actualmente, su repositorio tiene más de catorce mil registros. Se pueden realizar búsquedas y filtrar los resultados por

categorías, protocolo y el tipo de formato que utiliza la API para el intercambio de datos. En nuestro caso, la API para la descripción de servicios se limita a servicios REST que usan el formato JSON para el intercambio de datos.

- **Galileo**: es una plataforma de análisis para la monitorización, visualización e inspección del tráfico de las API y microservicios. Galileo [22] permite:
    - Ver el registro de las peticiones y respuestas según van llegando.
    - Filtrar datos para que el usuario pueda ver los datos que le interesan y ver cómo se están solicitando sus servicios y cómo están respondiendo los servidores.
    - Ver tendencias en los datos a partir de las gráficas de series temporales y así detectar visualmente cambios y anomalías.
    - Crear cuadros de mando personalizados con métricas y datos.
    - Programar informes.
- **SSWAP**: SSWAP (Simple Semantic Web Architecture and Protocol) [23] contiene un repositorio de servicios web que se pueden combinar en línea utilizando una interfaz gráfica. Este es un ejemplo de cómo aprovechar la descripción de los servicios no solo para crear repositorios y realizar búsquedas sino también para componer servicios.

3 Análisis

En este capítulo se definen los requisitos que tienen que cumplir las API y la interfaz de monitorización y gestión para alcanzar los objetivos especificados en el apartado 1.2. En primer lugar, se describen los requisitos para la gestión de descripciones y para la monitorización y gestión. Por último, se detallan los requisitos de la interfaz de monitorización y gestión.

3.1 Gestión de descripciones de servicios

El servicio para la gestión de descripciones de servicios REST tiene como principal objetivo recoger la información necesaria para describir un servicio. Estas descripciones permiten conocer qué hace el servicio y cómo se debe invocar. Para hacer esta gestión de descripciones el servicio deberá proporcionar una API REST que tendrá que cumplir con los siguientes requisitos:

- La API permitirá obtener un listado de las descripciones de los servicios.
- La API deberá permitir obtener la descripción de un servicio en concreto.
- La API permitirá crear descripciones de servicios.
- La API deberá permitir modificar la descripción de un servicio.
- La API permitirá borrar la descripción de un servicio.

La descripción de los servicios deberá contener la siguiente información:

- Una URI que identifique el recurso al que queremos acceder. A su vez la URI se compone de varias partes:
    - El **protocolo** de comunicación que se debe usar para acceder al recurso.
    - La **dirección del servidor** (host) donde se encuentra el recurso, éste se puede especificar mediante una dirección IP o mediante un nombre de dominio. Opcionalmente se puede especificar el **puerto de conexión**, si no se indica se utilizará el puerto estándar del protocolo. El protocolo http utiliza por defecto el 80.
    - La **ruta del recurso** dentro del servidor (path).
    - Opcionalmente se puede añadir una ***query string,*** que contiene parámetros adicionales que se suman a la URI y son procesados en el servidor por la aplicación. Su uso más común es el de especificar criterios de búsqueda o filtrado o bien añadir parámetros o tokens de control.

  Un ejemplo de URI sería la siguiente: "http://www.miservidor.com/rest/libros?autor=Camilo José Cela". En este caso estaríamos pidiendo todos los libros del autor Camilo José Cela
- El método http que se ha de utilizar para acceder al recurso.

- La especificación de los parámetros que necesita recibir el servicio, de forma que se pueda saber las características de cada parámetro y de qué forma se han de enviar: en el path, como una *query string* o en el cuerpo de la llamada http (payload).
- Un enlace a la documentación del servicio dónde se explique detalladamente el uso de la API.
- Una descripción general de lo que hace el servicio.

Además, el servicio de gestión de descripciones deberá generar ficheros log para registrar las acciones y errores que se producen durante la ejecución del servicio.

3.2 Monitorización y gestión de servicios

Una vez tenemos un repositorio con descripciones de servicios lo que se pretende es poder monitorizarlos y gestionarlos.

El servicio encargado de la monitorización y la gestión de los servicios REST debe ser capaz por una parte, de generar informes de forma automática cada cierto tiempo y analizar dichos informes para decidir cuándo se ha de enviar por correo alertas a los administradores de los servicios. Por otra parte, el servicio ha de proporcionar una API REST que permita realizar la gestión de las descripciones de monitorización, de los informes y de los servicios.

A continuación, se describe por una parte los requisitos de la API para la monitorización y gestión de servicios divididos en tres bloques: la gestión de las descripciones de monitorización, la gestión de los informes y la gestión de los servicios, y por otra parte los requisitos que ha de cumplir el servicio que proporciona la API para automatizar las tareas de monitorización.

Los requisitos que ha de cumplir la API para la monitorización y gestión de servicios son los siguientes:

- Para la gestión de las descripciones de monitorización:
    - La API permitirá obtener todas las descripciones de monitorización almacenadas en el servidor.
    - La API deberá permitir obtener la descripción de monitorización de un servicio en concreto.
    - La API permitirá crear descripciones de monitorización.
    - La API deberá permitir modificar la descripción de monitorización de un servicio.
    - La API permitirá borrar la descripción de monitorización de un servicio.

- Para la gestión de los informes:
    - La API deberá permitir obtener el último informe de monitorización guardado.
    - La API permitirá crear informes sobre el estado de los servicios.
    - La API deberá permitir actualizar el informe de un servicio en concreto.

- Para la gestión de los servicios:

- La API permitirá arrancar un servicio que se encuentre parado.
- La API deberá permitir parar un servicio que esté en funcionamiento.

Los requisitos que deberá cumplir el servicio son:

- El servicio deberá ser capaz de generar el informe de monitorización automáticamente cada cierto tiempo. Este informe deberá contener el estado en el que se encuentra cada servicio así como su tiempo de respuesta y el tiempo que ha permanecido funcionando o parado.
- El servicio deberá enviar correos de alerta a los administradores de los servicios cuando se detecte que algún servicio se ha caído o se ha recuperado.
- El servicio deberá generar ficheros log para registrar las acciones y errores que se producen durante la ejecución del servicio.

3.3 Interfaz para la monitorización y gestión de servicios

Una vez tenemos almacenados los informes de monitorización lo que se pretende es que esa información sea fácilmente accesible junto con el resto de funcionalidades de la API a través de una interfaz amigable.

La interfaz para la monitorización y gestión deberá cumplir con los siguientes requisitos:

- La interfaz permitirá visualizar el contenido de los informes de monitorización.
- La interfaz deberá permitir crear, modificar y borrar las descripciones de monitorización de los servicios.
- La interfaz permitirá ejecutar y parar los servicios que sean locales.
- La interfaz deberá ser capaz de actualizar el informe de monitorización en cualquier momento en el que se solicite.

4 Diseño

Después de ver los requisitos que se tienen que cumplir, en este capítulo se describen las especificaciones de las API así como el diseño de la interfaz de monitorización. Además, como las API se basan en el estilo arquitectónico REST deben cumplir con los principios de diseño de esta arquitectura tal y como se vio en el estado del arte.

4.1 Especificación de la API para la descripción de servicios

En el estado del arte se vieron dos formas de implementar servicios web: SOAP y REST. SOAP tiene la ventaja de ser un estándar y contar con un lenguaje para la descripción de servicios (WSDL), pero en la práctica es más difícil de implementar debido a su gran complejidad. Por ello, seguiremos el estilo arquitectónico REST que como vimos es mucho más ligero y fácil de implementar. Además, REST no obliga a utilizar un lenguaje de descripción específico. Para las descripciones utilizaremos el formato JSON ya que es muy sencillo y su uso está muy extendido.

Para cumplir con los objetivos fijados en el análisis a continuación se detalla la especificación de la API para la descripción de servicios. En esta especificación se explica qué llamadas se pueden realizar, cómo se pueden realizar y la respuesta que devuelven. Además de esto, el formato de la descripción de un servicio debe estar estructurado como un JSON y contener las claves que se especifican en la siguiente tabla.

Estructura de la descripción de un servicio			
Clave	**Tipo**	**Valor de ejemplo**	**Descripción**
id	Int	1	Identificador del servicio. Se genera automáticamente en el servidor
name	String	Zodiac	Nombre del servicio
method	String	GET	Método http que se ha de utilizar sobre el endpoint
description	String	Este servicio toma como parámetro una fecha de nacimiento (día y mes) y devuelve el signo del zodiaco correspondiente	La descripción del servicio
endpoint	String	http://restwebservices.pythonanywhere.com/zodiac/{day}/{month}	Es una plantilla que hay que completar con los parámetros. Una vez completado es el punto final donde se encuentra el recurso
parameters	List	[{'name':'day', 'place': 'Path', 'required': 'Yes', 'valid-options': [], 'default-value': '', 'description': 'day',	Es una lista de parámetros. Cada parámetro consiste en un JSON. Los valores de este JSON se especifican en la tabla 3

		'type': 'Int'}]	

Tabla 2: Estructura de la descripción de los servicios

En la siguiente tabla se especifica la estructura de los parámetros del servicio:

Estructura de un parámetro de una descripción		
Clave	**Valor de ejemplo**	**Descripción**
name	Day	Día de la semana
place	Path	Determina en qué lugar ha de ir el parámetro. Hay tres posibilidades: Path, Payload o Parameters
required	Yes	Indica si el parámetro es obligatorio o no. Hay dos posibilidades: Yes o No
valid-options	[1,2,3, ..., 31]	Indica los únicos valores que son aceptables
default- value	1	Indica el valor por defecto que toma el parámetro
description	Un día del mes	Describe el significado del parámetro
type	Int	Define de que tipo es el parámetro

Tabla 3: Estructura de un parámetro de una descripción

Como se ha mencionado anteriormente, en los siguientes puntos se detallan las llamadas que admite la API de gestión de descripciones de servicios.

4.1.1 Obtener un listado las descripciones almacenadas en el servidor

```
GET restwebservices.pythonanywhere.com/restwebservices
```

Parámetros de la llamada

Esta llamada no tiene ningún parámetro.

