ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

Monitorización a nivel de Plataforma de servicios desplegados en la Nube

Monitoring Platform of Cloud Services

Realizado por

Samuel Gómez Cabrera

Tutorizado por

Francisco Javier Durán Muñoz

Departamento

Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, Julio 2015

Fecha defensa:
El Secretario del Tribunal

Resumen

El desarrollo y crecimiento de tecnologías basadas en *Cloud Computing* o Computación en la Nube nos proporciona un nuevo modelo de entender el desarrollo y computación de aplicaciones y servicios. Este modelo está basado en internet y permite poner al alcance de usuarios y desarrolladores de software un conjunto de recursos que pueden utilizar remotamente sin el esfuerzo derivado de la implantación y mantenimiento de equipos físicos. Así, los distintos proveedores de servicios de computación en la Nube proporcionan alta disponibilidad, flexibilidad, y diferentes capas de abstracción con diferentes características y recursos bajo demanda.

A pesar de las numerosas ventajas de la Nube, el despliegue de aplicaciones y servicios trae de la mano nuevos desafíos. Cuestiones como la interoperabilidad entre distintas Nubes, o la reconfiguración y gestión dinámica de servicios son algunos de los problemas que es necesario abordar de forma eficiente. En este trabajo, se desarrolla una infraestructura para dar soporte a la gestión y monitorización de servicios distribuidos en diferentes nubes. En concreto, se utilizan mecanismos para la especificación de componentes y *sensores* para su despliegue a través de diferentes proveedores de Nube. La monitorización de la plataforma subyacente a cada aplicación es realizada a través de estos sensores, encargados de recopilar información sensible de análisis. Los datos recopilados de cada componente permiten a los usuarios tomar decisiones de reconfiguración de sus aplicaciones, y ejecutarlas a través de tecnologías que permiten cambiar el estado del servicio.

Palabras claves: Computación en la Nube, Monitorización,

Abstract

Cloud Computing is a rapidly growing area that provide a new way to understand software and applications development. This Internet-based model provides a set of resources that can manage remotely without the investment needed to deploy and maintain their own infrastructure, to both users and software developers. Thus, Cloud computing providers offer high availability, flexibility, and different abstraction layers with different features and on-demand resources.

Despite cloud benefits, deployment of applications and services brings new challenges to address. Issues such as interoperability between clouds, reconfiguring tasks and dynamic services management are part of the problems that we need to deal with. This project developed an infrastructure to managing and monitoring distributed systems among different Clouds. In particular, a set of technologies have been developed in order to specify components and sensors to be deployed through different Cloud providers. Monitoring the underlying platform of each application is carried out by sensors, which are responsible of collecting value data. The information obtained supports users to make robust decisions to reconfigure their applications.

Keywords: Cloud Computing, monitoring

Índice

Índice de figuras

Índice de tablas

1. Introducción

1.1. Motivación

Actualmente existen multitud de proveedores de Cloud que ofertan sus recursos de computación mediante la modalidad de pago por servicio o mediante una cuota fija durante cierto tiempo (mensual, anual). Ofrecen a su vez diferentes capas de abstracción: IaaS, PaaS y Saas.

- *Infraestructura como servicio* o Infraestructure as a Service (IaaS)
- *Plataforma como servicio* o Platform as a Service (PaaS)
- *Software como servicio* o Software as a Service (SaaS)

Las oportunidades y ventajas que ofrece este nuevo paradigma resultan muy amplias y nos proporcionan no sólo la capacidad de disponer de determinados recursos computacionales en un instante determinado de tiempo, sino también poder reservar y usar estos recursos bajo demanda. Sin embargo, cuestiones como la dependencia de un proveedor determinado o la gestión dinámica y escalabilidad de estos recursos es un factor importante a tener en cuenta tanto en términos económicos como de rendimiento y calidad del servicio.

Saber adaptarse, y reaccionar al cambio en un entorno tan heterogéneo como el de la Nube es un factor vital para el cumplimiento de los requisitos de negocio. Así, es deseable disponer de las herramientas necesarias de cara a la toma de decisiones y para llevar a cabo la reconfiguración o migración de aplicaciones y servicios.

La monitorización resulta por tanto imprescindible en este entorno para conocer cómo se está comportando nuestra aplicación o servicio y poder decidir, por ejemplo, si adquirir más recursos, o reducir los mismos, migrar a otro proveedor, o incluso considerar un cambio en la implementación de los componentes de nuestro sistema.

Las particularidades de cada sistema, cada tecnología y cada implementación hacen necesario el uso de componentes que no sólo sean capaces de monitorizar y extraer datos basados en métricas y parámetros estándar, si no que sepan flexibles y adaptables a las necesidades del entorno y a la plataforma subyacente.

1.2. Objetivos

El presente trabajo se encuentra englobado dentro de la línea *Gestión de aplicaciones basadas en servicios en la Nube*, encaminada a la búsqueda de soluciones para distintos problemas surgidos a partir de la adopción de este paradigma, tales como, interoperabilidad[I] entre proveedores de servicios, distribución de aplicaciones en diferentes Nubes y gestión dinámica de las mismas.
El nombre escogido, *Monitorización a nivel de Plataforma de servicios desplegados en la Nube*, trata de ser claro y poner el foco en lo que se quiere conseguir a la finalización del mismo. No obstante, dado el carácter novedoso de este tipo de tecnologías y el uso de vocablos técnicos, se debe familiarizar y poner en contexto al lector sobre lo que se quiere expresar.

El concepto de *plataforma* que nombramos en el título, difiere del utilizado al definir las capas de abstracción (que se abordará más adelante), el cual ha sido traducido directamente al castellano en la literatura.
En lo que a la informática se refiere, el término plataforma aparece en la vigésimo tercera edición del diccionario de la R.A.E. [1] como: *"Entorno informático determinado, que utiliza sistemas compatibles entre sí"*. En las distintas definiciones en inglés [2, 3], se define este concepto, como la confluencia de un tipo de hardware y sistema operativo concretos. En general, se trata de una definición bastante abierta. En lo que sigue se usará la siguiente definición de plataforma: *"máquina, física o virtual, con un sistema operativo y características hardware determinadas"*.

El primer desafío a este nivel, es desarrollar nuevas herramientas de monitorización, o adaptar las existentes, para conseguir componentes lo suficientemente flexibles para poder adaptarse a la plataforma.
El tipo de monitorización que se deba realizar también vendrá determinada por la plataforma, el tipo de servicio y debe aportar información con las siguientes características:

- adaptable a determinadas reglas y/o restricciones (QoS[II], necesidades específicas, etc .)
- la información debe ser completa, robusta y precisa
- adecuada, legible e interpretable por humanos y máquinas
- finalmente, debe ser sensible de análisis de cara a la toma de decisiones

[I] La interoperabilidad es la capacidad de dos o más sistemas, o componentes para intercambiar información y usar la información que se ha intercambiado. Su introducción en el vocabulario relativo al ámbito de las tecnologías de la información se debe a la traducción del término inglés *interoperability*.
[II] Quality of Service - Calidad de Servicio

El principal objetivo final será poder conocer el estado de un servicio desplegado[III] en la Nube, por ello, la flexibilidad y adaptabilidad debe darse a estos dos niveles.

1.3. Estructura de la memoria

En la siguiente sección se introducirá con mayor detalle el concepto de *Computación en la Nube*, así como su recorrido, estado reciente y tecnologías desarrolladas y encaminadas a resolver problemas similares a los que aquí se plantean. Se profundizará también en la monitorización de sistemas, herramientas existentes y trabajos desarrollados al respecto

En la sección 3 se ahondará en todo lo referente al análisis, especificación, diseño e implementación resultado de este trabajo.

Se dedicará la sección 4 a exponer los distintos resultados obtenidos haciendo uso de la herramienta desarrollada, indicando los procesos y tareas llevados a cabo.

La sección 5 finalmente contiene la interpretación de los resultados obtenidos a lo largo del desarrollo de este proyecto, los hechos y conclusiones que se derivan del mismo.

[III] En software, *desplegar* hace referencia a las actividades encaminadas a hacer una aplicación o servicio disponible para su uso.

2. Estado del arte

2.1. Computación en la Nube

Computación en la Nube o *Cloud-Computing* es un término relativamente reciente en el mundo de las TIC's (Tecnologías de la Información), pero que poco a poco se está convirtiendo en un referente y un modelo a seguir para el despliegue de aplicaciones por parte de las empresas.

2.1.1. Antecedentes

Aunque la denominación sí lo es, la idea de Computación en la Nube no es nueva y viene de varias décadas atrás. Concretamente, se le atribuye al Profesor John McCarthy [4] quien en 1961, durante un discurso para celebrar el centenario del MIT (Massachusetts Institute of Technology), sugirió que la tecnología de tiempo compartido (Time-Sharing) podría evolucionar hacia un futuro donde la computación e incluso aplicaciones específicas podrían venderse como un servicio.

Más tarde, en 1969, Leonard Kleinrock uno de los científicos a la cabeza de ARPANET (Advanced Research Projects Agency Network), habló de *"servicios de computación"*de los cuales disfrutaríamos, al igual que servicios como electricidad o teléfono, gracias al avance de la industria de la computación en el siglo XXI [5]. En 1976, IBM anunció su producto Virtual Storage Personal Computing (VSPC) [6]. El objetivo era permitir a sus usuarios, ejecutar procesos de forma remota a través de terminales conectados a través de módems a equipos localizados en los centros de servicios de IBM. A pesar de que esta tendencia se vio mermada a principio de los 80 por la aparición de los ordenadores personales, a partir de la década de los 90 (coincidiendo con el auge de Internet), varios paradigmas recogieron el testigo con el desarrollo de modelos con características similares [7] como: *Web Hosting*, *Application Service Provider* (ASP), *Volunteer Computing*, *Online File Sharing* o *Social Networks*.

Finalmente, el avance de las redes de comunicaciones, la evolución imparable de la industria de la computación, y la virtualización de sistemas han hecho posible el desarrollo, implantación y crecimiento de este modelo.

2.1.2. Definición y características

Diferentes investigadores y entidades han tratado de dar una definición [5,7–9] estándar sobre este nuevo modelo. Sin embargo cada definición trata el concepto desde un punto de vista diferente (modelo de negocio, tecnología, servicio, etc.).

Quizás, la definición elaborada por el NIST [8] (The National Institute of Standrards and Technology) sea, además de una de las más aceptadas, la más completa:

"Computación en la Nube es un modelo que permite el acceso conveniente bajo demanda a un conjunto compartido y configurable de recursos de computo (por ejemplo, redes, servidores, almacenamiento, aplicaciones y servicios), que pueden ser rápidamente aprovisionados y liberados con un mínimo esfuerzo de gestión y de interacción con el proveedor del servicio".

A pesar de no contar con una definición estandarizada, casi todos los autores coinciden en algunas de las características más identificables:

- **Servicio bajo-demanda**: Se pone a disposición del usuario la capacidad para decidir en un momento determinado de que recursos o servicios quiere disponer. El funcionamiento automatizado hace posible que no se precise intervención humana por parte del proveedor, esta es sin duda una de las característica que mejor representan el paradigma de la Computación en la Nube.
- **Escalabilidad y elasticidad**: La posibilidad de reservar y liberar recursos con celeridad, otorga al usuario una visión en la cual dispone de un conjunto de recursos ilimitados de los cuales puede hacer uso según sus necesidades o las reglas de negocio.
- **Acceso a través de internet** : Estos servicios o recursos de computo son accesibles a través de los mecanismos, redes y protocolos estándares. Esto hace posible su uso en diversos dispositivos cliente, como PCs, móviles, tablets, etc.
- **Recursos distribuidos**: Los proveedores de servicios distribuyen o asignan de forma dinámica ese conjunto de recursos (físicos) a través de tecnologías de virtualización[I]. De esta forma, los recursos (o servicios) pueden ser reasignados en función de su uso y según se requiera por parte del usuario. Este proceso se ejecuta de forma totalmente transparente al usuario.

2.1.3. Capas de abstracción

Los centros de datos o *datacenters* de un proveedor se componen de equipos servidores físicos, la tendencia creciente [10] en las TIC es la virtualización de servidores, así, es posible disponer de multitud de servidores virtuales en ejecución en un sólo servidor físico.

Las capas de abstracción, también denominadas modelos de servicio o de negocio, determinan el tipo de recurso o *servicio* ofertado por el proveedor y del cual puede hacer uso el cliente. La capa de abstracción en la que se encuentre cada servicio, va a depender

[I]En Informática, consiste en la emulación mediante software un hardware "virtual"sobre el que montar un servicio, como puede ser un sistema operativo, un dispositivo de almacenamiento, recursos de red, etc.

del control que el usuario puede tener sobre el mismo o, dicho de otra forma, de la abstracción del usuario con respecto a lo que es controlado por el proveedor.

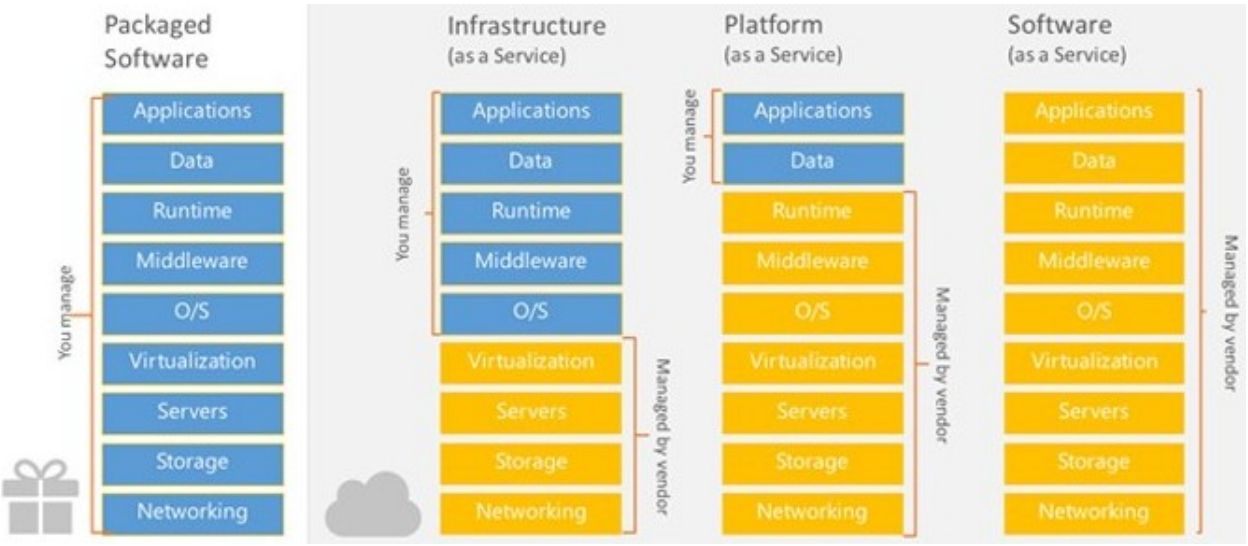


Figura 1: Capas de abstracción de la Nube y software tradicional

Como podemos apreciar en la Figura 1, cuando usamos nuestros equipos privados, disponemos del control sobre todo lo que ocurre en el mismo. No ocurre así con los distintos modelos de servicio de computación en la Nube, que explicamos a continuación:

Infraestructura como servicio (IaaS)

Tal y como se encuentra representado en la Figura 2, esta capa se encuentra en el nivel más bajo de la pirámide. En IaaS[II], los proveedores ofrecen procesamiento, almacenamiento, conexiones e interfaces de red, y otros recursos de cómputo mediante los cuales el cliente usuario es capaz de desplegar y ejecutar software. Este software puede incluir tanto sistemas operativos como aplicaciones.

En esta modalidad, el usuario tiene la percepción de estar manejando una máquina física remota, pero sin tener conciencia de la infraestructura subyacente y delegando el mantenimiento de esta al proveedor. El usuario posee el control sobre el sistema operativo, el almacenamiento de archivos y probablemente sobre la configuración de las conexiones de red.

La libertad de elegir configuraciones (componentes, redes) va a depender de las especificaciones que ofrece cada proveedor. En este tipo de servicios, hablamos normalmente de máquinas virtuales o instancias, y por lo general existe un catálogo bastante amplio

[II]En ocasiones IaaS también es llamado HaaS (Hardware as a Service).

en el que cada proveedor nos permite elegir no sólo el sistema operativo que queremos que ejecute nuestra máquina, sino también seleccionar sus componentes, tales como tipo y numero de procesadores, tamaño de memoria RAM o disco duro.

Esto lo hace especialmente conveniente para ser el sustituto natural de los equipos físicos, con la ventaja de poder adaptarlos a nuestras necesidades específicas en cada momento. Entre los proveedores de servicios que ofrecen IaaS se encuentran Amazon Web Services (AWS), Windows Azure, Rackspace, Google Compute Engine, HP o IBM.

Plataforma como servicio (PaaS)

Ocupando la capa intermedia se encuentra PaaS, destinada principalmente para el despliegue de aplicaciones o servicios específicos. En su mayoría, el catálogo de servicios del proveedor, se compone de un conjunto de herramientas específicas destinadas a alojar aplicaciones o servicios desarrollados en un lenguaje de programación o tecnología determinada. Entre estos servicios, también se incluyen los destinados a cubrir las fases del ciclo de vida de desarrollo de software.

La conciencia por parte del usuario de estar ejecutando una máquina física, a la vez que su control sobre la misma se ven reducidas drásticamente. A diferencia del modelo anterior, el conjunto de servicios constituye un grupo heterogéneo y dependiente de la implementación dada por cada proveedor. Este hecho, unido a la falta de estándares, dificulta la interoperabilidad y migración entre distintos proveedores [10], a la vez que propicia lo que se conoce como *"vendor lock-in"*[III].

Entre los de proveedores de servicios PaaS más representativos, destacan Amazon Web Services (AWS), Windows Azure, Google App Engine, Red Hat OpenShift o Heroku.

Software como servicio (SaaS)

En la cima de la pirámide, SaaS representa el conjunto de aplicaciones accesibles a través de internet como servicio. En este caso, el usuario únicamente actúa como consumidor de un servicio ofertado y que cumple una determinada función. La diferencia con el modelo tradicional, es que los usuarios o clientes se subscriben a una determinada aplicación o servicio que se distribuye a través de la red, en lugar de desarrollarla o comprarla e instalarla en sus sistemas físicos.

Este tipo de aplicaciones, pueden a su vez dar soporte o complementar a otros servicios. Desarrolladores de software o entidades, pueden decidir hacer uso de un determinado

[III]Situación en la cual el cliente que usa un producto o servicio no tiene la capacidad de migrar a otros de la competencia.

servicio accesible a través de la red para enriquecer sus propios sistemas.

El *vendor lock-in* resulta, igual que en el modelo anterior, uno de los principales obstáculos para la adopción de servicios basados en SaaS. Dejando de lado esta dependencia, otra de las cuestiones a tener en cuenta es la falta de control total sobre los procesos ejecutados por el usuario al no disponer del control del cual disponíamos en casos anteriores, sobre la aplicación o sobre la infraestructura subyacente.

Algunos ejemplos de servicios SaaS más conocidos son Google Apps, Microsoft Online Services o Salesforce.

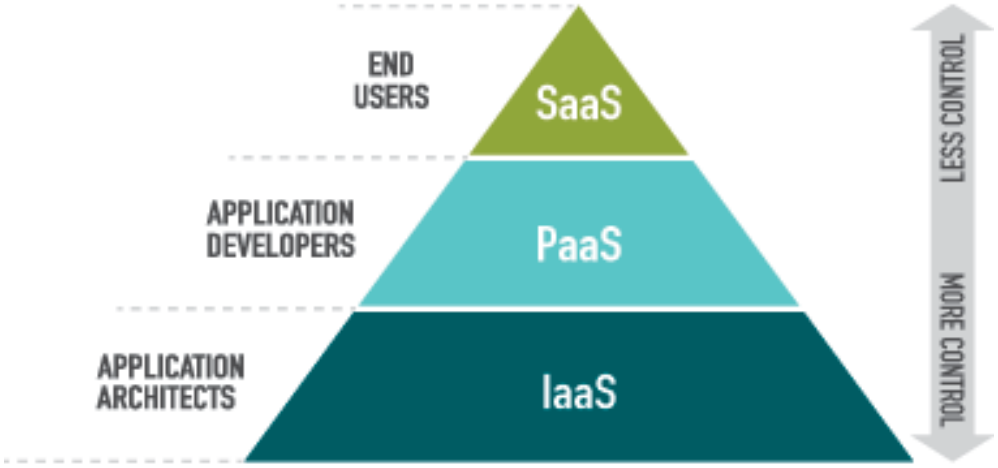


Figura 2: Pirámide de capas de abstracción de la Nube

2.1.4. Tipos de Nubes

Si tenemos en cuenta como son desplegados o implementados los servicios que dan soporte a una Nube, podemos clasificarlas según los siguientes tipos:

- **Nube privada**: tipo de Nube en la cual el consumidor tiene acceso exclusivo virtual o físico a sus servicios de cómputo.
- **Nube pública**: es aquel modelo de Nube en el cual la infraestructura y sus recursos lógicos pertenecientes al entorno se encuentran disponibles para el público en general o para un amplio grupo de usuarios.
- **Nube híbrida**: en este caso se combinan dos o más tipos de Nubes (Pública, Privada o Comunitaria) que se mantienen como entidades separadas pero con un nexo de unión a través de tecnologías estandarizadas o propietarias, que permiten la portabilidad de datos y aplicaciones.

- **Nube comunitaria**: aquí, la infraestructura es compartida por diversas organizaciones y su principal objetivo es soportar a una comunidad específica con un conjunto de preocupaciones u objetivos en común.

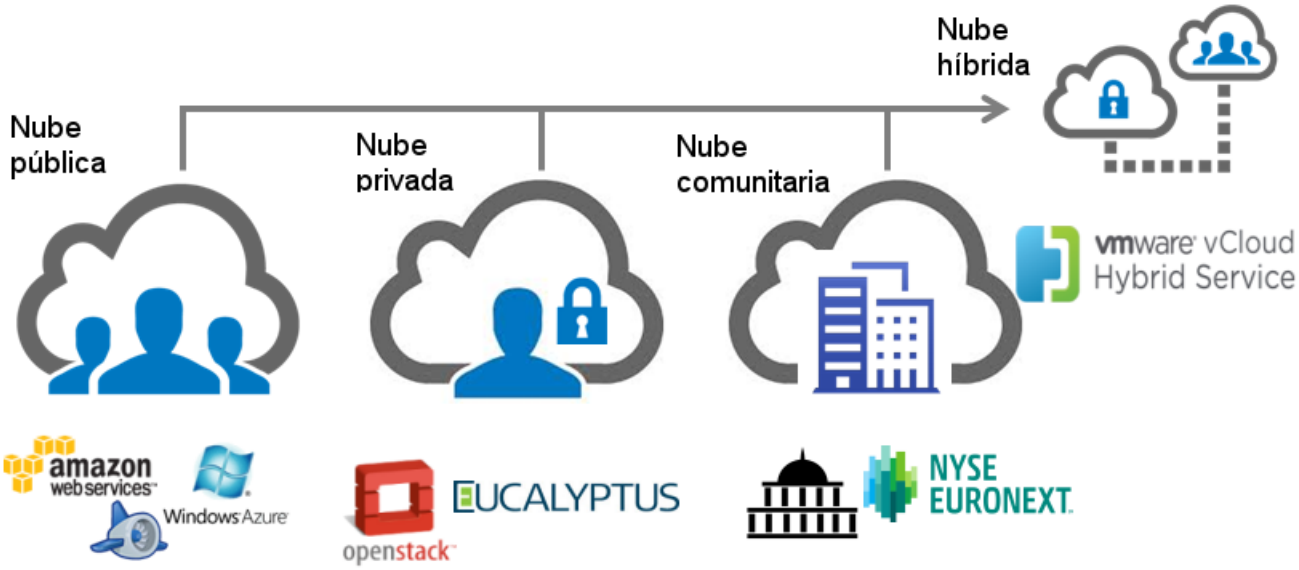


Figura 3: Tipos de Nube y ejemplos

En la Figura 3 podemos observar algunos ejemplos de soluciones o implementaciones de cada tipo de Nube.

Sin duda las Nubes públicas son las que tienen un mayor recorrido y por ello multitud de proveedores ofrecen soluciones de este tipo, no obstante estos mismos proveedores están comenzando a poner al alcance de los usuarios alternativas a sus servicios, en ocasiones como software libre como es el caso de OpenStack [11], a nivel privado, híbrido o incluso comunitario. Este último caso es el de NYSE Technologies [12].

2.2. Desafíos e ideas claves de la Nube

En esta sección se exponen los retos e ideas clave a las que se debe hacer frente en el estado actual de los servicios de computación en la Nube. Estos desafíos, son objeto de los principales trabajos de investigación en la actualidad sobre la materia, y todos ellos guardan una estrecha relación con el objetivo del presente trabajo.

2.2.1. Interoperabilidad entre Nubes: IaaS y PaaS

Generalmente, las capas de abstracción *IaaS* y *PaaS* son las más demandadas por parte de los desarrolladores y empresas desarrolladoras de software. Su consumo crece año tras año gracias a su flexibilidad, simplicidad, y sobretodo por requerir una menor inversión en comparación con la necesaria para la instalación de infraestructuras propias.

Las soluciones de PaaS, resultan especialmente convenientes para el desarrollo y despliegue de aplicaciones usando una determinada tecnología, por ejemplo un lenguaje de programación y framework de desarrollo concreto. La implementación de estos servicios son llevadas a cabo por parte del proveedor, no existiendo un estándar para la creación de los mismos. Este hecho dificulta enormemente la interoperabilidad y aumenta el vendor lock-in, además dependemos del control que se nos otorgue desde la interfaz de gestión para configurar nuestros componentes.

La ausencia de diseños apoyados en estándares, hace que resulte complicado establecer soluciones a corto plazo para facilitar tareas como la de interoperabilidad, gestión dinámica o reconfiguración de servicios, que sean capaces de operar con la gran cantidad de proveedores existentes. Iniciativas como Cloud4SOA [13], han tratado de abordar este problema ofreciendo un servicio que da soporte a una serie de proveedores PaaS, estableciendo mecanismos para la gestión y migración de aplicaciones entre los mismos. No obstante, tareas como la migración entre estos proveedores no han sido posible en todos los casos.

Esta heterogeneidad, pone de manifiesto que resulta necesaria la convergencia de los proveedores de servicio para el desarrollo de servicios basados en estándares comunes para conseguir una mayor y mejor competitividad, que retorne asimismo en un mayor beneficio para los usuarios.

En la capa de IaaS y con un mayor control sobre la infraestructura virtual, como vimos en las Figuras 1 y 2, resulta abordable el desarrollo de mecanismos para proporcionar una solución a la problemática expuesta.

Con estas ideas en mente, las investigaciones sobre *Inter-Clouds* [14], tienen como propósito garantizar calidad de servicio, así como rendimiento y disponibilidad de cada servicio, a través de la interconexión de Nubes de diferentes proveedores. Aquí aparte de la interoperabilidad necesaria, cobran también especial importancia las *SLA*[IV], la *QoS*, y el uso de interfaces estándar.

La entrada en juego del *Inter-Cloud* y su necesidad [15] tienen como argumento sus beneficios tales como: diversificación geográfica de recursos, mayor capacidad de recupe-

[IV] SLA - Service Level Agreement, en castellano acuerdo de nivel de servicio

ración ante desastres, se evita el vendor lock-in y los proveedores son capaces de ofrecer mejores SLA a los usuarios.

Tipos de Inter-Cloud

Dentro de los llamados Inter-Clouds, podemos clasificarlos en las dos siguientes categorías [15]:

- *Clouds o Nubes federadas*: se da cuando un conjunto de proveedores de servicios de Nube, de forma voluntaria, realizan interconexiones entre sus infraestructuras con el objetivo de permitir el mutuo intercambio de recursos. Este enfoque resulta especialmente interesante para organizaciones o entidades gubernamentales.
- *Multi-Cloud*: hace referencia al uso de múltiples e independientes Nubes por parte de un cliente o servivio. A diferencia de la anterior categoría, en este caso un entorno multi-cloud no implica la interconexión entre y compartición de recursos por parte de proveedores. La responsabilidad del manejo, planificación y gestión de los recursos recae sobre el cliente o los servicios que actúen como intermediarios (denominados brokers en la literatura en inglés) del mismo.

Dentro de los Inter-Clouds, el enfoque *multi-cloud* será el que ocupe mayor atención a lo largo del trabajo. En este modelo, se asume que no existe a priori ningún acuerdo entre proveedores de servicios.