Respuestas

- **Código 200:** cuando la petición se ha realizado correctamente se devuelve una lista de JSONs donde cada uno contiene la descripción de un servicio. En las tablas 2 y 3 se especifica la estructura que tiene dicho JSON.

Ejemplo de respuesta

```
[{"endpoint":
"http://restwebservices.pythonanywhere.com/zodiac/{day}/{month}",
 "name": "Zodiac",
 "parameters": [{"name": "day",
            "default-value": "",
            "required": "Yes",
            "valid-options": [],
            "place": "Path",
            "type": "Int",
```

```
            "description": "day"},
             {"name": "month",
             "default-value": "",
             "required": "Yes",
             "place": "Path",
             "valid-options": [],
             "type": "Int",
             "description":"month"}],
 "documentation": "http://restwebservices.pythonanywhere.com/zodiac",
 "method": "GET",
 "id": "29",
"description": "This service takes as parameters the day and month of
birth and returns a json with the sign of the right zodiac"}]
```


4.1.2 Obtener la descripción de un servicio en concreto

```
GET restwebservices.pythonanywhere.com/restwebservices/{id}
```

Parámetros de la llamada

- **id**: es el identificador de la descripción a la cual queremos acceder.

Respuestas

- **Código 200**: cuando se ha encontrado la descripción especificadase devuelve un JSON con su descripción.
  Ejemplo:


  ```
  {"endpoint": "http://restwebservices.pythonanywhere.com/people",
  "name": "People",
  "parameters":[],
  "documentation":
  "http://restwebservices.pythonanywhere.com/people",
  "method": "GET",
  "id": "4",
  "description": "This service returns a list of people with their
  name, age and date of birth."}
  ```


- **Código 400**: cuando no se encuentra ninguna descripción con la id especificada se devuelve un JSON con el siguiente mensaje de error: `{"error": "Description service not found"}`.

4.1.3 Crear una descripción

```
PUT restwebservices.pythonanywhere.com/restwebservices

Content-Type: application/json
```

Parámetros de la llamada

- **JSON**: para crear una nueva descripción se ha de enviar un JSON en el payload con los datos especificados en las tablas 2 y 3.

Ejemplo:

```
{'name': 'Ziptastic',
 'method': 'GET',
'endpoint': 'http://zip.getziptastic.com/v2/ES/{postalcode}',
 'description': 'Ziptasticis a simple API and powerful web service
      that allows people to ask which Country, State, County and City
      are associated with a postal code',
 'parameters': [{'name': 'postalcode',
'place': 'Path',
                 'required': 'Yes',
                 'valid-options': [],
                 'default-value': '',
                 'description': 'Postal code. Example: 12006',
                 'type': 'Int'}],
 'documentation': 'https://www.getziptastic.com/'}
```

Respuestas

- **Código 200**: cuando la descripción se ha creado correctamente se devuelve un JSON con el identificador (id) de la descripción y un mensaje de confirmación: `{"id": "5", status": "The service description was created successfully"}`.
- **Código 400**: si no se ha podido crear la descripciónse devuelve un JSON con el correspondiente mensaje de error.
  Los posibles mensajes de error son los siguientes:
  - `{"error": "A service description already exists with that name"}`
  - `{"error": "The creation of the service description failed"}`
  - `{"error": "The service name is empty"}`
  - `{"error": "The json received is invalid"}`
  - `{"error": "It has not received anything. It is expecting a json"}`

4.1.4 Modificar una descripción

```
PUT restwebservices.pythonanywhere.com/restwebservices/{id}

Content-Type: application/json
```

Parámetros de la llamada

- **id:** es el identificador de la descripción que queremos modificar.
- **JSON:** para modificar una descripción se ha de enviar un JSON en el payload con los nuevos datos. El JSON debe contener todos los datos especificados en las tablas 2 y 3.

Ejemplo:

```
{'name': 'Ziptastic_V2',
 'method': 'GET',
'endpoint': 'http://zip.getziptastic.com/v2/ES/{postalcode}',
 'description': 'Ziptasticis a simple API and powerful web service
      that allows people to ask which Country, State, County and City
      are associated with a postal code',
 'parameters': [{'name': 'postalcode',
```

```
'place': 'Path',
                'required': 'Yes',
                'valid-options': [],
                'default-value': '',
                'description': 'Postal code. Example: 12006',
                'type': 'Int'}],
 'documentation': 'https://www.getziptastic.com/'}
```

Respuestas

- **Código 200**: cuando la descripción se ha modificado correctamente se devuelve un JSON con el siguiente mensaje de confirmación: `{"status": "The service description has been successfully changed"}`.
- **Código 400**: si no se ha podido modificar la descripción se devuelve un JSON con el correspondiente mensaje de error.
  Los posibles mensajes de error son los siguientes:
  - `{"error": "The service description does not exist"`
  - `{"error": "The service name is empty"}`
  - `{"error": "The json received is invalid"}`
  - `{"error": "It has not received anything. It is expecting a json"}`

4.1.5 Borrar una descripción

```
DELETE restwebservices.pythonanywhere.com/restwebservices/{id}
```

Parámetros de la llamada

- **id:** es el identificador de la descripción que queremos borrar.

Respuestas

- **Código 200**: cuando la descripción se ha borrado correctamente se devuelve un JSON con el siguiente mensaje de confirmación: `{"status": The service description has been deleted successfully}`.
- **Código 400**: si el id de la descripción no existe se devuelve un JSON con el siguiente mensaje de error: `{"error": "The service description does not exist"}`.

4.2 Especificación de la API para la monitorización y gestión de servicios

Como se pudo ver en el estado del arte la monitorización de los servicios nos permite obtener información sobre su funcionamiento. En nuestro caso vamos monitorizar el estado de los servicios, el tiempo que permanecen en ejecución o parados y el tiempo de respuesta. Las gestión de los servicios nos va a permitir saber cuántos servicios tenemos, de qué se ocupan,

cuáles están en ejecución y generar alertas cuando se produzcan cambios importantes (por ejemplo la caída de un servicio).

Al igual que en el apartado anterior, para cumplir con los objetivos fijados en el análisis a continuación se detalla la especificación de la API para la monitorización y gestión de servicios. En esta especificación se explica qué llamadas se pueden realizar, cómo se pueden realizar y la respuesta que devuelven. Además de esto, el formato de la descripción de monitorización debe estar estructurado como un JSON y contener las claves que se especifican en la siguiente tabla.

Clave	**Tipo**	**Valor de ejemplo**	**Descripción**
name	String	Zodiac	Nombre del servicio.
email-admin	List	[al054956@uji.es]	Es una lista de correos.
script-path	String	./toyservices/zodiac.py	Es la dirección donde se encuentra el script del servicio. Se usa para ejecutar el servicio.
local	Bool	True	Indica si el servicio es local o no.

Tabla 4: Estructura de la descripción de monitorización

A continuación también se detalla la estructura de los informes de monitorización en la siguiente tabla.

Clave	**Tipo**	**Valor de ejemplo**	**Descripción**
id	String	1	Identificador del servicio. Se genera automáticamente en el servidor
name	String	Zodiac	Nombre del servicio. Es único dentro del sistema
status	Bool	True	Indica si el servicio está funcionando o no
status-code	Int	200	El código http devuelto al realizar una petición de comprobación
status-value	String	OK	Es el significado del status-code
status-date	String	08-11-2015 17:03:05	Es la fecha y hora en la que se actualizó el status del servicio
ping	Float	0,305	Es el tiempo en segundos que ha tardado el servicio en responder
local	Bool	True	Indica si el servicio es local o no
email-admin	List	[pepe@uji.es, juan@uji.es]	Es la lista de direcciones de correo a los que se les enviará una alerta cuando los servicios dejen de funcionar o se hayan recuperado de una caída.
days	Int	1	Indica el número de días que un servicio ha permanecido en el status actual desde el último informe
hours	Int	3	Indica el número de horas que un servicio ha permanecido en el status actual hasta un límite de 23 horas
minutes	Int	34	Indica el número de minutos que un servicio ha permanecido en el status actual hasta un límite de 59 minutos
seconds	Int	15	Indica el número de minutos que un servicio ha permanecido en el status actual hasta un

			límite de 59 segundos
time	String	1d 3h 34m 15s	Es una cadena de texto que muestra los días, horas, minutos y segundos construido a partir de los datos anteriores.
total-seconds	Int	34567	Es el tiempo total en segundos que ha permanecido un servicio en su status actual

Tabla 5: Estructura del informe de monitorización

Como se ha mencionado anteriormente, en los siguientes puntos se detallan las llamadas que admite la API de monitorización y gestión.

4.2.1 Obtener todas las descripciones de monitorización

```
GET localhost:8085/monitoring
```

Parámetros de la llamada

Esta llamada no tiene ningún parámetro

Respuestas

- **Código 200**: cuando la petición se ha realizado correctamente se devuelve una lista de JSONs. Cada JSON contiene la descripción de monitorización de cada servicio. En la tabla 4 se especifica la estructura de la descripción.

Ejemplo de respuesta

```
[{"script-path": "./toyservices/zodiac.py", "email-admin":
["al054956@uji.es"], "local": true, "name": "Zodiac2", "id": "5"}]
```

4.2.2 Obtener la descripción de monitorización de un servicio

```
GET localhost:8085/monitoring/{id}
```

Parámetros de la llamada

- **id**: es el identificador de la descripción de monitorización a la cual queremos acceder.

Respuestas

- **Código 200**: cuando la petición se ha realizado correctamente y se ha encontrado la descripción de monitorización, se devuelve un JSON con la descripción especificada. Ejemplo:

  ```
  {"script-path": "./toyservices/people.py", "email-admin":
  ["al054956@uji.es"], "local": true, "name": "People2", "id":
  "6"}
  ```

- **Código 400**: cuando no se encuentra ninguna descripción monitorización con la id especificada se devuelve un JSON con el siguiente mensaje de error: `{"error": "Monitoring description not found"}`.