Multi-Cloud: iniciativas, herramientas y tecnologías

A continuación se hará un resumen de los proyectos y tecnologías en relación a multi-cloud que han sido objeto de análisis y estudio. La Tablas 1 y 2 no pretende ser un informe exhaustivo de cada una de las herramientas existentes, no obstante, puede servir para ofrecer una visión global de las más relevantes, de sus principales características y de su estado actual.

	Tipo/Organización	Estado	Licencia	Caracterísitcas
Contrail [16]	EU project	terminado	Apache 2.0	Nube federada, SLA, IaaS & PaaS, portabilidad
OPTIMIS [17]	EU project	terminado	BSD license	Nube federada, Nube comunitaria, IDE, SLA
mOSAIC [18]	EU project	terminado	Apache 2.0	Nube federada y multicloud, API, portabilidad, semántica
STRATOS* [19]	York University. Apoyada por fondos de NSERC, Amazon and CA Inc.	terminado	Apache 2.0	PaaS, aplicaciones determinadas, desarrollo y ejecución
OpenTOSCA [20]	Institute of Architecture of Application Systems, Stuttgart	en desarrollo	Apache 2.0	estándar TOSCA, ecosistema, contenedores de aplicaciones
MODAClouds [21]	EU project	en desarrollo	Apache 2.0	IDE, adaptación, QoS, prototipado temprano, Model-driven , semántica
SeaClouds [22]	EU project	en desarrollo	Apache 2.0	orquestación, SLA, monitorización, basado en estándares, reconfiguración
Cloud4SOA	EU project	terminado	GPL	recomendación, semántica, PaaS, discovery
Apache Brooklyn [23]	CloudSoft	en desarrollo	Apache 2.0	CAMP, despliegue, gestión de aplicaciones distribuidas

Tabla 1: Proyectos multi-cloud

*Donado a Apache en 2013

	Tipo	Ult. versión	Lenguaje	licencia	Caracterísitcas
Apache jclouds [24]	librería	1.9.0	java, clojure	Apache 2.0	portabilidad, control absoluto, desarrollo muy activo, gran comunidad, popular, importante rango de proveedores, plantillas
Apache LibCloud [25]	librería	0.17.0	python	Apache 2.0	abstracción, importante rango de proveedores, simplicidad, gran comunidad
Apache DeltaCloud [26]	librería	1.1.3	ruby	Apache 2.0	RESTful API, proveedores más importantes
SimpleCloud* [27]	librería	2.0.1	PHP	BDS License	abstracción, simplicidad, almacenamiento y bases de datos, poca implementación a nivel IaaS
Apache Nuvem [28]	librería	1.0-incubating	java	Apache 2.0	poco activo, solo soporta Google App Engine y Amazon EC2

Tabla 2: Librerías y herramientas multi-cloud

*Incluida en Zend Framework como *Zend_Cloud* [29]

Como se puede observar en la tabla 2, las librerías multi-cloud más populares y con

mayor número de proveedores en su catálogo son Apache jclouds y Apache LibCloud, por ello constituyen un estándar *de facto* para desarrolladores de *java* y *python* respectivamente.

La herramienta Brooklyn, desarrollada por Cloudsoft y actualmente incluida dentro del proyecto *Apache Incubator*[V], hace uso de jclouds y es sin duda una de las más avanzadas y prometedoras del sector. El proyecto *SeaClouds*, en el cual participa la Universidad de Málaga, usa actualmente Brooklyn para el despliegue de aplicaciones.

2.2.2. SLA & QoS

Tal y como vimos en la sección 2.2.1, los conceptos de SLA y QoS resultan clave. Para aprovechar la ventajas ofrecidas por la computación en la Nube es imprescindible poder establecer acuerdos fiables con los proveedores para garantizar el cumplimiento de los requisitos necesarios, y la calidad del servicio.

SLA: Acuerdo a nivel de servicio

El término SLA [30] se define como un contrato entre proveedor y cliente acerca de un servicio, donde se especifica la función que debe realizar éste, sus límites en cuanto a rendimiento, las obligaciones de ambos actores, y cómo deben manejarse las posibles desviaciones respecto de las condiciones acordadas inicialmente.

El SLA contiene en esencia, el conjunto de requisitos no funcionales del servicio que el potencial usuario solicita al proveedor. Este es un área objeto de multitud de trabajos de investigación en la actualidad, y su estandarización, evolución y automatización suponen un reto.

Cada proveedor de servicios de Nube establece e informa a los usuarios de los SLA de sus productos, estos constituyen una garantía y son sin duda un factor muy importante a tener en cuenta. La importancia se debe a lo crítico que puede llegar a ser para las organizaciones una violación de alguno de los requisitos que deben cumplir sus sistemas.

Los contenidos [31] que debería incluir un SLA son:

- el conjunto de servicios ofertados por el proveedor
- una definición especifica de cada uno de los servicios
- las responsabilidades del proveedor y clientes
- el conjunto de métricas usadas para determinar si el proveedor está suministrando los servicios tal y como se prometió

[V] http://incubator.apache.org

- un mecanismo de auditoría
- los mecanismos disponibles para la resolución de conflictos en el caso de que estos términos no se llegaran a cumplir
- las condiciones en las cuales el SLA puede cambiar a lo largo del tiempo

A pesar de que algunos proveedores ofrecen condiciones negociables en función de los requisitos específicos del cliente, en Nubes públicas habitualmente esto no ocurre así, y los clientes pueden no ver satisfechas sus necesidades por completo con los SLA estándar.

Esto se une a que, normalmente, las especificaciones de los SLA se encuentran basadas en plantillas expresadas en lenguaje natural, de carácter ambiguo, por lo que resulta difícil la convergencia entre estas especificaciones y los requisitos, QoS y necesidades de los sistemas de información reales.

Con esta visión en mente, cobra más importancia si cabe el desarrollo de tecnologías inter-cloud que permitan a los proveedores mejorar las condiciones de los SLA (p. ej. a través de Nubes federadas) y a clientes controlar el cumplimiento de los requisitos de sus sistemas, mediante técnicas avanzadas y adaptables de monitorización, y evitar el vendor lock-in con tecnlogías multi-cloud.

QoS: Calidad de servicio

En computación en la Nube, QoS [32] hace referencia a los niveles de rendimiento, fiabilidad y disponibilidad ofrecida por un determinado servicio o infraestructura de un proveedor. Parte de estos parámetros pueden encontrarse especificados en los SLA en forma de niveles mínimos que deben cumplirse.

El criterio de calidad de servicio no es único, y resulta altamente dependiente de la aplicación o servicio desplegado en la Nube y de su entorno. Así, por ejemplo, si un determinado proveedor ofrece unos parámetros de latencia determinados en función de la localización de su centro de datos, esta no será la misma para el acceso de usuarios desde cualquier localización.

De la misma forma, un servicio puede requerir diferentes parámetros de calidad de servicio dependiendo de los cambios en las reglas del negocio, la época del año, o incluso la hora de día.

Resulta evidente que un control de la QoS en tiempo real mediante técnicas de monitorización marcará la diferencia y permitirá reaccionar ante los requisitos cambiantes de los servicios, mediante la reconfiguración de los mismos, de la infraestructura, mediante escalabilidad (con recursos del mismo proveedor, de otro o de manera híbrida), o bien

los asociados a las condiciones del proveedor. Dentro de esta último caso, se encuentran aquellas situaciones asociadas a los cambios en las condiciones del servicio por parte del proveedor, ya sea directamente QoS, tarificación, aparición de nuevos competidores o políticas de seguridad.

Evitar el vendor lock-in y tener poder de decisión para redistribuir, reconfigurar o escalar nuestros servicios, nos conduce a la obtención de una mejor QoS para nuestros servicios.

2.3. Monitorización

La palabra monitorizar se define como:

"observar mediante aparatos especiales el curso de uno o varios parámetros fisiológicos o de otra naturaleza para detectar posibles anomalías".

En *ingeniería del software* [33], usamos el término *monitorización* para referirnos al registro de la ocurrencias de un evento específico durante la ejecución de un programa, con el objetivo de obtener información que no es posible mediante el estudio de su implementación. Esta información incluye el comportamiento del componente en ejecución así como información acerca del sistema operativo sobre el que se ejecuta.

Cuando tratamos con sistemas distribuidos, el concepto de monitorización o supervisión no sólo se asocia al control del estado de cada elemento integrante del sistema, sino también al rendimiento de éste y su cumplimiento de determinados patrones individuales que formarán parte de los requisitos conjuntos del sistema completo. Esta monitorización en tiempo real, debe cumplir el objetivo primordial de mantener el rendimiento del sistema dentro de un rango de valores, de manera que, que no varíe el orden ni el comportamiento de los eventos del sistema.

El nivel de monitorización no siempre es el mismo en todo los sistemas, y de igual forma, tampoco lo es dependiendo del destinatario final de estos resultados. Los expertos más técnicos probablemente desearán conocer la información acerca del funcionamiento a bajo nivel. En cambio, para los profesionales más cercanos al sistema directivo serán de utilidad aquellos parámetros sobre los procesos de alto nivel de su sistema.

Por ello, se debe recopilar la información adecuada en cada caso, representarla con claridad y de manera que aporte valor y soporte la toma de decisiones, ya sea esta, a largo o medio plazo o incluso en tiempo de ejecución.

2.3.1. Monitorización en la Nube

La monitorización continua de la QoS de los servicios desplegados en la Nube resulta clave y necesaria para satisfacer las reglas de negocio y verificar el cumplimiento de los SLA [34, 35]. Expertos coinciden en que será un aspecto aún más relevante en el futuro, con el crecimiento en el uso y en la complejidad de los componentes. Esto convierte la monitorización en un aspecto vital para asegurar el éxito de las compañías.

Algunas de las razones por las cuales se justifica la relevancia de la monitorización en la Nube quedan recogidas en la Tabla 3.

	Descripción
Seguridad	Una de las mayores barreras de cara a la adopción de la Nube. En ciertos entornos e industrias este resulta un factor más crítico aún, que sin duda hay que tener en cuenta.
Gestión de SLA	Verificar el cumplimiento práctico de los SLA es una tarea que requiere monitorización continua, y exige medios avanzados para su tratamiento.
Plan de aprovisionamiento	Identificar que recursos se están utilizando en la Nube. El rendimiento y capacidad como requisitos pueden evaluarse y adaptarlos para alcanzar el nivel adecuado y satisfacer a los clientes.
Manejo de recursos	De igual forma la gestión de nuestros recursos nos ayuda a maximizar su uso para aprovecharlos al máximo, minimizando los costes de la infraestructura completa.
Resolución de problemas	La diagnosis e identificación de qué componentes están fallando o dando problemas es vital. En una arquitectura compleja, las herramientas de monitorización ayudan en la diagnosis y a rectificar con mayor rapidez
Rendimiento	El seguimiento del rendimiento de los recursos controlados por nuestra aplicación o servicios nos ayuda a mantener su disponibilidad y funcionalidad óptimas.
Facturación	Un requisito básico es conocer nuestra facturación por uso de recursos y saber como sacar el mayor provecho a los servicios del proveedor.

Tabla 3: Importancia de la monitorización en la Nube

En el entorno de la Nube delegamos el control en la entidad proveedora de servicios y, al igual que ocurría con los SLA, los proveedores suelen ofrecer algunos mecanismos para la medición de métricas estándar. Este conjunto de métricas son fijas, y en la mayoría de ocasiones no serán suficientes para verificar el cumplimiento de los SLA, QoS, y en definitiva los requisitos no funcionales de nuestros sistemas.

Una monitorización efectiva, debe servir de soporte para la toma de decisiones. Detectar el incumplimiento de alguno de los requisitos de sistema y poder reconfigurar éste de forma dinámica, va a depender de que seamos capaces de obtener datos relevantes a través de métricas que nos aporten valor. Estas métricas, deben ser flexibles y configurables de forma dinámica, es decir, el sistema de monitorización debe permitir en tiempo real añadir nuevas métricas, ya que, los requisitos un servicio también varían en tiempo de ejecución.

Las herramientas de monitorización existentes [36], como por ejemplo las utilizadas en *Computación Grid*[VI], han tratado de adaptarse a este nuevo paradigma de computación en la Nube. Sin embargo, estas herramientas, normalmente se encuentran restringidas a la localización y homogeneidad de los objetos monitorizados, y no son capaces de adaptarse y aplicarse a este nuevo entorno.

Este tipo de tecnologías por tanto, se encuentran ancladas a su objetivo desde la etapa de diseño de las mismas [34]. No obstante, como se ha visto, debido a la naturaleza dinámica de la Nube no podemos asumir las herramientas o métricas necesarias a priori en fase de diseño de una aplicación. Más si cabe, sin conocer en qué infraestructura será desplegado ni cual será su comportamiento futuro.

Aunque el uso de herramientas ofrecidas por los proveedores pueden ayudar en determinados casos a soportar la toma de decisiones acerca de la reconfiguración de nuestro servicio en dicho proveedor, estas no resultan eficaces a largo plazo. Para nuestro objetivo, en un entorno multi-cloud, con entornos de diferentes proveedores, debemos disponer de mecanismos flexibles y adaptables a las necesidades requeridas en un entorno tan heterogéneo.

2.3.2. Herramientas

A lo largo del desarrollo del trabajo, una serie de herramientas de monitorización han sido analizadas y estudiadas. Las propiedades más relevantes de cada una de ellas se presentan en la Tablas 4 y 5.

[VI] Computación grid - https://en.wikipedia.org/wiki/Grid_computing

En la Tabla 4, se analizan las herramientas de monitorización que ofrece cada proveedor para la supervisión de sus propios servicios. La Tabla 5 está dedicada a servicios de monitorización genéricos. En ambas, la columna "Características", muestra una serie de comentarios o palabras que hacen referencia a su funcionalidad, soporte o elementos destacables.

Cuando se habla de escalabilidad, se hace referencia a la capacidad de adaptación de una herramienta a un nuevo entorno en crecimiento, ya sea a partir de la detección de nuevas instancias o máquinas, o bien mostrando la información de un sistema agrupada como un único elemento abstrayéndose del número de nodos de la infraestructura. La propiedad para ser extensible, indica que se proporcionan interfaces para definir y añadir nuevos elementos personalizados (en mayor o menor grado) a la herramienta. El soporte para de alarmas, también es algo común a muchas herramientas de este tipo, la definición de alarmas permite la detección de determinados umbrales o patrones definidos previamente por el usuario. En cuanto al modelo *freemium*, es aquel modelo de negocio consistente en ofertar servicios con ciertos límites de forma gratuita, que pueden complementarse o ampliarse con el pago de una cuota.