4.2.3 Crear una descripción de monitorización

```
PUT localhost:8085/monitoring

Content-Type: application/json
```

Parámetros de la llamada

- **JSON**: para crear una nueva descripción de monitorización se ha de enviar un JSON en el payload con los datos especificados en la tabla 4.

Ejemplo:

```
{"script-path": "./toyservices/people.py", "email-admin":
["al054956@uji.es"], "local": true, "name": "People2"}
```

Respuestas

- **Código 200**: cuando la descripción de monitorización se ha creado correctamente se devuelve un JSON con el identificador (id) de la nueva descripción de monitorización y un mensaje de confirmación: `{"id": "3", status": "The monitoring description was created successfully"}`.
- **Código 400**: si no se ha podido crear la descripción de monitorización se devuelve un JSON con el correspondiente mensaje de error.
  Los posibles mensajes de error son los siguientes:
  - `{"error": "The creation of the monitoring description failed"}`
  - `{"error": "A monitoring description already exists with that name"}`
  - `{"error": "Failed attempt to parse the json to turn it into a dictionary"}`
  - `{"error": "Downloading the list of services repository failed"}`
  - `{"error": "The json received is invalid"}`

4.2.4 Modificar una descripción de monitorización

```
PUT localhost:8085/monitoring/{id}

Content-Type: application/json
```

Parámetros de la llamada

- **id:** es el identificador de la descripción monitorización que queremos modificar.
- **JSON:** para modificar una descripción de monitorización se ha de enviar un JSON en el payload con los nuevos datos. El JSON debe contener todos los datos especificados en la tabla 4. Hay que tener en cuenta que no se permite modificar el nombre de la descripción.

Ejemplo:

```
{"script-path": "./toyservices/people.py", "email-admin":
["admin@uji.es"], "local": true, "name": "People2"}
```

Respuestas

- **Código 200**: cuando la descripción de monitorización se ha modificado correctamente se devuelve un JSON con el siguiente mensaje de confirmación: `{"status": "The monitoring description has been successfully changed"}`.
- **Código 400**: si no se ha podido modificar la descripción se devuelve un JSON con el correspondiente mensaje de error.
  Los posibles mensajes de error son los siguientes:
  - `{"error": "The service description does not exist"`
  - `{"error": "The service name is empty"}`
  - `{"error": "The json received is invalid"}`
  - `{"error": "It has not received anything. It is expecting a json"}`

4.2.5 Borrar una descripción de monitorización

```
DELETE localhost:8085/monitoring/{id}
```

Parámetros de la llamada

- **id:** es el identificador de la descripción de monitorización que queremos borrar.

Respuestas

- **Código 200**: cuando la descripción de monitorización se ha borrado correctamente se devuelve un JSON con el siguiente mensaje de confirmación: `{"status": "The monitoring description has been deleted successfully"}`
- **Código 400**: si el id de la descripción de monitorización no existe se devuelve un JSON con el siguiente mensaje de error: `{"error": "The monitoring description does not exist"}`

4.2.6 Obtener el último informe de monitorización guardado

```
GET localhost:8085/monitoring/report
```

Parámetros de la llamada

Esta llamada no tiene ningún parámetro.

Respuestas

- **Código 200**: cuandose obtiene el informe correctamente se devuelve una lista de JSONs. Cada JSON contiene el informe de un servicio monitorizado, su estructura se corresponde con la descrita en la tabla 5 (estructura del informe de monitorización).
  Ejemplo:

[{"status": false, "status-date": "20-11-2015 03:25:25", "total-seconds": 42, "name": "Date Convert2", "seconds": 42, "local": true, "ping": "none", "days": 0, "hours": 0, "email-admin": ["al054956@uji.es"], "time": "0d 0h 0m 42s", "status-value": "No response", "minutes": 0, "id": "1", "status-code": 0}]

- **Código 400**: si no se ha generado ningún informe de monitorización se devuelve un JSON con el siguiente mensaje de error: `{"error": "No report available"}`.

4.2.7 Crear un informe de todos los servicios monitorizados

`GET localhost:8085/monitoring/report/update`

Parámetros de la llamada

Esta llamada no tiene ningún parámetro.

Respuestas

- **Código 200**: cuando se crea el informe correctamente se devuelve un JSONs con el siguiente mensaje de confirmación: `{"status": "The report was updated successfully"}`.
- **Código 400**: si no se pudo actualizar el informe de monitorización se devuelve un JSON con el siguiente mensaje de error: `{"error": "It failed to update the report"}`.

4.2.8 Actualizar el informe de un servicio en concreto

`GET localhost:8085/monitoring/report/update/{id}`

Parámetros de la llamada

- **id:** es el identificador de la descripción monitorización que queremos actualizar.

Respuestas

- **Código 200**: cuando se actualiza el informe del servicio correctamente se devuelve un JSON con el siguiente mensaje de confirmación: `{"status": "The report was updated successfully"}`.
- **Código 400**: si no se pudo actualizar el informe de monitorización se devuelve un JSON con el siguiente mensaje de error: `{"error": "It failed to update the report"}`.

4.2.9 Obtener una lista de los servicios ejecutados

`GET localhost:8084/rwsmanage`

Parámetros de la llamada

Esta llamada no tiene ningún parámetro.

Respuestas

- **Código 200**: cuando se obtiene la lista de servicios ejecutados correctamente se devuelve una lista de JSON. Cada JSON contiene el nombre, el identificador del servicio y su identificador de proceso.
  Ejemplo:

  ```
  [{"pid": 8496, "id": "6", "name": "People2"}, {"pid": 1372,
  "id": "5", "name": "Zodiac2"}, {"pid": 392, "id": "1", "name":
  "Date Convert2"}]
  ```

- **Código 400**: si no se pudo obtener la lista de servicios ejecutados se devuelve un JSON con el siguiente mensaje de error:`{"error": "Load the list of services executed failed"}`.

4.2.10 Obtener la información de ejecución de un servicio

```
GET localhost:8084/rwsmanage/{id}
```

Parámetros de la llamada

- **id:** es el identificador del servicio del que queremos obtener la información de ejecución.

Respuestas

- **Código 200**: cuando se encuentra la información de ejecución del servicio especificado se devuelve en un JSON.
  Ejemplo:

  ```
  {"pid": 8496, "id": "6", "name": "People2"}
  ```

- **Código 400**: si no se pudo encontrar el servicio especificado se devuelve un JSON con el siguiente mensaje de error: `{"error": "Service not found"}`.

4.2.11 Arrancar un servicio

```
GET localhost:8084/rwsmanage/start/{id}
```

Parámetros de la llamada

- **id:** es el identificador de la descripción que contiene la información del servicio que queremos arrancar.

Respuestas

- **Código 200**: cuando se arranca el servicio.se devuelve un JSON con el siguiente mensaje de confirmación: `{"status": "The service ran successfully"}`.
- **Código 400**: si no se pudo arrancar el servicio se devuelve un JSON con el correspondiente mensaje de error. Los posibles mensajes de error son los siguientes:

- `{"error": "Failed service execution"}`
- `{"error": "The service is already running"}`
- `{"error": "Script path not found"}`
- `{"error": "Service not found"}`

4.2.12 Parar un servicio

```
GET localhost:8084/rwsmanage/stop/{id}
```

Parámetros de la llamada

- **id:** es el identificador de la descripción que contiene la información del servicio que queremos parar.

Respuestas

- **Código 200**: cuando se para el servicio.se devuelve un JSON con el siguiente mensaje de confirmación: `{"status": "The process is killed successfully"}`.
- **Código 400**: si no se pudo parar el servicio se devuelve un JSON con el siguiente mensaje de error: `{"error": "The killing of process failure"}`.

4.3 Diseño de los ficheros log

Todas las API implementadas cuentan con ficheros log que registran las acciones y errores que se pueden producir en la ejecución de los servicios y son muy útiles a la hora de depurar errores.

Los datos que se registran en los ficheros log son los siguientes:

- Fecha y hora del registro.
- Nivel de registro, que puede ser info, debug y error.
- Mensaje que se quiere registrar.
- Para las peticiones de los clientes se añade la IP y el tipo de petición
- La línea y el nombre de la función donde se creó el mensaje.
- La ubicación del fichero donde se encuentra la función que generó el mensaje.

Ejemplo de anotación en los ficheros log:

```
[2015-11-15 18:36:06,822] - INFO - The report was created successfully
- [line: 1212 - function: CreateReport -
C:\Users\manu\workspace\TFM\rwsmonitoring.py]
```

Ejemplo de anotación en los ficheros log cuando un cliente realiza una petición:

```
[2015-11-15 20:04:38,523] - INFO - It has received a request to query
from: [ 127.0.0.1 - ' GET /restwebservices HTTP/1.1 ' ] - [line: 240 -
function: rws - C:\Users\manu\workspace\TFM\restwebservices.py]
```

4.4 Envío de las alertas por correo

En el estado del arte se pudo ver que un aspecto importante en la gestión de los servicios es informar de las alertas a los administradores, por ello la API de monitorización también se encarga de enviar alertas por correo electrónico cuando en un informe se detecta que algún servicio se ha caído y también cuando éste se ha recuperado. Además en el fichero de configuración de la API se puede establecer un tiempo de espera antes de enviar el correo, es decir, si se establece un tiempo de espera de 30 minutos y un servicio ha estado caído 25 minutos y después se recupera el correo no se enviará. Esto evita mandar correos innecesarios por pequeñas caídas de los servidores. En la figura 1 se muestra un ejemplo de alerta enviada por correo.

Los destinatarios de los correos pueden ser de dos tipos:

- **Global**: recibirán las alertas de todos los servicios monitorizados. La lista de correos se cargará del fichero de configuración de la API.
- **Individual**: recibirán solo las alertas de los servicios en los que se hayan registrado

Los usuarios que no quieran recibir ninguna alerta pueden incluir su correo en la lista de exclusión del fichero de configuración.