	Proveedor	Soporte	licencia	Características
CloudWatch [37]	Amazon	AWS	propietaria	pocas métricas, pago por uso, extensible, API, alarmas
AzureWatch [38]	Paralelap	Azure	propietaria	escalable, pago por uso, extensible, autónoma, adaptable
CloudKick [39]	Rackspace	Amazon EC2, GoGrid, Linode, Rackspace, Joyent, OpSource and SoftLayer	propietaria	escalable, alarmas, adaptable, variedad de métricas, incluido con el registro
Google Cloud Monitoring [40]	Google	Google Cloud Platform	propietaria	versión beta, simple, pocos servicios, pocas métricas, gratuito en beta
HP Cloud Monitoring [41]	HP	HP Cloud	propietaria	versión beta, API, alarmas, notificaciones, gratuito en beta

Tabla 4: Servicios de monitorización de Nube ofrecidos por los proveedores

	Tipo/Organización	Soporte*	licencia	Caracterísitcas
Zenoss [42]	Zenoss Inc.	AWS, CloudStack, OpenStack and vCloud/vSphere, Azure	GPL 2/propietaria	extendido a la Nube, plugins, configurable, eventos, análisis, modelo freemium, sencillez
Hyperic [43]	VMware	AWS, VMware	Apache 2.0/propietaria	extendido a la Nube, auto-discovery, identificación de problemas, análisis, control de acceso, modelo freemium
Ganglia [44]	proyecto comunitario	-	BSD	orientado a clusters y computación grid, histórico, usa XML, escalable, distribuido, gratuito
Nagios [45]	Nagios Enterprises	AWS, OpenStack	GPL 2/propietaria	extendido a la Nube, basado en extensiones, diagnosis, modelo freemium
collectl [46]	proyecto comunitario	-	GPL 2	configurable, diferentes formatos, soporte de sockets, basado en subsistemas
collectd [47]	proyecto comunitario	-	GPL 2	diseño modular, plugins, notificaciones, diferentes formatos
OpenNMS [48]	OpenNMS Group	-	Affero 3	escalable, monitorización de redes, sencillez, configurable, alertas, integración modelo de soporte
Zabbix [49]	Zabbix SIA.	-	GPL	basado en módulos, diferentes herramientas de visualización, alertas, flexible, altamente escalable

Tabla 5: Servicios y herramientas de monitorización

*Se hace referencia al soporte específico de proveedores determinados (conectores), en aquellas herramientas en las que este factor no es relevante no se muestra información.

3. Desarrollo de la Infraestructura

3.1. Estudio del sistema

A lo largo de la sección 2 se ha tratado, en primer lugar, de poner en contexto al lector sobre los conceptos clave de computación en la Nube. Posteriormente se ha profundizado sobre el trabajo de investigación y análisis llevado a cabo para la consecución de este proyecto. Sin el conocimiento adquirido a partir de la investigación de diferentes fuentes y tecnologías resultaría imposible llevar a cabo el diseño e implementación de la infraestructura fruto del trabajo.

3.1.1. Alcance

Tal y como se introdujo en la sección 1.2 el objetivo del trabajo es el desarrollo de una infraestructura para la monitorización de servicios desplegados en la Nube a nivel de plataforma. Esta monitorización debe realizarse de tal forma que nos permita conocer el "estado del servicio".

Dentro de este objetivo principal, deben considerarse otra serie de objetivos derivados de las condiciones del entorno de realización del trabajo:

- nuestro sistema debe operar en un ecosistema multi-cloud
- la monitorización debe llevarse a cabo usando mecanismos reutilizables y adaptables a la plataforma
- los datos deben presentarse de forma coherente y adecuada al tipo de usuario
- estos datos deben ser consistentes y aportar valor añadido
- la recogida de información de monitorización debe ser susceptible de análisis y ayudar a la toma de decisiones
- estas decisiones incluirán el cambio en la topología de las aplicaciones desplegadas y la reconfiguración de estos servicios
- las condiciones del entorno requerirán el uso de componentes flexibles

A la finalización del trabajo, este conjunto de objetivos deben ser cubiertos por la infraestructura.

3.1.2. Situación actual

El estudio del *Estado del arte* ha permitido conocer y entender el estado actual de las investigaciones sobre computación en la Nube. Esta investigación y recopilación de información se ha enfocado con una estrategia *top-down*, es decir, introduciendo en primer lugar aquellos conceptos más genéricos referentes a la Nube, para poco a poco acercarnos a los conceptos más específicos sobre entornos multi-cloud, interoperabilidad y monitorización.

Comprender los avances, opiniones y enfoques de las diversas fuentes y expertos en la materia, hace posible valorar y abordar las necesidades y requisitos que deben satisfacerse.

3.1.3. Estudio y valoración de alternativas

El conjunto de Tablas 1, 2, 4 y 5, resume el estudio realizado de las diferentes iniciativas, proyectos y herramientas relacionadas con el multi-cloud y la monitorización en la Nube.

Tras la interpretación de los resultados, se debe valorar que herramientas resultan más adecuadas a nuestras necesidades. La elección pasa por seleccionar aquellos componentes que conduzcan a lograr el alcance de nuestros objetivos.

A la finalización del estudio la herramienta Brooklyn [23] ha sido elegida como el componente encargado de la gestión y despliegue de las aplicaciones.

Brooklyn es una herramienta definida como un framework [I] o librería para la gestión y despliegue de aplicaciones distribuidas. Para el despliegue de aplicaciones en la Nube hace uso de Apache jclouds [24], por lo que nos permite acceder al amplio catálogo de servicios de proveedores IaaS. Se trata de una herramienta con una implementación robusta y basada en la figura de las *entidades*. Estas entidades son módulos o componentes cuya interconexión forma una aplicación completa. Su carácter multiplataforma, al estar escrita en el lenguaje *java*, y su licencia han contribuido a que sea una de las herramientas más populares y de mayor crecimiento en comparación con el resto de opciones disponibles.

De cara a la monitorización, *collectl* [46] es una herramienta madura de monitorización que es capaz de recopilar información de diferentes tipos de recursos del sistema. A la vez resulta ligera, eficiente y configurable convenientemente tanto en formato de representación, como en el nivel de detalle. Su eficiencia también es un factor que debe tenerse en cuenta, ya que la monitorización de los servicios debe realizarse sin sobrecargar los recursos del sistema.

[I] Framework - https://en.wikipedia.org/wiki/Software_framework

3.2. Análisis del sistema

Después del estudio de los elementos necesarios para dar una solución al problema, ahora se debe especificar correctamente cada uno de los elementos que van a servir de base para el diseño del sistema.

Mediante el *Análisis del sistema*, se especificará de manera más detallada la infraestructura que vamos a desarrollar, sus objetivos y los requisitos que debe satisfacer.

3.2.1. Objetivos

En la sección de 3.1.1 se encuentran descritos en alto nivel los objetivos que debe cumplir la infraestructura a desarrollar. Estos objetivos, sin embargo, deben dividirse y especificarse de manera más formal y adecuada a la posterior fase de diseño.

En la Tabla 6 se clasifica y resume el conjunto de objetivos que se desea cumplir.

Nombre	Objetivo
Despliegue de aplicaciones distribuidas en la Nube	El sistema debe proporcionar un mecanismo para la descripción y despliegue de los servicios y sus componentes en la Nube.
Despliegue de sensores en la Nube	El sistema debe proporcionar un mecanismo para la descripción y despliegue de mecanismos sensores para recopilar datos de monitorización.
Recopilación de datos	El sistema debe establecer mecanismos para la obtención de los datos de sensores y de las aplicaciones desplegadas en la Nube.
Presentación de datos e información	Los datos recopilados y la información sobre los servicios, deben presentarse de manera adecuada mediante interfaces gráficas intuitivas y que permitan realizar análisis
Almacenamiento de datos	Los datos recopilados deben poder almacenarse pertinentemente y de forma robusta y consistente en una base de datos
Consulta de datos	Los datos recopilados, deben encontrarse accesibles a posteri, para su consulta y análisis
Toma de decisiones y reconfiguración	Los componentes desplegados en la Nube, deben poder reconfigurarse y cambiar su comportamiento en base a las decisiones adoptadas por el usuario.
Control y acceso de usuarios	Los usuarios deben poder acceder a sus configuraciones y servicios de forma independiente y poder configurar su perfil.

Tabla 6: Descripción de componentes y objetivos

3.2.2. Requisitos del sistema

Tras la definición de los objetivos a cumplir, los requisitos marcarán aquellas condiciones o funcionalidades que harán que se cumplan estos objetivos.
A continuación en la Tabla 7 se muestra el catálogo de requisitos definido.

Crear usuario	Crear un nuevo usuario y su almacenamiento en base de datos
Eliminar usuario	Eliminar un usuario de la base de datos
Inicio de sesión	Inicio de sesión del usuario en el sistema
Especificar aplicación	Definir una aplicación para su posterior despliegue, implicando esto la definición de sus características
Crear sensor	Creación y almacenamiento de un sensor para su posterior despliegue, implicando esto la definición de su localización y características
Eliminar sensor	Eliminar un determinado sensor de base de datos
Desplegar aplicación	Desplegar una determinada aplicación ya definida a través de su especificación
Desplegar sensor	Desplegar un determinado sensor ya definido a través de su especificación
Consulta servicios	Consultar el estado de los servicios desplegados
Modificar perfil	Modificar parámetros del perfil del usuario
Monitorizar aplicación	Obtener datos de los sensores que monitorizan la aplicación
Reconfigurar aplicación	Reconfigurar o cambiar el estado de una aplicación desplegada
Consultar histórico	Consultar datos del histórico de monitorización de aplicaciones almacenado en base de datos
Consultar sensor	Consultar la especificación de los sensores definidos en base de datos
Cierre de sesión	Terminar la sesión del usuario en el sistema

Tabla 7: Descripción de los requisitos del sistema

3.2.3. Casos de uso

En la especificación de requisitos de la Tabla 7, se han identificado los requisitos que debe cumplir nuestro sistema. A continuación se ilustran los *casos de uso*, estos deben especificar y ayudar aún más a la comprensión de los requisitos y la relación entre el sistema y su entorno.

Estos casos de uso, deben entenderse como casos de alto nivel. Esto quiere decir que a pesar de que tratan de especificar los requisitos con más detalle, no constituyen una especificación completa de la funcionalidad deseada. A medida que se avance hacia las fases de diseño, esta funcionalidad será descrita y especificada con mayor detalle de cara a la implementación.

En un caso de uso, se entiende por actor a todo aquel usuario o entidad que accede al sistema para satisfacer sus necesidades a través del servicio que se le ofrece. Los actores se dividen según su rol, pudiendo pertenecer un actor a varios roles distintos.

Los actores que van a interactuar con nuestro sistema serán de los dos siguientes tipos:

- **Usuario**: el usuario, puede realizar todas las acciones sobre los sensores, el despliegue, la monitorización y configuración de servicios. De igual forma podrá cambiar su usuario y contraseña si lo desea. Sin embargo, no podrá realizar consultas a la base de datos sobre objetos que no le pertenezcan o no hayan sido creados por él. Los objetos pertenecientes al sistema si serán accesibles, pero no podrá realizar operaciones de borrado sobre ellos ni tampoco sobre su usuario ni el resto de usuarios del sistema.
- **Administrador**: el administrador posee aparte de las funcionalidades atribuidas al usuario normal, la capacidad para la crear y borrar usuarios.

Casos de uso:

- **CU 01.**(véase Figura 4): describe las funciones encaminadas a la autenticación del usuario en el sistema, así como la modificación del perfil del mismo. El inicio de sesión en el sistema por parte del usuario será un paso necesario para la realización del resto de funciones, por lo que el resto de casos de uso deberán incluir el de inicio de sesión. Este comportamiento no se ha ilustrado mediante inclusiones para mayor simplicidad y no cargar de complejidad los diagramas. El usuario accederá al

sistema mediante sus credenciales haciendo uso del nombre de usuario y contraseña, este nombre de usuario y contraseña es creado por el *Administrador*, que es el usuario con rol suficiente para la creación y borrado de usuarios. El Administrador, podrá seleccionar cualquier usuario de la lista de usuarios y borrarlo y asimismo dispondrá de un formulario para la creación de ellos.

- **CU 02.**(véase Figura 5): especifica las distintas operaciones sobre sensores, es decir, la creación, consulta y el borrado de los mismos. La creación de sensores es realizable por cualquier tipo de usuario a través de un formulario en el cual se escriben las características del mismo. Estos sensores son objetos pertenecientes a su creador, salvo los sensores que serán objetos de sistema y serán accesibles por cualquier usuario. La consulta de sensores para cada usuario devolverá una lista con los sensores creados por él mismo, pudiendo realizar las operaciones de consulta y borrado para cada uno de ellos.

- **CU 03**.(véase Figura 6): en este caso de uso se determinan las funciones de despliegue de aplicaciones y sensores. La extensión nos indica que la operación de desplegado de sensores se puede realizar conjuntamente al desplegar la propia aplicación. Las operaciones de selección de sensor y especificación de la aplicación aparecen con el estereotipo de inclusión para reflejar que son necesarios para el cumplimiento del objetivo en cada caso. El despliegue de aplicaciones se realiza a través de un formulario encaminado a tal fin, mediante la especificación de una aplicación podemos a continuación seleccionar un sensor, de los accesibles a través de la operación consulta, para su despliegue. Estas operaciones se pueden realizar por tanto de forma conjunta como se indica en el diagrama, o puede indicarse el despliegue de sensores de forma independiente.

- **CU 04.**: la consulta de servicios por parte del usuario no implica ningún caso de inclusión ni extensión. A través de esta operación consulta, el usuario obtendrá una lista con las aplicaciones que se encuentran desplegadas incluyéndose el estado actual de la misma.

- **CU 05.**(véase Figura 7): las operaciones que realizamos sobre las aplicaciones ya desplegadas vienen indicadas en este caso de uso, y son las correspondientes a las funciones de monitorización y reconfiguración. La monitorización es junto al despliegue de aplicaciones y servicios uno de los casos de uso más relevantes. Para la monitorización, el usuario selecciona del menú de aplicaciones disponibles aquellos elementos que desea monitorizar y a continuación se reciben los datos pertinentes. La reconfiguración es también realizada mediante la selección en este menú de aquellas acciones que se encuentran disponibles para ejecutarse para cada aplicación.

- **CU 06.**: en el caso de uso de consulta de histórico solo disponemos de la operación de consulta por parte del usuario sin ningún caso de inclusión o extensión. Mediante un formulario, el usuario puede especificar filtros sobre los campos de búsqueda para la

obtención de las entradas disponibles que cumplan dichos patrones. Estos patrones, proporcionan mecanismos para determinar características como su rango, fecha o nombre. El conjunto de valores obtenidos serán únicos para cada usuario ya que serán datos obtenidos por un usuario concreto. De esta forma, tras la búsqueda el usuario obtendrá aquellos objetos pertenecientes únicamente a su usuario y no los datos recopilados por cualquier otro.

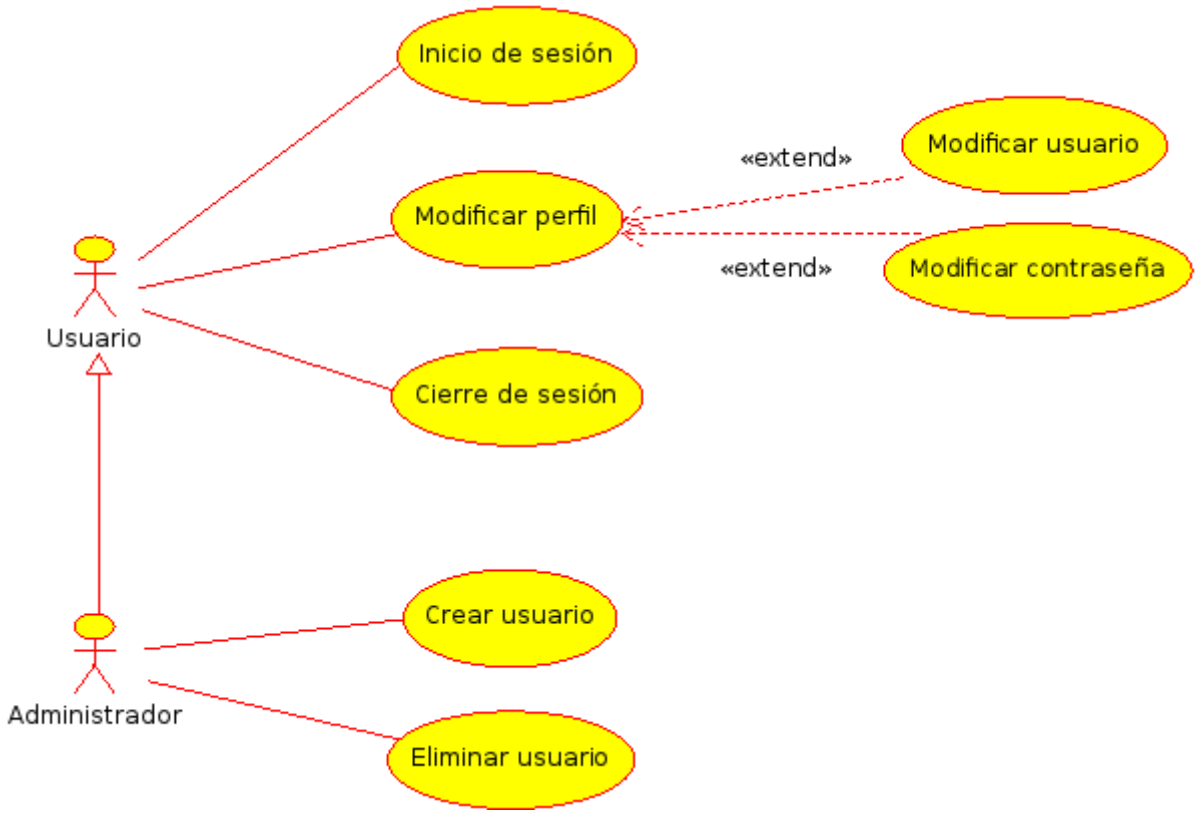


Figura 4: CU 01. Casos de uso de acceso al sistema y modificar perfil, y gestión de usuarios

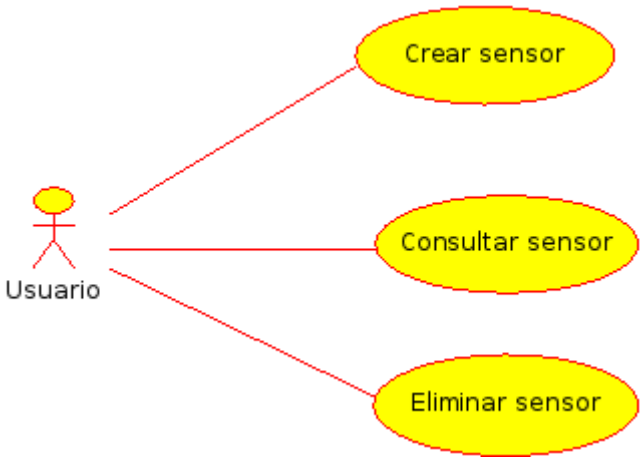


Figura 5: CU 02. Casos de uso de creación, consulta y borrado de sensor

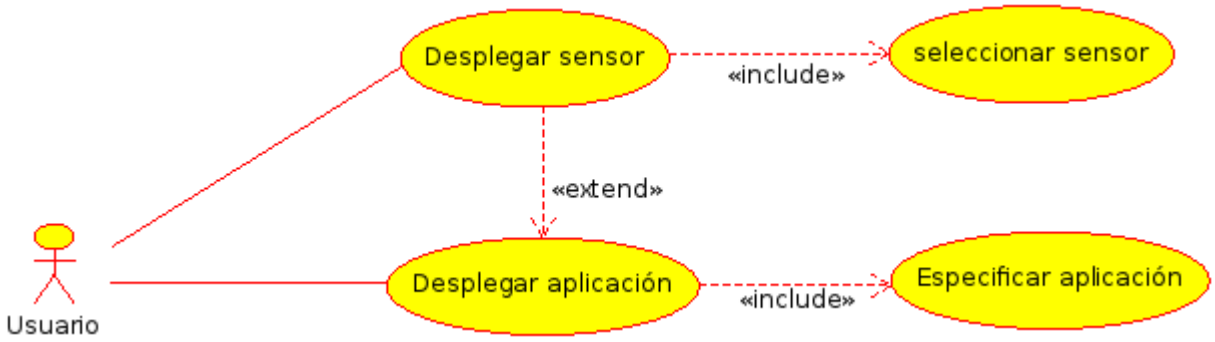


Figura 6: CU 03. Casos de uso de despliegue de aplicaciones y sensores

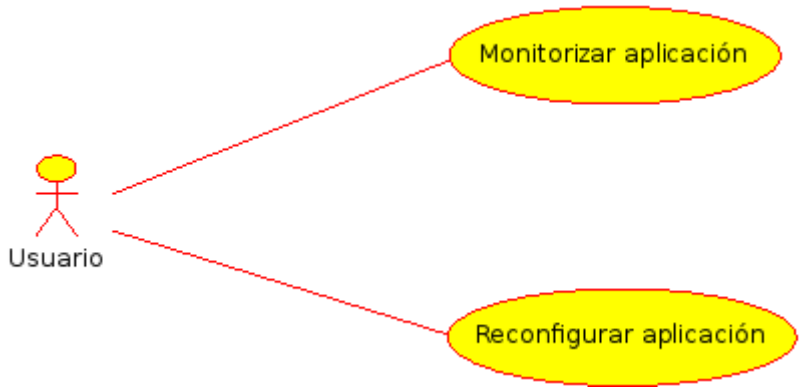


Figura 7: CU 05. Casos de uso de monitorización y reconfiguración de aplicaciones

3.2.4. Identificación de subsistemas

A partir de la descripción de los casos de uso en la sección anterior, podemos empezar a identificar algunos subsistemas que comparten procesos de la misma categoría. La separación de estos subconjuntos ayuda a comprender las diferentes funciones y sus relaciones con el sistema completo. Estos subsistemas se pueden resumir en los siguientes:

- Subsistema de control de acceso
- Subsistema de control de perfil
- Subsistema de control de usuario
- Subsistema de despliegue
- Subsistema de consulta de servicios o aplicaciones
- Subsistema de sensores
- Subsistema de histórico
- Subsistema de monitorización y reconfiguración
- Subsistema de utilidades y servicios

En la Figura 8 se representa un diagrama de los subsistemas involucrados en los diferentes procesos, así como las relaciones de los mismos.

La infraestructura a desarrollar está basada en el patrón modelo vista controlador (MVC). Este patrón de arquitectura establece una separación entre los datos y la lógica de negocio de una aplicación, y su interfaz de usuario y el control de eventos y comunicaciones. Este modelo y la abstracción de cada una de las capas, permite la reutilización de componentes y a la vez facilita la tarea de desarrollo y mantenimiento de aplicaciones y servicios.

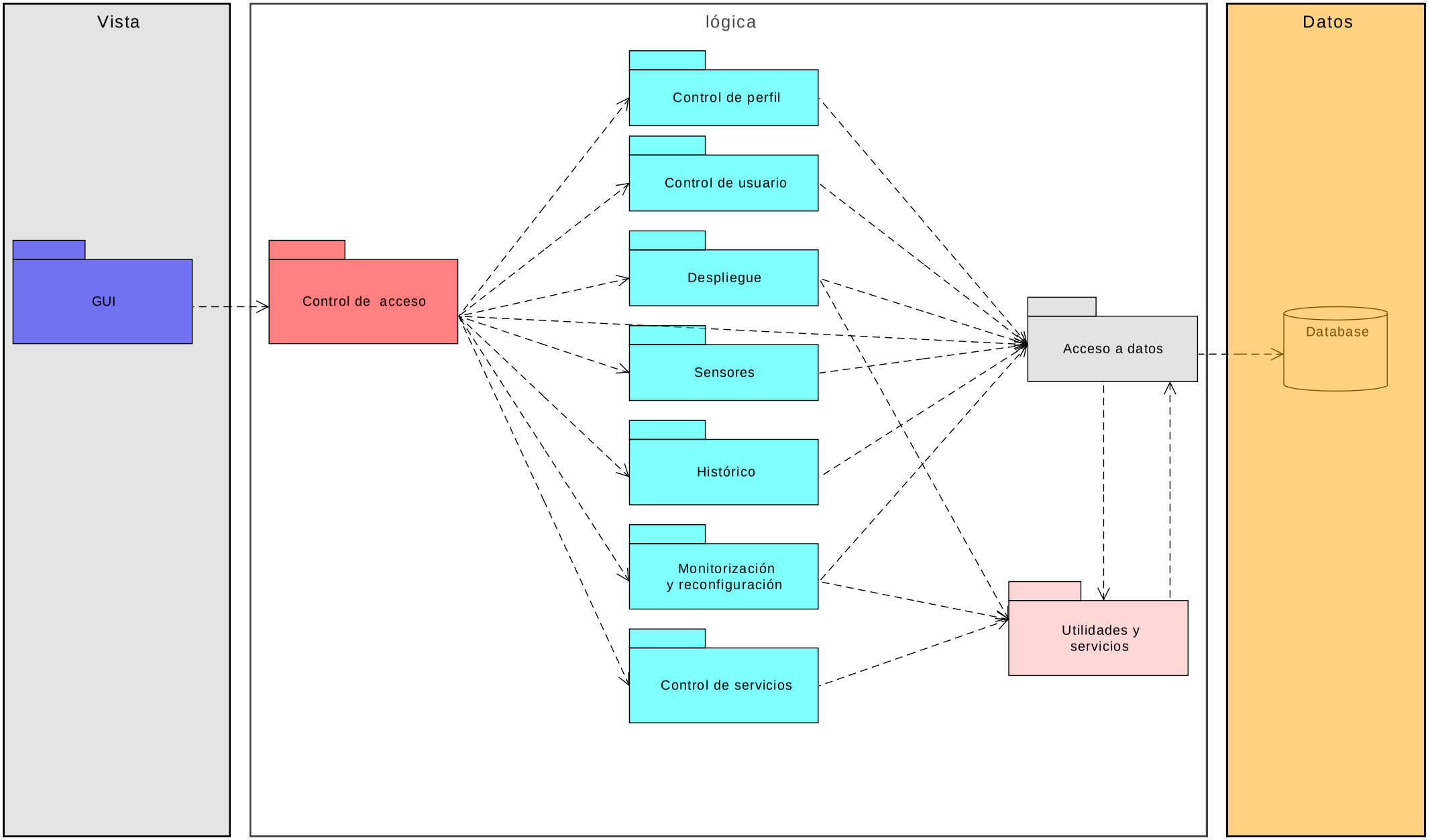


Figura 8: Diagrama de subsistemas

3.3. Diseño y especificación del sistema

El objetivo del proceso de diseño es la definición de la arquitectura del sistema y de su entorno, junto con una descripción detallada de cada uno de los componentes.
A partir de estas especificaciones, será posible llevar a cabo la tarea de implementación final de la infraestructura.

3.3.1. Arquitectura del sistema

Mediante la definición de la arquitectura general del sistema, especificamos las particiones físicas del mismo, la descomposición de sus subsistemas de diseño y la ubicación de cada uno de ellos. En esta fase de modelado, se deben incluir detalles de la infraestructura tecnológica que dará soporte al sistema. Para ello, se ampliarán los detalles acerca de la implementación concreta de cada uno de los nodos involucrados en el sistema.

Para el modelo de sistema que identificamos anteriormente en la sección 3.2.4, debemos asignar aquellos recursos hardware, software y de comunicaciones que soporten la carga técnica y las especificaciones del sistema para su implementación.

La infraestructura diseñada va a estar compuesta principalmente por los siguientes elementos:

- Servidor web
- API de acceso al servidor web
- Base de datos
- Servidor Brooklyn

La tecnología de desarrollo de cada uno de estos componentes debe elegirse de acuerdo a las necesidades del entorno que se detallaron en la fase de estudio. Hoy en día, el patrón MVC es implementado por muchos frameworks webs de diferentes lenguajes de programación. Estos frameworks se han popularizado enormemente en los últimos tiempos en el desarrollo de aplicaciones web. Para la valoración del framework y lenguaje más adecuado se han tenido en cuenta aquellos más extendidos y popularizados, primando como características más relevantes la eficiencia, rendimiento y la capacidad para comunicarse con los componentes externos.

De igual forma, el tipo de base de datos escogida para dar soporte a los datos recopilados por la infraestructura se ha seleccionado de entre las disponibles por su capacidad para ofrecer un rendimiento y capacidad acorde a los requisitos del sistema.