Los correos contienen los siguientes datos:

- El identificador del servicio.
- El nombre del servicio.
- El estatus en el que se encuentra, es decir, el código http devuelto por el servicio y su significado.
- El tiempo que ha permanecido en dicho estatus.

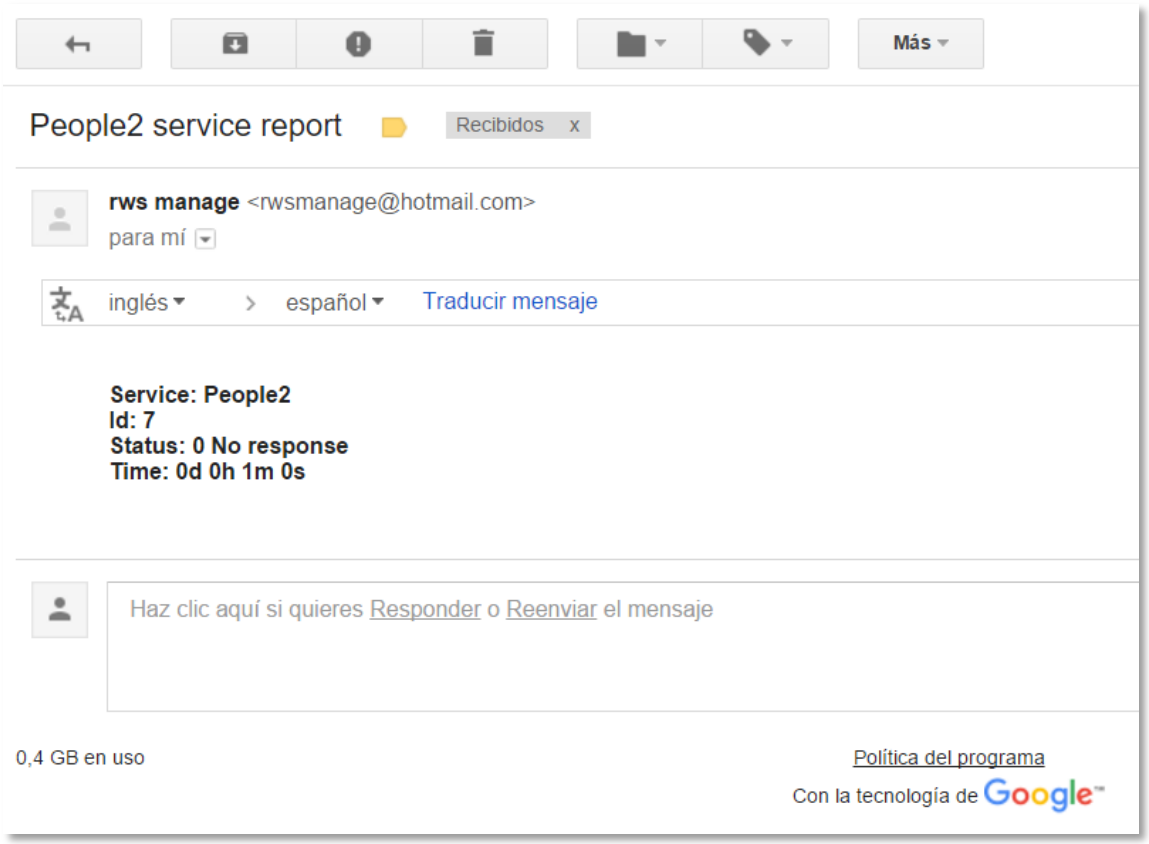


Figura 1: Ejemplo de alerta enviada por correo

4.5 Interfaz para la monitorización y gestión de servicios

La interfaz de monitorización y gestión de servicios sirve para utilizar la API de monitorización y gestión de una manera amigable. Desde esta interfaz se pueden realizar distintas tareas que se explican a continuación.

- **Creación de la descripción de monitorización de un servicio:** para monitorizar un servicio se tiene que crear una nueva descripción de monitorización. Para crear esta descripción se tiene que rellenar un formulario como el que se ve en la figura 2. Los campos que hay que rellenar son los siguientes:

    - **Name:** nombre del servicio que se quiere monitorizar. Este nombre debe ser exactamente igual al nombre del servicio especificado en la descripción del servicio y es obligatorio.
    - **Local:** indica si un servicio es un servicio local o un servicio externo.
    - **Email:** corresponde al email del administrador del servicio. Se pueden indicar varios emails separados por comas.
    - **Script path:** la ruta del fichero que se ha de ejecutar para arrancar el servicio. Este campo solo es necesario cuando se trata de servicios locales.

Cuando se obtienen los datos del formulario se crea un JSON con ellos y se hace una petición HTTP PUT al servidor para almacenarla. La API comprueba que no exista una descripción con el mismo nombre y que ese nombre se corresponda con algún nombre de las descripciones de servicios. Si existe algún error la API lo comunica tal y como se ha visto en su especificación.

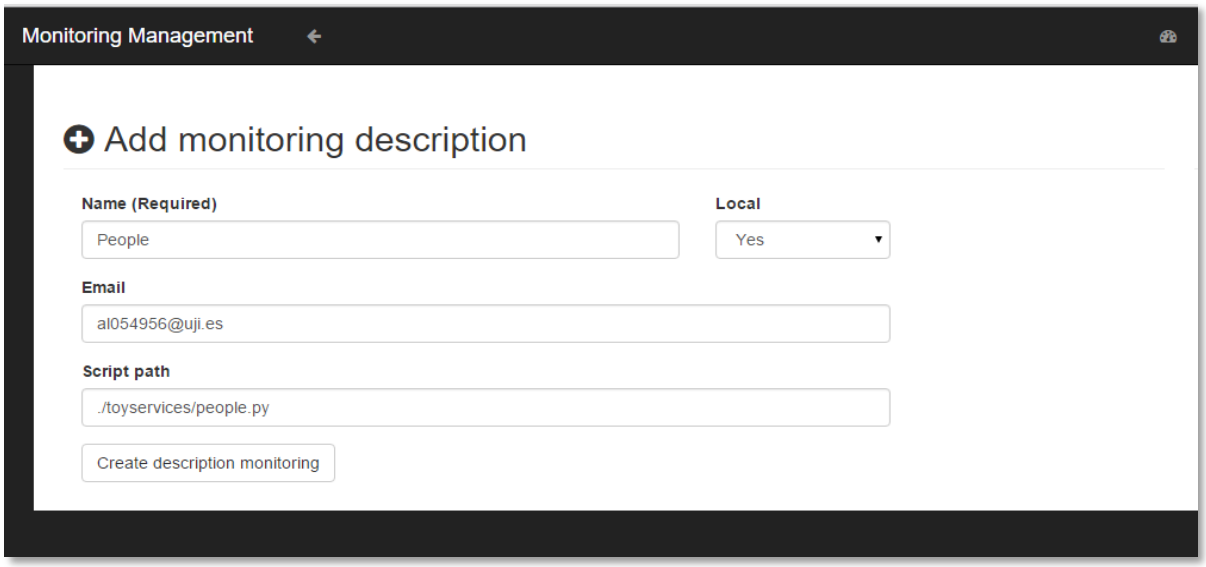


Figura 2: Creación de la descripción de monitorización

- **Modificación de la descripción de monitorización de un servicio:** para modificar la descripción de monitorización de un servicio se tiene que utilizar el formulario que se ve en la figura 3. En este formulario se muestran los valores almacenados de la descripción que se recuperan llamando a la API mediante una petición HTTP GET con el identificador del servicio. Este formulario permite modificar todos los campos de la descripción de monitorización con excepción del nombre ya que es el parámetro que relaciona las descripciones de monitorización con el repositorio de servicios. Después de modificar el formulario se siguen los mismos pasos que cuando se crea una descripción.

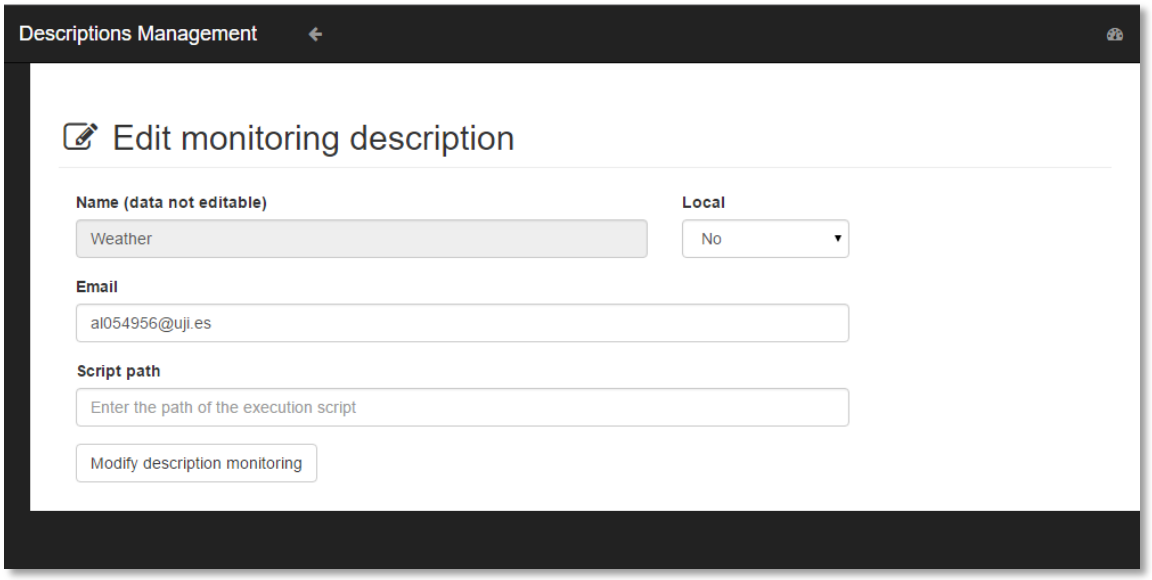


Figura 3: Edición de la descripción de monitorización

- **Visualización de una descripción de monitorización:** para visualizar una descripción se recupera la información igual que se hace en la modificación y se muestra en forma de tabla. En la figura 4 se puede ver un ejemplo de visualización de una descripción.

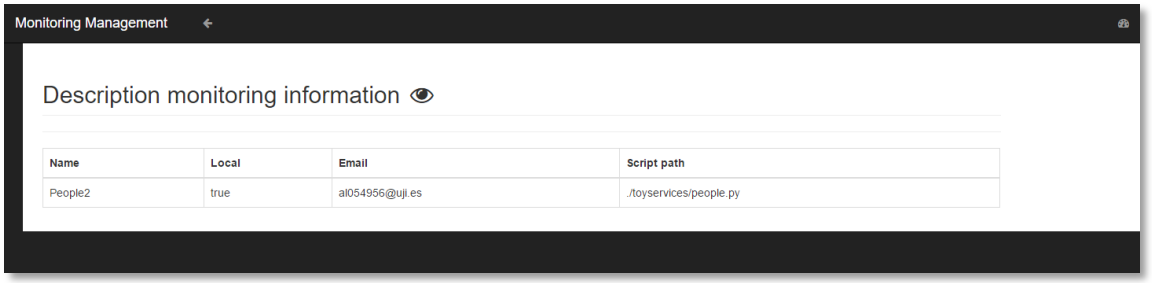


Figura 4: Visualización de la descripción de monitorización

- **Eliminación de una descripción de monitorización:** para eliminar una descripción se hace una solicitud HTTP DELETE a la API de monitorización junto con el identificador de la descripción que se quiere borrar.

- **Visualizar el informe de monitorización de los servicios:** para obtener el listado de todos los servicios que están siendo monitorizados se hace una solicitud HTTP GET a la API de monitorización y ésta nos devuelve una lista de JSON. A partir de esa lista se crea la tabla en la que muestra la información del informe (ver figura 5).

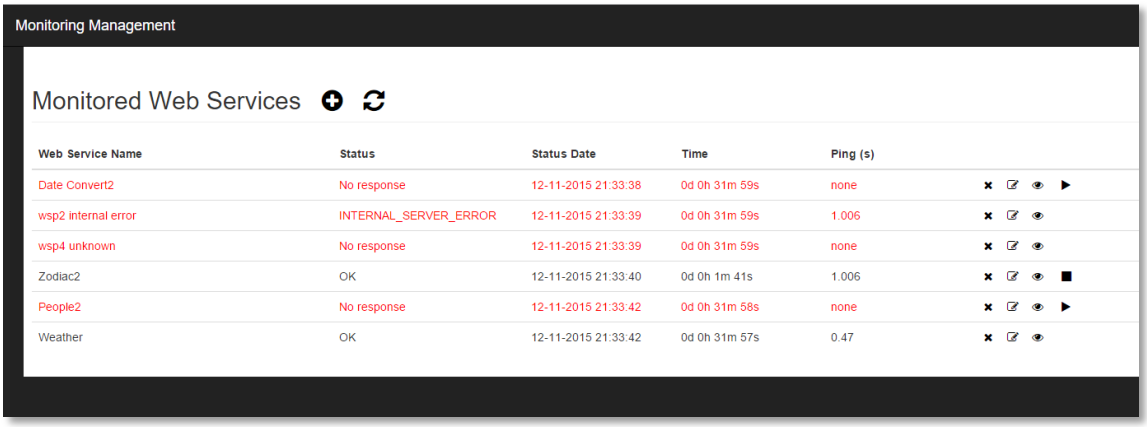


Figura 5: Informe de monitorización

- **Ejecución/Parada de un servicio monitorizado:** cuando se quiere arrancar o parar un servicio que está siendo monitorizado se hace una llamada HTTP GET a la API con la url correspondiente y junto con el identificador del servicio y el servicio se ejecuta o se para. Si se produce algún error la API lo comunica mediante un mensaje.

Todas las tareas que se acaban de describir se pueden realizar en la interfaz usando los botones correspondientes. En la figura 6 se muestra la pantalla principal de la interfaz y en la tabla 6 se indica para qué sirve cada uno de los botones.

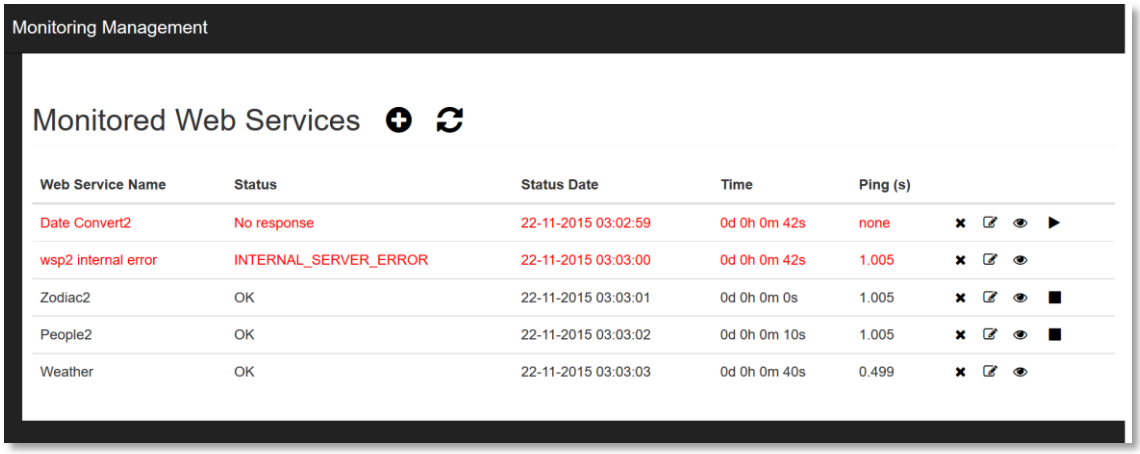


Figura 6: Pantalla principal de la interfaz de monitorización y gestión

Botón	Funcionalidad
	Crear una nueva descripción de monitorización.
	Generar un nuevo informe para todos los servicios monitorizados.
	Elimina la descripción de monitorización.
	Edita la descripción de monitorización.
	Visualiza el contenido de la descripción de monitorización.
	Arranca un servicio local que esté parado.
	Para la ejecución de un servicio.

Tabla 6: Botones de la interfaz de monitorización y gestión

5 Implementación

Después de ver en el capítulo anterior la especificación de las API y el diseño de la interfaz de monitorización y gestión de servicios, se explican a continuación los detalles de implementación tanto de los servicios y sus API como de la interfaz.

5.1 Implementación de los servicios

Los servicios se han implementado con el lenguaje de programación Python y las API se han desarrollado utilizando el framework Bottle, un micro web-framework *WSGI* (Web Server Gateway Interface) para Python rápido, simple y ligero.

Como ya se ha visto en el diseño, se ha utilizado el formato JSON para intercambiar mensajes entre el cliente y las API. Se ha decidido utilizar este formato ya que resulta más sencillo y ligero de utilizar que otros formatos de intercambio de datos como por ejemplo XML.

Las API de los servicios se han creado utilizando *decorators*, que nos permiten definir las rutas de la url que vamos a tratar. Para cada ruta o conjunto de rutas se define una función que se ejecutará al acceder a la ruta especificada. Los principales decorators que se han utilizado para la implementación de las API son @get, @put y @delete. A continuación se muestra un pequeño ejemplo de cómo se definen las rutas de las url en Bottle.

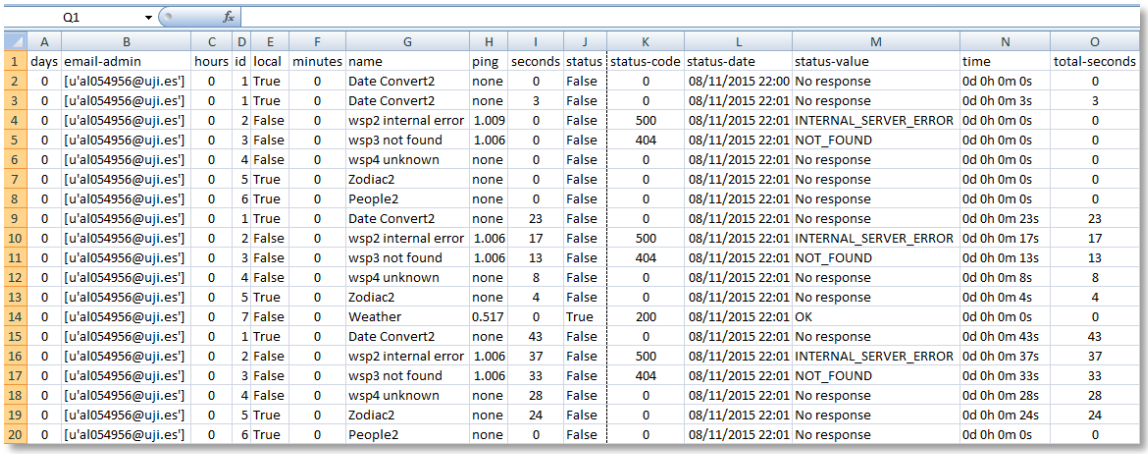
```
1 from bottle import get, run

2 @get('/restwebservices')
3 def rws():
4      #Código a ejecutar
5 run(host='localhost', port=8080)
```

En este ejemplo se define una ruta en la línea 2. Esta línea significa que cuando el cliente realice una petición GET sobre esa ruta se ejecutará la función asociada, que en este caso es *rws*. Para ejecutar la función desde un navegador se debe escribir la url: *http://localhost:8080/restwebservices*.