La Figura 9 pretende ilustrar los nodos y componentes involucrados en el sistema así como las distintas tecnologías necesarias para su funcionamiento. La plataforma de monitorización constituye el núcleo del sistema y su implementación debe realizarse proveyendo mecanismos para comunicarse con el resto de componentes que forman la infraestructura.

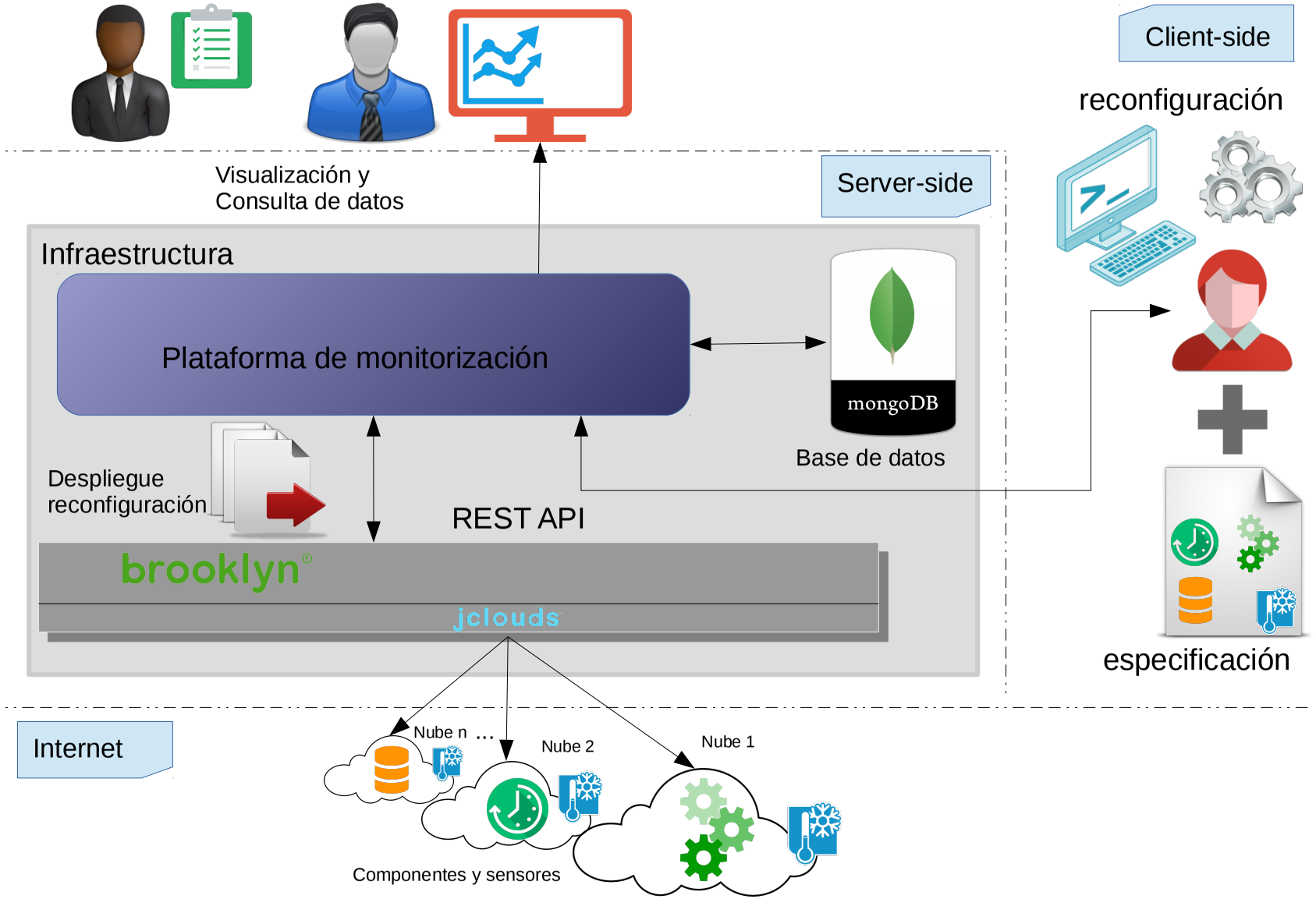


Figura 9: Arquitectura del sistema

3.3.2. Diseño de la realización de los casos de uso

El objetivo de esta fase es definir cómo interactúan entre sí los objetos que cooperan en la realización de un determinado caso de uso o escenario. Este tipo de diagramas, llamados diagramas de secuencia, ayudan a describir el comportamiento dinámico del sistema y el paso de mensajes entre los diferentes componentes.

En las Figuras 10 a 23, se ilustran los diagramas de secuencia correspondientes a los casos de uso:

- Inicio de sesión
- Modificar usuario
- Modificar contraseña
- Crear usuario
- Borrar usuario
- Crear sensor
- Consultar sensor
- Eliminar sensor
- Desplegar aplicación
- Desplegar sensor
- Consultar servicios
- Monitorizar aplicación
- Reconfigurar aplicación
- Consultar histórico