Por otra parte, los datos de las descripciones de los servicios y de las descripciones de monitorización descritas en el diseño se almacenan en diccionarios. Estas estructuras se guardan en ficheros de texto porque la inserción de datos es muy rápida ya que simplemente se van añadiendo al final del fichero. Sin embargo, las operaciones de modificación y borrado son más costosas ya que se tiene que leer el fichero, crear una lista con los datos recuperados, recorrerla para encontrar el dato que se quiere modificar o borrar y volver a guardar todos los datos de la lista en el fichero. No obstante, como las estructuras de datos que se han de almacenar son sencillas se ha elegido este sistema, que es fácil de implementar si se compara con la creación de una base de datos.

Además, los informes de monitorización se guardan con formato CSV en un fichero de texto. Para ello se utiliza la librería *csv*. Se ha decidido utilizar este formato ya que existen muchas herramientas que lo utilizan para hacer análisis.

Además, el histórico de informes se almacena en formato CSV para poder analizarlo a posteriori por otros programas como por ejemplo las hojas de cálculo. Esto se ha implementado utilizando la librería de python *csv.* En la figura 7 se puede ver un ejemplo de histórico de informes en Excel.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	days	email-admin	hours	id	local	minutes	name	ping	seconds	status	status-code	status-date	status-value	time	total-seconds
2	0	[u'al054956@uji.es']	0	1	True	0	Date Convert2	none	0	False	0	08/11/2015 22:00	No response	0d 0h 0m 0s	0
3	0	[u'al054956@uji.es']	0	1	True	0	Date Convert2	none	3	False	0	08/11/2015 22:01	No response	0d 0h 0m 3s	3
4	0	[u'al054956@uji.es']	0	2	False	0	wsp2 internal error	1.009	0	False	500	08/11/2015 22:01	INTERNAL_SERVER_ERROR	0d 0h 0m 0s	0
5	0	[u'al054956@uji.es']	0	3	False	0	wsp3 not found	1.006	0	False	404	08/11/2015 22:01	NOT_FOUND	0d 0h 0m 0s	0
6	0	[u'al054956@uji.es']	0	4	False	0	wsp4 unknown	none	0	False	0	08/11/2015 22:01	No response	0d 0h 0m 0s	0
7	0	[u'al054956@uji.es']	0	5	True	0	Zodiac2	none	0	False	0	08/11/2015 22:01	No response	0d 0h 0m 0s	0
8	0	[u'al054956@uji.es']	0	6	True	0	People2	none	0	False	0	08/11/2015 22:01	No response	0d 0h 0m 0s	0
9	0	[u'al054956@uji.es']	0	1	True	0	Date Convert2	none	23	False	0	08/11/2015 22:01	No response	0d 0h 0m 23s	23
10	0	[u'al054956@uji.es']	0	2	False	0	wsp2 internal error	1.006	17	False	500	08/11/2015 22:01	INTERNAL_SERVER_ERROR	0d 0h 0m 17s	17
11	0	[u'al054956@uji.es']	0	3	False	0	wsp3 not found	1.006	13	False	404	08/11/2015 22:01	NOT_FOUND	0d 0h 0m 13s	13
12	0	[u'al054956@uji.es']	0	4	False	0	wsp4 unknown	none	8	False	0	08/11/2015 22:01	No response	0d 0h 0m 8s	8
13	0	[u'al054956@uji.es']	0	5	True	0	Zodiac2	none	4	False	0	08/11/2015 22:01	No response	0d 0h 0m 4s	4
14	0	[u'al054956@uji.es']	0	7	False	0	Weather	0.517	0	True	200	08/11/2015 22:01	OK	0d 0h 0m 0s	0
15	0	[u'al054956@uji.es']	0	1	True	0	Date Convert2	none	43	False	0	08/11/2015 22:01	No response	0d 0h 0m 43s	43
16	0	[u'al054956@uji.es']	0	2	False	0	wsp2 internal error	1.006	37	False	500	08/11/2015 22:01	INTERNAL_SERVER_ERROR	0d 0h 0m 37s	37
17	0	[u'al054956@uji.es']	0	3	False	0	wsp3 not found	1.006	33	False	404	08/11/2015 22:01	NOT_FOUND	0d 0h 0m 33s	33
18	0	[u'al054956@uji.es']	0	4	False	0	wsp4 unknown	none	28	False	0	08/11/2015 22:01	No response	0d 0h 0m 28s	28
19	0	[u'al054956@uji.es']	0	5	True	0	Zodiac2	none	24	False	0	08/11/2015 22:01	No response	0d 0h 0m 24s	24
20	0	[u'al054956@uji.es']	0	6	True	0	People2	none	0	False	0	08/11/2015 22:01	No response	0d 0h 0m 0s	0

Figura 7: Informe de monitorización en Excel

En cuanto a la configuración de los servicios, se ha decidido crear un fichero de configuración que también se guardará como un fichero de texto. Para manejar este fichero se ha utilizado la librería *Configparser* ya que cuenta con métodos para acceder fácilmente a los datos de configuración.

Por último, para registrar las acciones y errores que se pueden producir en la ejecución de los servicios se crean ficheros log. Estos ficheros se crean automáticamente usando la librería *logging*.

Estructura de ficheros

Los tipos de fichero que se utilizan son .py para el código Python, .cfg para la configuración, .json y .txt para los diccionarios de las descripciones y .log para el registro de las acciones realizadas y los errores. A continuación, se explican los archivos que utilizan cada uno de los servicios.

Servicio para la gestión de las descripciones de servicios:

- **restwebservices.py**: es el script de Python que contiene todo el código implementado del servicio.
- **restwebservices_config.cfg**: es el fichero de configuración del servicio y debe estar ubicado en la misma carpeta donde se ejecuta el script del servicio. En la configuración se definen los parámetros de configuración del servicio y su estructura de ficheros.

- **service_description.json**: es un fichero de texto que contiene un JSON con la estructura de la descripción de los servicios. Esta estructura se utiliza para validar las descripciones que envía el cliente al crear una nueva descripción.

Con la configuración actual al ejecutar el servicio se crea una carpeta llamada **restwebservices,** donde se almacenan los ficheros generados por el servicio. La ubicación de la esta carpeta se define en el fichero de configuración y contiene:

- Una carpeta llamada **data** donde se guardan dos ficheros de datos:
  - **restwebservices_data.txt:** almacena las descripciones de los servicios.
  - **restwebservices_id.txt:** almacena el último identificador (un número) que se ha utilizado para identificar las descripciones de los servicios.
- **restwebservices_debug.log:** es el fichero donde se registran los mensajes de información y de error que se producen durante la ejecución del servicio.

Servicio para la gestión de las descripciones de monitorización:

- **rwsmonitoring.py**: es el script de Python que contiene todo el código implementado del servicio.
- **rwsmonitoring_config.cfg**: es el fichero de configuración del servicio y debe estar ubicado en la misma carpeta donde se ejecuta el script del servicio. En la configuración se definen los parámetros de configuración del servicio y su estructura de ficheros.
- **monitoring_description.json**: es un fichero de texto que contiene un JSON con la estructura de la descripción de monitorización. Esta estructura se utiliza para validar las descripciones de monitorización que envía el cliente al crear una nueva descripción de monitorización.

Con la configuración actual al ejecutar el servicio se crea una carpeta llamada **rwsmonitoring,** donde se almacenan los ficheros generados por el servicio. La ubicación de esta carpeta se define en el fichero de configuración y contiene:

- **monitoring_data.txt:** almacena las descripciones de monitorización de los servicios.
- **monitoring_id.txt:** almacena el último identificador (un número) que se ha utilizado para identificar las descripciones de los servicios.
- **rwsmonitoring_debug.log:** es el fichero donde se registran los mensajes de información y de error que se producen durante la ejecución del servicio.
- **report.csv:** contiene el histórico de informes guardados en formato csv.
- **monitoring_lastreport.json:** contiene el último informe de monitorización que se ha realizado.
- **fallen_alert_waiting.data:** contiene la lista de servicios caídos que están a la espera de que pase el tiempo establecido para enviar el correo de alerta.
- **new_inactives_alert_waiting.data:** contiene la lista de los nuevos servicios inactivos que están a la espera de que pase el tiempo establecido para enviar el correo de alerta.
- **recovered_alert_waiting.data:** contiene la lista de servicios recuperados que están a la espera de que pase el tiempo establecido para enviar el correo de alerta.

Servicio para la gestión de servicios:

- **rwsmanage.py**: es el script de Python que contiene todo el código implementado del servicio.
- **rwsmanage_config.cfg**: es el fichero de configuración del servicio y debe estar ubicado en la misma carpeta donde se ejecuta el script del servicio. En la configuración se definen los parámetros de configuración del servicio y su estructura de ficheros.

Con la configuración actual al ejecutar el servicio se crea una carpeta llamada **rwsmanage** donde se almacenan los ficheros generados por el servicio. La ubicación de la esta carpeta se define en el fichero de configuración y contiene:

- **restwebservices_data.txt:** almacena la lista de servicios que se han ejecutado.
- **restwebservices_debug.log:** es el fichero donde se registran los mensajes de información y de error que se producen durante la ejecución del servicio.

Para poner en marcha el servicio de descripción de servicios se ha de ejecutar el archivo restwebservices.py. Se ha de tener en cuenta que en la carpeta donde se ejecute este archivo deben estar los ficheros restwebservices_config.cfg y service_description.json.

Para ejecutar el servicio de gestión y monitorización se ha de ejecutar previamente el servicio de descripción de servicios ya que necesita consultar la lista de descripciones de servicios. Una vez hecho esto, se ejecutan los archivos rwsmonitoring.py y rwsmanage.py. Se ha de tener en cuenta que en la carpeta donde se ejecuten este archivos deben estar los ficheros rwsmonitoring_config.cfg, rwsmanage_config.cfg y monitoring_description.json.

5.2 Implementación de la interfaz

La interfaz de monitorización y gestión de servicios se ha implementado usando HTML5, CSS3, Javascript y el framework *Bootstrap* para el diseño de la interfaz.

Se utilizan formularios para recoger la información de las descripciones de monitorización. Estos datos se almacenan en un diccionario y se envían mediante la llamada correspondiente a la API de monitorización de servicios.