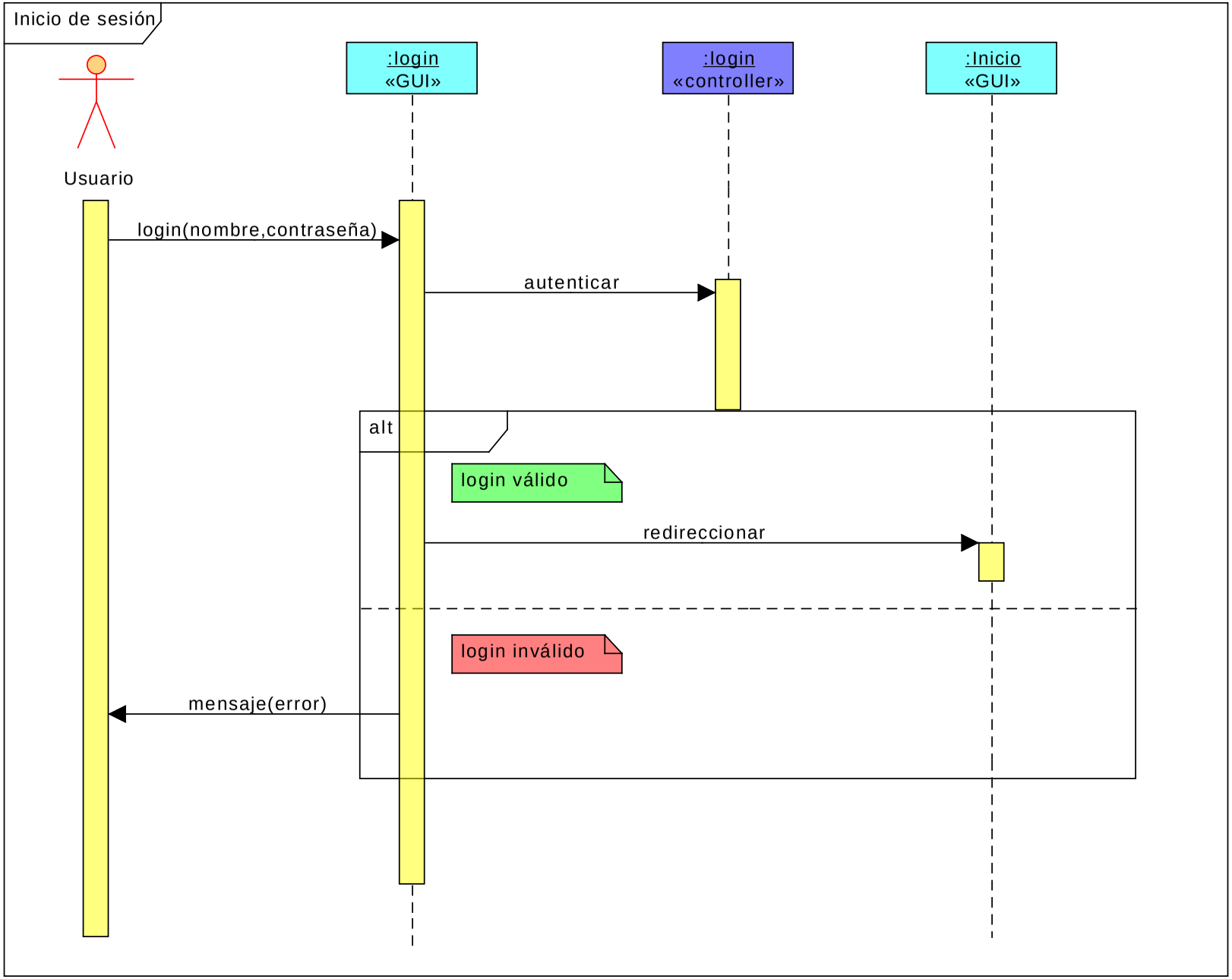


Figura 10: Diagrama de secuencia inicio de sesión

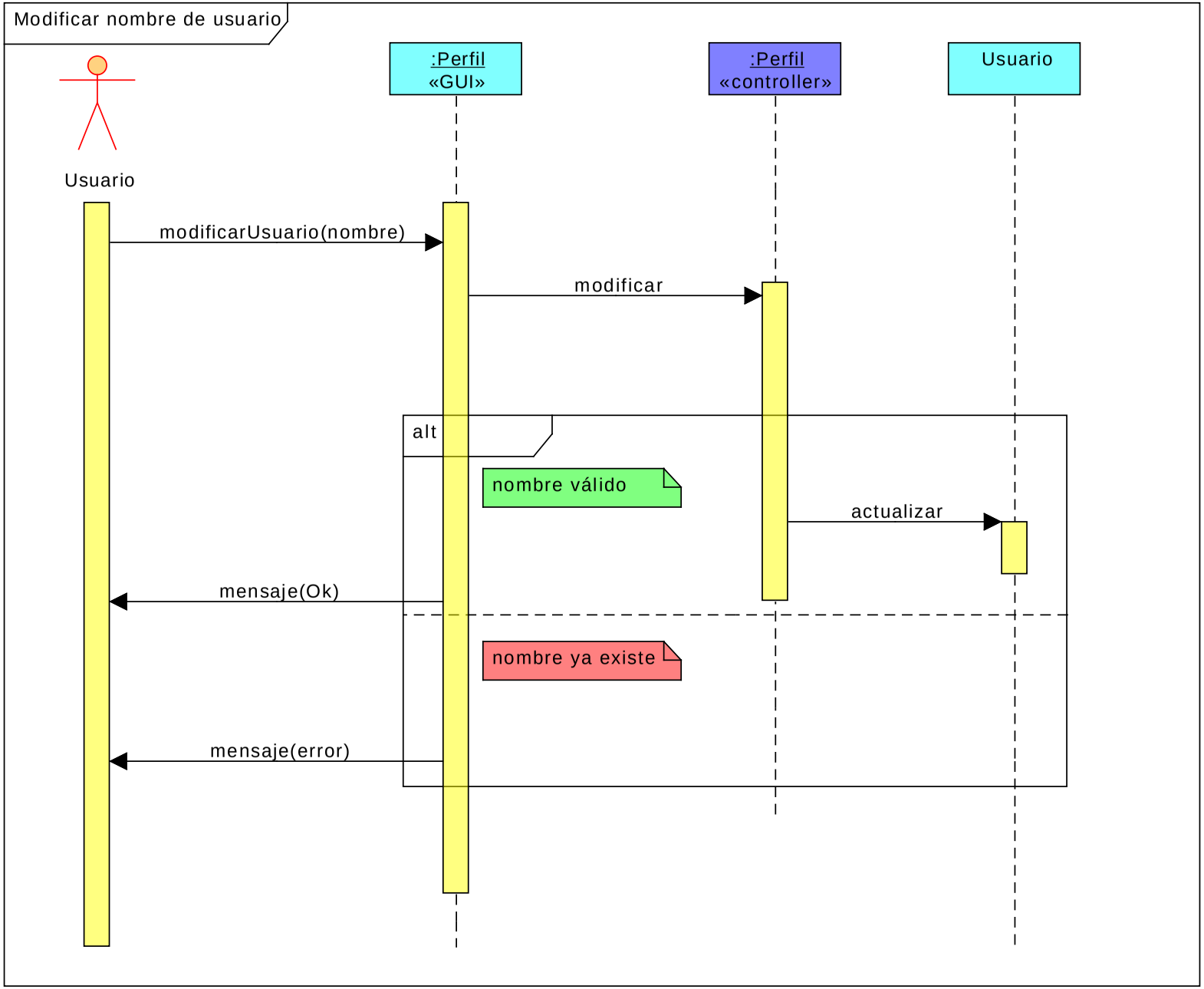


Figura 11: Diagrama de secuencia modificar usuario

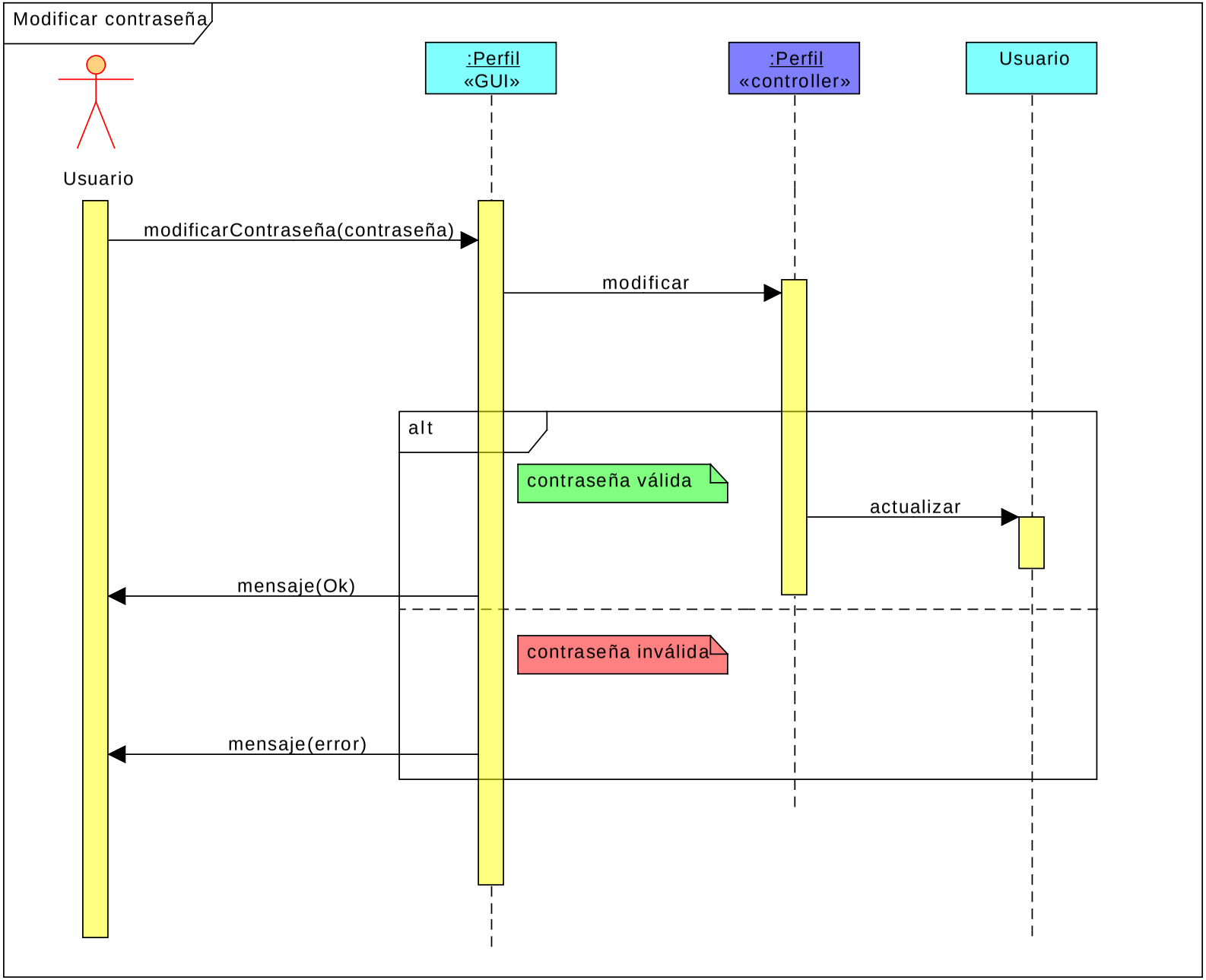


Figura 12: Diagrama de secuencia modificar contraseña

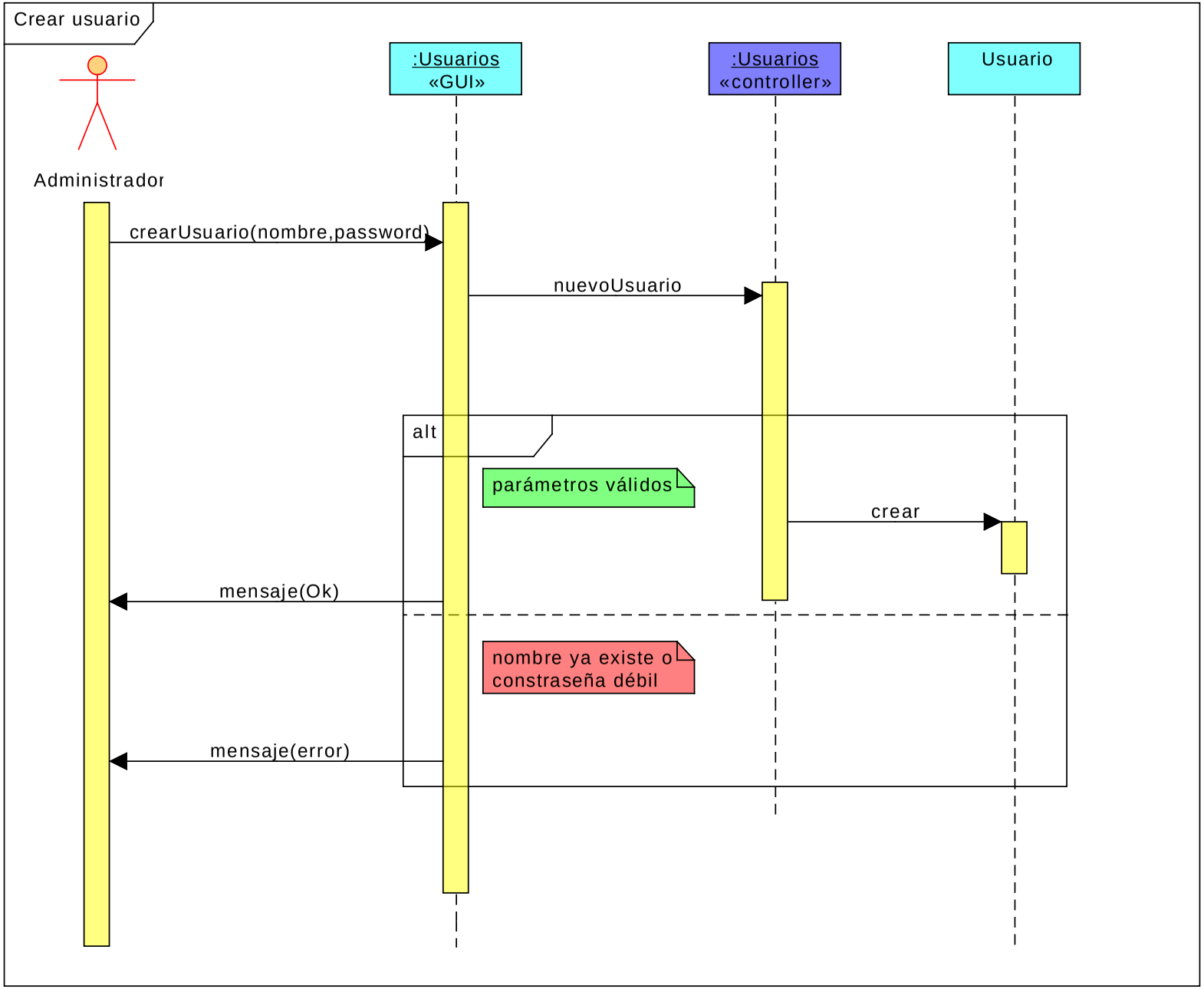


Figura 13: Diagrama de secuencia crear usuario

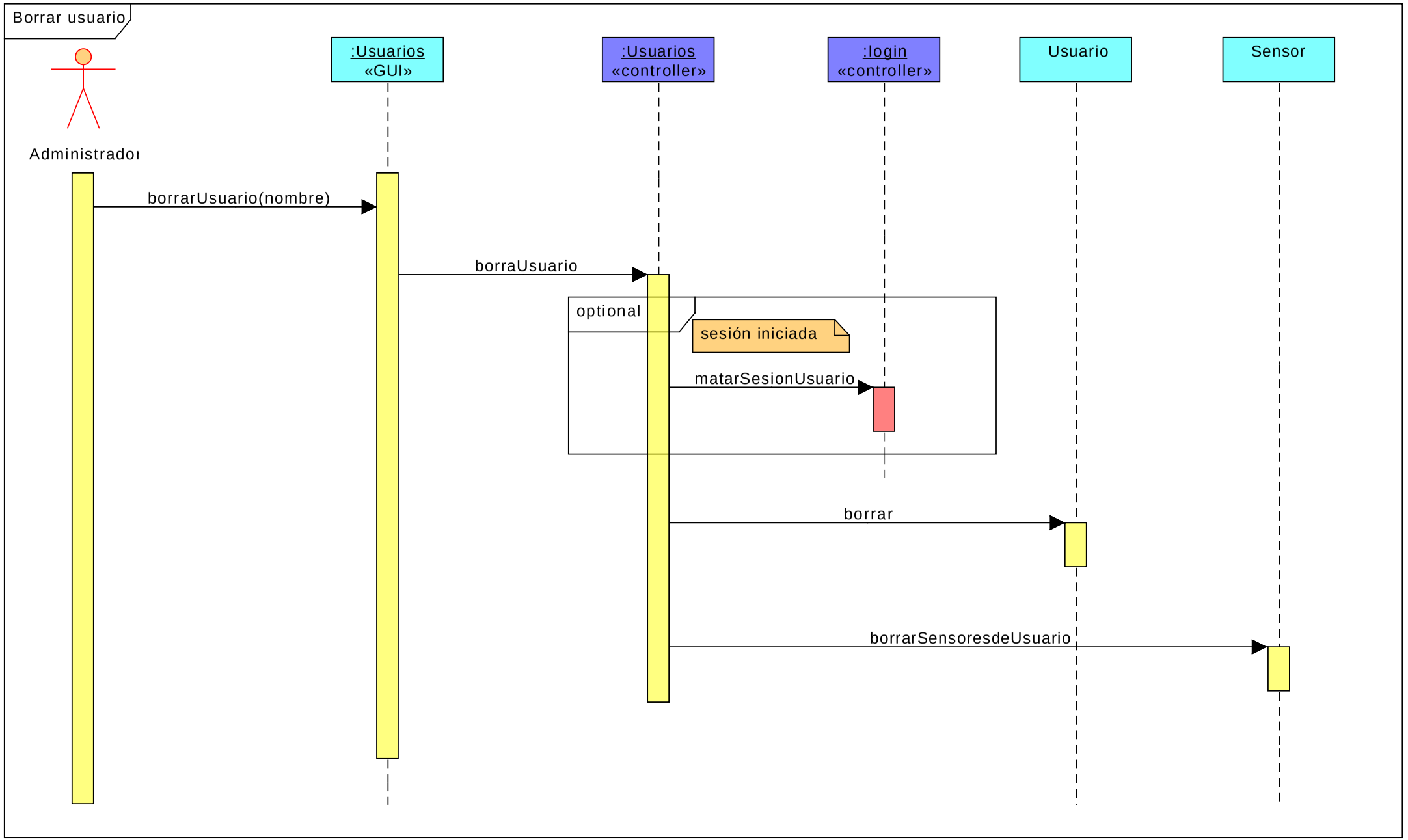


Figura 14: Diagrama de secuencia borrar usuario