Además, la pantalla de monitorización que muestra la información de cada servicio, se crea de forma dinámica a partir de la lista de informes que se obtiene de la API de monitorización.

Por último, todas las operaciones que se pueden realizar mediante la interfaz se implementan haciendo la llamada correspondiente a la API de monitorización y gestión tal y como se explica en su especificación. Para realizar estas llamadas se hacen peticiones HTTP usando AJAX con XMLHttpRequest.

Estructura de ficheros

Los tipos de ficheros que se utilizan en la interfaz son HTML, CSS y Javascript. A continuación se describe cada uno de ellos:

- Ficheros HTML:
  - **index.html:** es el principal archivo de la interfaz en el que se encuentra el código HTML para mostrar la pantalla de monitorización.
  - **adddescription.html:** archivo que tiene el HTML necesario para mostrar el formulario de alta de la descripción de monitorización de un servicio.
  - **editdescription.html:** archivo que tiene el HTML necesario para mostrar el formulario de modificación de la descripción de monitorización de un servicio.
  - **viewdescription.html:** archivo que tiene el HTML necesario para mostrar la descripción de monitorización de un servicio.
- Ficheros CSS:
  - **style.css:** contiene los estilos generales de la aplicación.
  - **form.css:** contiene los estilos de los formularios.
- Ficheros Javascript:
  - **config.js:** archivo en el que se especifican los valores de las url necesarias para hacer las llamadas a las API.
  - **createdescription.js:** contiene la función para crear las descripciones de monitorización.
  - **editdescription.js:** contiene las funciones para cargar y modificar la información de las descripciones de monitorización.
  - **deletedescription.js:** contiene la función para eliminar la descripción de monitorización de un servicio.
  - **viewdescription.js:** contiene la función para visualizar la información de una descripción de monitorización.
  - **getdescriptions.js:** contiene las funciones para obtener el listado de todas las descripciones de monitorización.
  - **stopstartservice.js:** contiene las funciones para arrancar y detener un servicio.
  - **updatereport:** contiene la función para actualizar un informe.
  - **validationforms:** contiene la función para validar los formularios de alta y modificación de una descripción de monitorización.
  - **varietyfunctions:** contiene varias funciones de uso general.

6 Pruebas

Después de realizar la implementación, tanto de las API como de la interfaz de usuario para la monitorización y gestión de los servicios, en este apartado se describen las pruebas que se han realizado para comprobar su correcto funcionamiento.

6.1 API para la descripción de servicios

Para comprobar el correcto funcionamiento de la API para la descripción de servicios se han realizado las siguientes pruebas:

- Se ha pedido un listado de todas las descripciones almacenadas en el servidor y se ha comprobado que se recupera correctamente.
- Se ha pedido una descripción de un servicio concreto y se ha comprobado que se recupera correctamente.
- Se ha pedido una descripción de un servicio que no existe y se ha comprobado que se recibe un mensaje de error.
- Se ha solicitado la creación de una descripción enviando un JSON válido y se ha comprobado que se ha almacenado correctamente.
- Se ha solicitado la creación de una descripción enviando un JSON inválido y se ha comprobado que se recibe un mensaje de error.
- Se ha solicitado la creación de una descripción sin enviar datos y se ha comprobado que se recibe un mensaje de error.
- Se ha intentado crear una descripción con un nombre que ya existe y se ha comprobado que se recibe un mensaje de error.
- Se ha realizado una petición para modificar una descripción ya existente con un JSON válido y se ha comprobado que se modifica correctamente.
- Se ha realizado una petición para modificar una descripción ya existente con un JSON inválido y se ha comprobado que se recibe un mensaje de error.
- Se ha realizado una petición para modificar una descripción ya existente sin enviar datos y se ha comprobado que se recibe un mensaje de error.
- Se ha realizado una petición para modificar una descripción que no existe y se ha comprobado que se recibe un mensaje de error.
- Se ha solicitado el borrado de una descripción existente y se ha comprobado que se borra correctamente del servidor.
- Se ha solicitado el borrado de una descripción que no existe y se ha comprobado que se recibe un mensaje de error.

6.2 API para la monitorización y gestión de servicios

Para comprobar el correcto funcionamiento de la API para la monitorización y gestión de servicios se han realizado las siguientes pruebas:

- Para el listado, creación, modificación y borrado de las descripciones de monitorización se han seguido las mismas pruebas realizadas en la API para de descripción de servicios y se han obtenido los mismos resultados.
- Se ha enviado una descripción de monitorización con un nombre de servicio que existe en el repositorio de descripciones de servicios y se ha comprobado que se crea correctamente.
- Se ha enviado una descripción de monitorización con un nombre de servicio que no existe en el repositorio de descripciones de servicios y se ha comprobado que se recibe un mensaje de error.
- Después de haber ejecutado dos servicios en el servidor local se ha solicitado el listado de servicios que se están ejecutando actualmente y se ha comprobado que se ha obtenido el listado correctamente con los servicios ejecutados.
- Se ha enviado una solicitud de ejecución de un servicio local que existe en el repositorio de descripciones de monitorización y que tiene una dirección correcta donde se encuentra el script del servicio y se ha comprobado que se ha ejecutado correctamente el servicio.
- Se ha enviado una solicitud de ejecución de un servicio local que existe en el repositorio de descripciones de monitorización pero que la dirección del script no es correcta y se ha comprobado que se ha recibido un mensaje de error.
- Se ha enviado una solicitud de ejecución de un servicio local que existe en el repositorio de descripciones de monitorización pero con la dirección del script vacía y se ha comprobado que se ha recibido un mensaje de error.
- Se ha enviado una solicitud de ejecución de un servicio local que ya había sido ejecutado y se ha comprobado que se ha recibido un mensaje de error.
- Se ha enviado una solicitud de ejecución de un servicio que no existe en el repositorio de descripciones de monitorización y se ha comprobado que se recibe un mensaje de error.
- Se ha solicitado la parada de un servicio, previamente ejecutado, y se ha comprobado que se ha parado correctamente.
- Se ha solicitado la parada de un servicio, que no se ha ejecutado, y se ha comprobado que se recibe un mensaje de error.
- Después de que la API de monitorización hubiera realizado varios informes se ha pedido el último informe guardado en el servidor y se ha comprobado que se obtiene correctamente.
- Antes de que la API realizara algún informe se ha pedido el último informe guardado en el servidor y se ha comprobado que se recibe un mensaje de error.
- Se ha solicitado la creación de un informe nuevo y se ha comprobado que éste reemplaza al último guardado y además se registra en el histórico de informes en formato CSV.
- Se ha comprobado que se ejecuta automática y periódicamente la creación de informes según lo especificado en el fichero de configuración.
- Se ha comprobado que cuando un servicio se cae o se recupera y ha pasado el tiempo estipulado se envía correctamente un correo de alerta al administrador de ese servicio salvo que dicho correo esté incluido en la lista de exclusión.

6.3 Interfaz para la monitorización y la gestión

Dado que esta interfaz utiliza la API de monitorización y gestión y las llamadas que realiza a ésta se han comprobado en el apartado anterior, las pruebas han consistido simplemente en comprobar que hace las llamadas correspondientes según la operación solicitada (listado, creación, modificación, borrado, visualización, actualización del informe, ejecución y parada).

7 Conclusiones y trabajo futuro

En este proyecto se han diseñado y desarrollado dos API, una para la descripción de servicios y otra para su monitorización y gestión. La API para la descripción de servicios tiene como finalidad recopilar la información necesaria para que los servicios se puedan ejecutar. La API para la monitorización y la gestión sirve por una parte, para controlar el estado en el que se encuentran los servicios y así poder extraer información relevante para la toma de decisiones y por otra parte, para ejecutar y parar servicios locales de forma cómoda. Además, también se ha implementado una interfaz web que sirve para utilizar la API de monitorización y gestión de servicios de forma gráfica.

La implementación de las API se ha realizado siguiendo el estilo arquitectónico REST que como se vio en el estado del arte es más ligero y fácil de implementar que SOAP. Para la descripción de servicios, SOAP obliga a utilizar WSDL, en cambio, con REST se puede utilizar cualquier tipo de descripción. En nuestro caso, se han descrito los servicios en formato JSON.

Como trabajo futuro se quieren realizar mejoras tanto en la implementación de los servicios como en la interfaz. En la implementación de los servicios se puede añadir autenticación con tokens para mejorar la seguridad. La interfaz se puede mejorar ofreciendo la posibilidad de ordenar los datos de la tabla del informe de monitorización por columnas.

Además, las descripciones se podrían enriquecer añadiendo anotaciones semánticas para facilitar el descubrimiento y la composición de servicios. También sería interesante poder hacer la descripción semántica de servicios de forma automática o semiautomática.

8 Bibliografía

[1] Qué es Python. (2003, 19 de noviembre). Miguel Angel Alvarez. Fecha de consulta: noviembre 16, 2015 desde http://www.desarrolloweb.com/articulos/1325.php

[2] JSON. (2015, 11 de agosto). Wikipedia, La enciclopedia libre. Fecha de consulta: noviembre 16, 2015 desde https://es.wikipedia.org/w/index.php?title=JSON&oldid=84361566.

[3] Introducción a JSON .Fecha de consulta: noviembre 16, 2015 desde http://www.json.org/json-es.html

[4] Hypertext Transfer Protocol. (2015, 23 de octubre). Wikipedia, La enciclopedia libre. Fecha de consulta: noviembre 16, 2015 desde https://es.wikipedia.org/w/index.php?title=Hypertext_Transfer_Protocol&oldid=86045199.

[5] TwitterBootstrap. (2015, 13 de noviembre). Wikipedia, La enciclopedia libre. Fecha de consulta: noviembre 16, 2015 desde https://es.wikipedia.org/w/index.php?title=Twitter_Bootstrap&oldid=86863327.

[6] JavaScript. (2015, 15 de noviembre). Wikipedia, La enciclopedia libre. Fecha de consulta: noviembre 16, 2015 desde https://es.wikipedia.org/w/index.php?title=JavaScript&oldid=86898219.

[7] AJAX. (2015, 21 de agosto). Wikipedia, La enciclopedia libre. Fecha de consulta: noviembre 16, 2015 desde https://es.wikipedia.org/w/index.php?title=AJAX&oldid=84580040**.**

[8] Bottle. Fecha de consulta: noviembre 16, 2015 desde http://bottlepy.org/docs/dev/index.html

[9] WSGI. Fecha de consulta: noviembre 16, 2015 desde http://python.org.ar/wiki/WSGI

[10] Servicios Web.W3C. Fecha de consulta: agosto 15, 2015 desde http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb

[11] Representational State Transfer. (2013, 27 de septiembre). Wikipedia, La enciclopedia libre. Fecha de consulta: agosto 12, 2015 desde http://es.wikipedia.org/w/index.php?title=Representational_State_Transfer&oldid=77226353.

[12] LANTHALER, Markus; GÜTL, Christian. Seam less integration of RESTful services into the web of data. Advances in Multimedia, 2012, vol. 2012, p. 1.

[13] Simple Object Access Protocol. (2013, 1 de diciembre). Wikipedia, La enciclopedia libre. Fecha de consulta: agosto 11, 2015 desde http://es.wikipedia.org/w/index.php?title=Simple_Object_Access_Protocol&oldid=78471334.

[14] ELP-DSIC-UPV, R. N. M. REST vs Web Services.

[15] Fielding, R. T. (2000). Architectural styles and the design of network-based software architectures (Doctoral dissertation, University of California, Irvine).

[16] Servicios Web. María Jesús Lamarca Lapuente. Fecha de consulta: agosto 18, 2015 desde http://www.hipertexto.info/documentos/serv_web.htm

[17] Service Description. W3C. Fecha de consulta: agosto 18, 2015 desdehttp://www.w3.org/standards/webofservices/description

[18]Semantic Annotations for WSDL and XML Schema. (2007, 28 de agosto). W3C. Fecha de consulta: agosto 19, 2015 desde http://www.w3.org/standards/techs/sawsdl#w3c_all

[19] Web Services Architecture. (2004, 11 de febrero). W3C. Fecha de consulta: agosto 19, 2014 desde http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#service_management

[20]Monitoreo y gestión de servicios durante el tiempo deejecución. (2011, 5 de agosto). YuChang Jiao. Fecha de consulta: septiembre 3, 2014 desde http://www.ibm.com/developerworks/ssa/webservices/library/ws-servicemonitoring/ws-servicemonitoring-pdf.pdf

[21] ProgrammableWeb. Fecha de consulta: noviembre 15, 2015 desde http://www.programmableweb.com/apis/directory

[22] Galileo. Fecha de consulta: noviembre 15, 2015 desde https://getgalileo.io

[23] SSWAP. Fecha de consulta: noviembre 15, 2015 desde http://sswap.info/ipc/7614833f-a18e-4131-a4ee-ceaeaddf0b6e

Anexo I: Configuración del servicio para la descripción de servicios

En este anexo se describen los parámetros de configuración del servicio para la descripción de servicios REST.

El fichero de configuración restwebservices_config.cfg se divide en tres secciones:

- **DEFAULT**: esta sección contiene los parámetros que se pueden interpolar al resto de secciones. De esta forma una vez definimos un parámetro podremos hacer referencia a él en cualquier sección del fichero de configuración. A continuación se describen los parámetros que contiene:

    - **dir_service**: contiene el nombre de la carpeta donde se guardarán los ficheros del servicio.
    - **dir_data**: contiene el nombre de la carpeta donde se guardarán los ficheros de datos.
    - **file_data**: contiene el nombre del fichero donde se guardarán los datos.
    - **file_id**: contiene el nombre del fichero donde se guardará el último identificador usado en las descripciones.
    - **file_log**: contiene el nombre del fichero log del servicio.
    - **file_json**: contiene el nombre del fichero que contiene el JSON que se utiliza para validar las descripciones recibidas.

- **folders**: esta sección tiene como parámetros números enteros a los que se le asigna el nombre de una carpeta. Todas las carpetas que aparezcan en esta sección se crearán automáticamente al ejecutar el servicio.
- **files**: en esta sección se especifica la ruta donde se encuentran los ficheros que utiliza el servicio. A continuación se describen los parámetros que contiene:

    - **datafile:** contiene la ruta donde se encuentra el fichero file_data.
    - **idfile:** contiene la ruta donde se encuentra el fichero file_id.
    - **logfile:** contiene la ruta donde se encuentra el fichero file_log.
    - **jsonfile:** contiene la ruta donde se encuentra el fichero file_json.

Anexo II: Configuración del servicio para la monitorización de servicios

En este anexo se describen los parámetros de configuración del servicio para la monitorización de servicios REST.

El fichero de configuración rwsmonitoring_config.cfg se divide en cuatro secciones:

- **DEFAULT**: esta sección contiene los parámetros que se pueden interpolar al resto de secciones. De esta forma una vez definimos un parámetro podremos hacer referencia a él en cualquier sección del fichero de configuración. A continuación se describen los parámetros que contiene:

    - **dir_service**: contiene el nombre de la carpeta donde se guardarán los ficheros del servicio.
    - **file_log**: contiene el nombre del fichero log del servicio.
    - **setinterval**: contiene la cantidad de tiempo en segundos que transcurre entre la creación de un informe y otro
    - **file_lastreport**: contiene el nombre del fichero donde se guardará el último informe creado.
    - **file_jsondescription**: contiene el nombre del fichero que contiene el JSON que se utiliza para validar las descripciones de monitorización recibidas
    - **file_data**: contiene el nombre del fichero donde se guardarán los datos.
    - **file_id**: contiene el nombre del fichero donde se guardará el último identificador usado en las descripciones de monitorización.
    - **file_fallen:** contiene el nombre del fichero que guarda la lista de espera de los servicios caídos.
    - **file_recovered:** contiene el nombre del fichero que guarda la lista de espera de los servicios recuperados
    - **file_new_inactives:** contiene el nombre del fichero que guarda la lista de espera de los nuevos servicios inactivos.
    - **file_csv:** contiene el nombre del fichero que guarda el histórico de los informes de monitorización en formato csv.
- **email:** esta sección recoge los parámetros necesarios para que el servicio pueda enviar correos. También se incluyen otros parámetros para controlar el comportamiento del servicio a la hora de enviar correos. A continuación se describen los parámetros que contiene:
    - **user:** contiene en nombre de la cuenta de correo que se utilizará para enviar los correos**.**
    - **password:** contiene la contraseña de la cuenta de correo.
    - **smtpserver:** contiene nombre del servidor de la cuenta de correo.
    - **smtpport:** contiene el puerto del servidor de la cuenta de correo.
    - **sender:** contiene correo del remitente del mensaje.

- **recipients:** contiene una lista de correos de los destinatarios.
- **exclude_list:** contiene una lista de correos. Los correos que aparecen en esta lista no recibirán ningún correo.
- **setinterval_email:** contiene la cantidad de segundos que el servicio se ha de esperar entre cada envío de correo.
- **downtime:** contiene la cantidad de tiempo en segundos que el servicio esperará antes de enviar un correo de alerta. Esto se hace para evitar que lleguen correos de alerta cuando se producen caídas de los servicios que duran muy poco tiempo.

- **folders**: esta sección tiene como parámetros números enteros a los que se le asigna el nombre de una carpeta. Todas las carpetas que aparezcan en esta sección se crearán automáticamente al ejecutar el servicio.
- **files**: en esta sección se especifica la ruta donde se encuentran los ficheros que utiliza el servicio. A continuación se describen los parámetros que contiene:

  - **datafile:** contiene la ruta donde se encuentra el fichero file_data.
  - **idfile:** contiene la ruta donde se encuentra el fichero file_id.
  - **logfile:** contiene la ruta donde se encuentra el fichero file_log.
  - **jsonfile:** contiene la ruta donde se encuentra el fichero file_json.
  - **lastreportfile:** contiene la ruta donde se encuentra el fichero file_lastreport**.**
  - **jsondescfile:** contiene la ruta donde se encuentra el fichero file_jsondescription.
  - **fallenfile:** contiene la ruta donde se encuentra el fichero file_fallen
  - **recoveredfile:** contiene la ruta donde se encuentra el fichero file_recovered**.**
  - **newinactivesfile:** contiene la ruta donde se encuentra el fichero file_new_inactives**.**
  - **csvfile:** contiene la ruta donde se encuentra el fichero file_csv.

Anexo III: Configuración del servicio para la gestión de los servicios

En este anexo se describen los parámetros de configuración del servicio para la gestión de servicios REST.

El fichero de configuración rwsmanage_config.cfg se divide en tres secciones:

- **DEFAULT:** esta sección contiene los parámetros que se pueden interpolar al resto de secciones. De esta forma una vez definimos un parámetro podremos hacer referencia a él en cualquier sección del fichero de configuración. A continuación se describen los parámetros que contiene:

    - **dir_service**: contiene el nombre de la carpeta donde se guardarán los ficheros del servicio.
    - **file_data:** contiene el nombre del fichero donde se guardarán los datos.
    - **file_log:** contiene el nombre del fichero log del servicio.
    - **rws_monitoring_endpoint:** contiene el endpoint para acceder a la lista de descripciones de monitorización.
    - **rws_monitoring_report_endpoint:** contiene el endpoint que se utiliza para actualizar el informe de un servicio.

- **folders:** esta sección tiene como parámetros números enteros a los que se le asigna el nombre de una carpeta. Todas las carpetas que aparezcan en esta sección se crearán automáticamente al ejecutar el servicio.
- **files:** en esta sección se especifica la ruta donde se encuentran los ficheros que utiliza el servicio. A continuación se describen los parámetros que contiene:

    - **datafile:** contiene la ruta donde se encuentra el fichero file_data.
    - **logfile:** contiene la ruta donde se encuentra el fichero file_log.