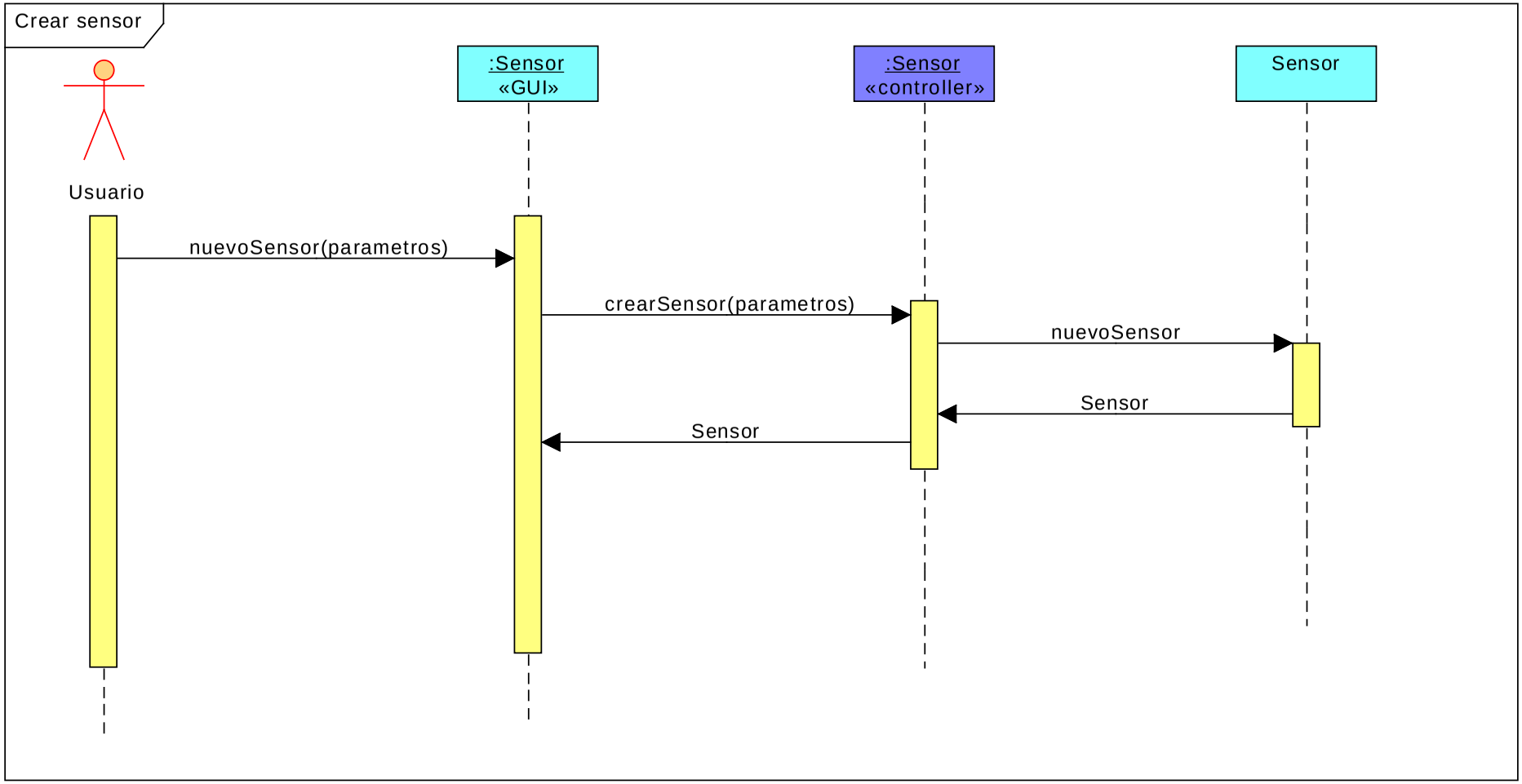


Figura 15: Diagrama de secuencia crear sensor

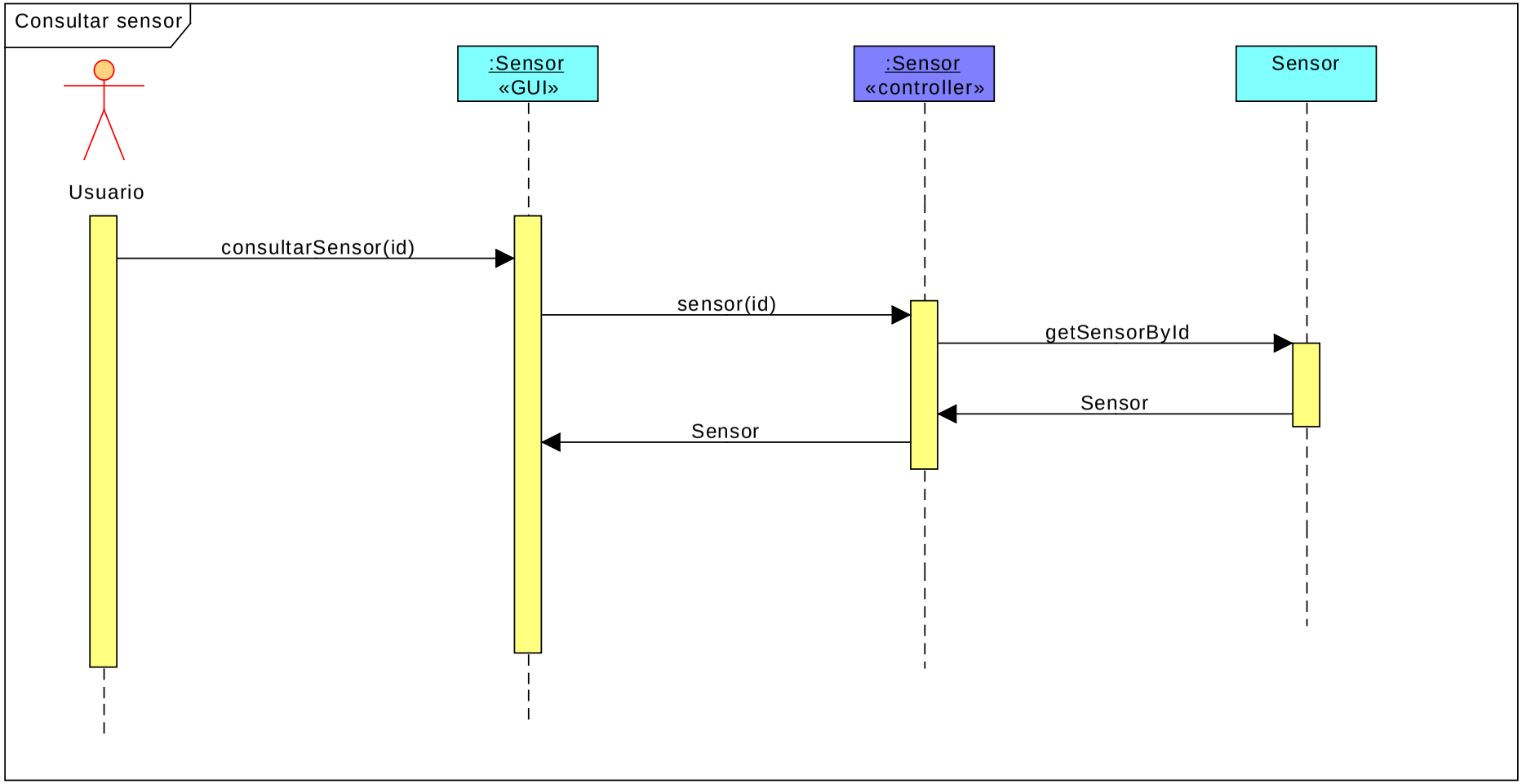


Figura 16: Diagrama de secuencia consultar sensor

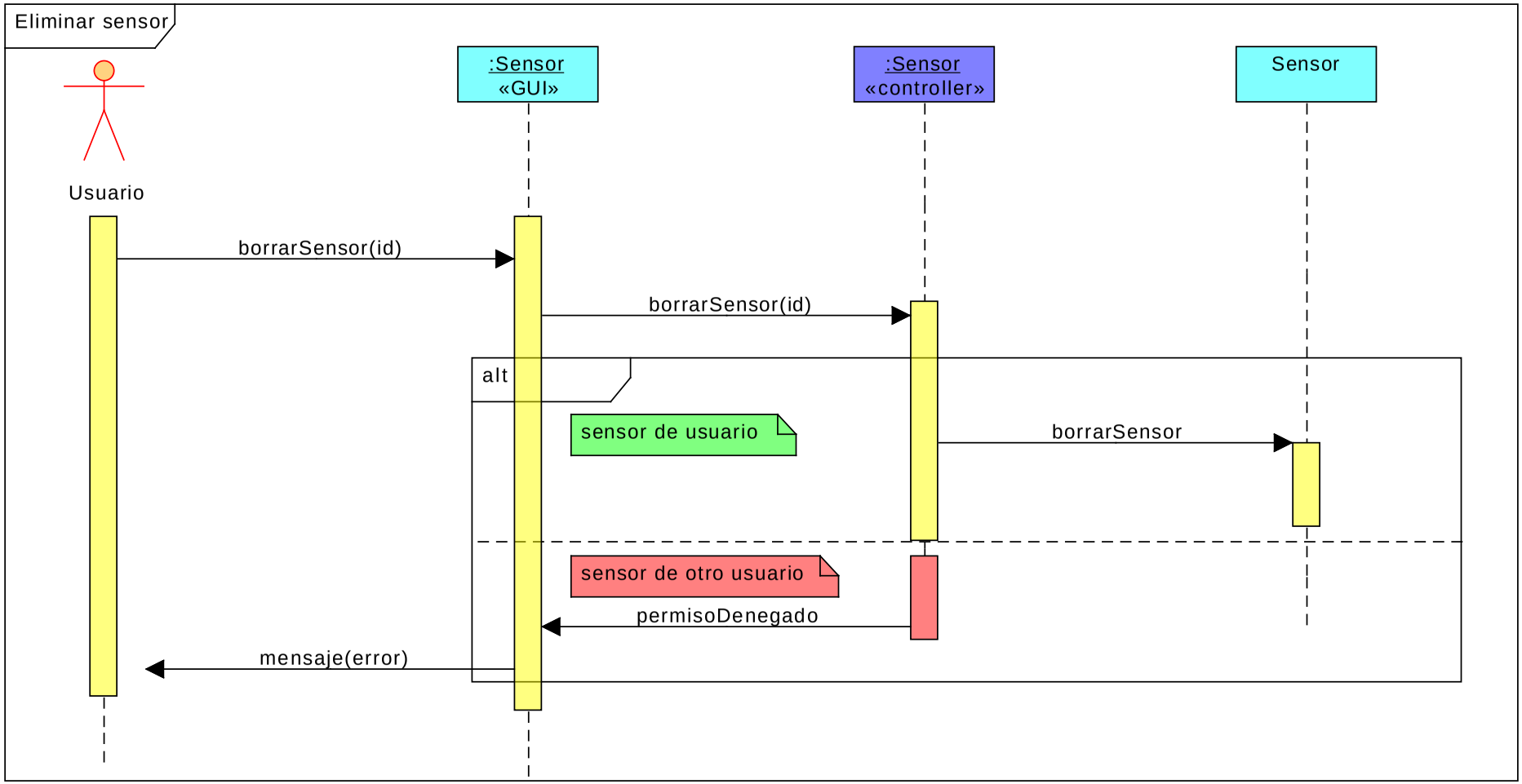


Figura 17: Diagrama de secuencia eliminar sensor

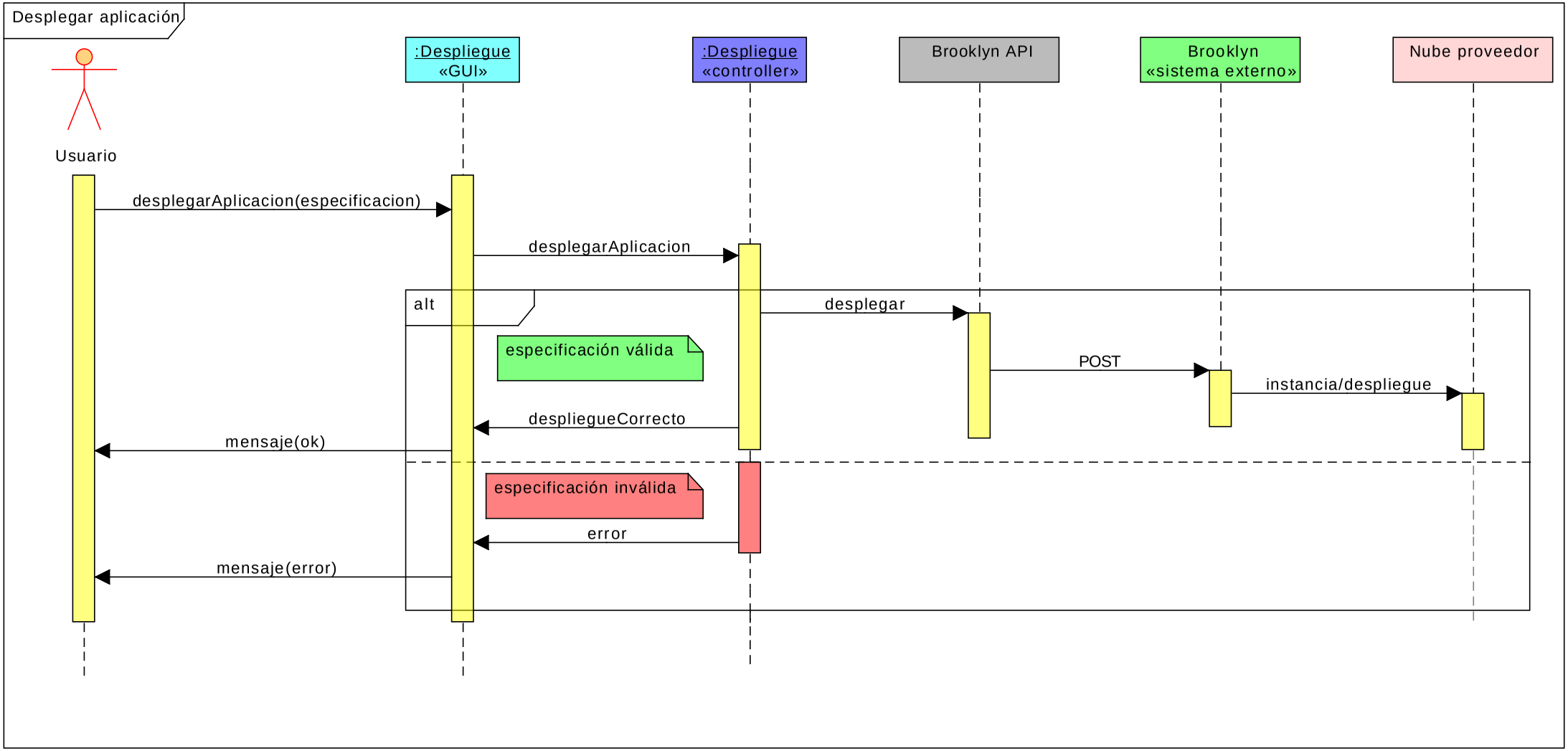


Figura 18: Diagrama de secuencia desplegar aplicación

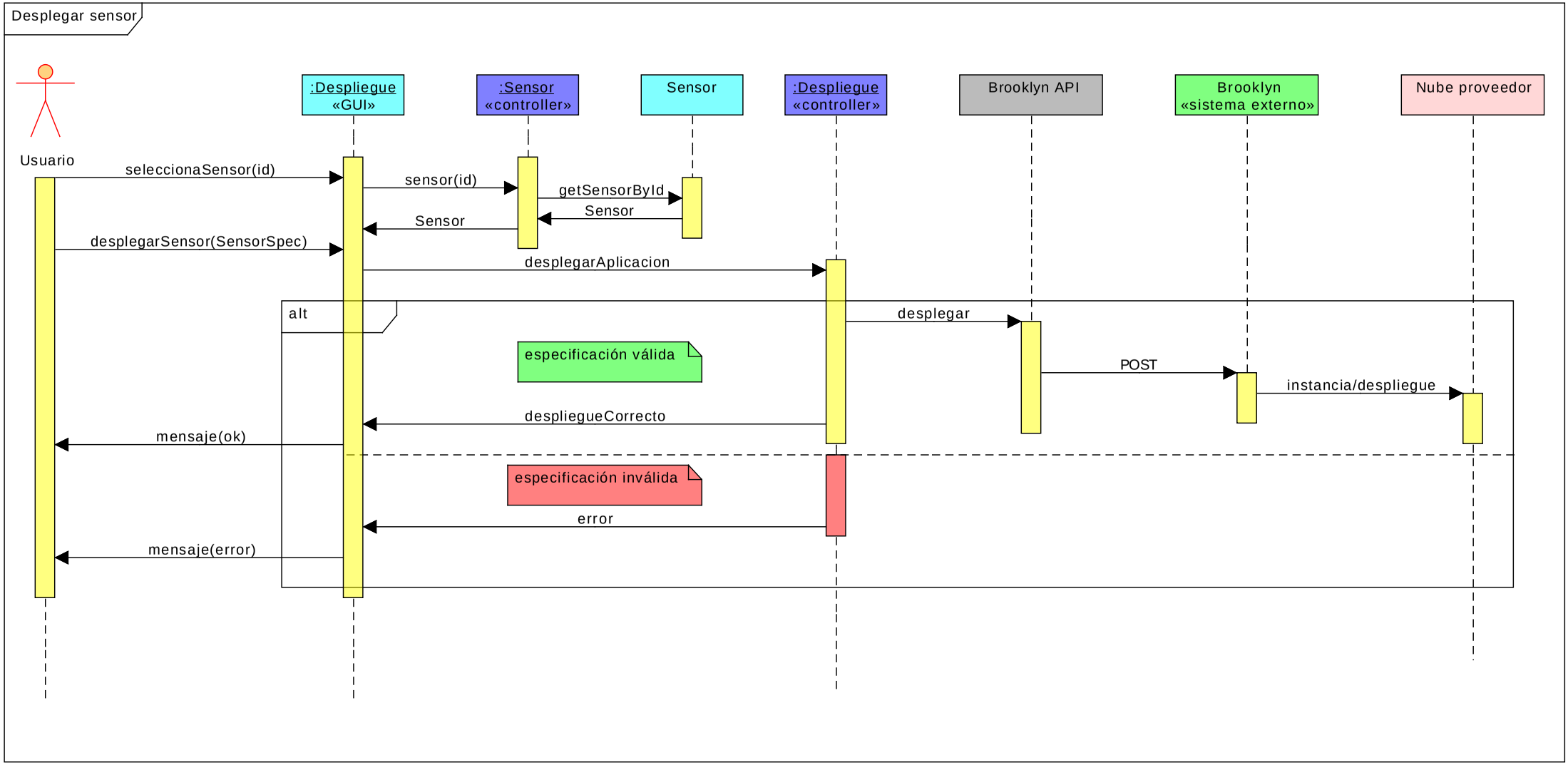


Figura 19: Diagrama de secuencia desplegar sensor

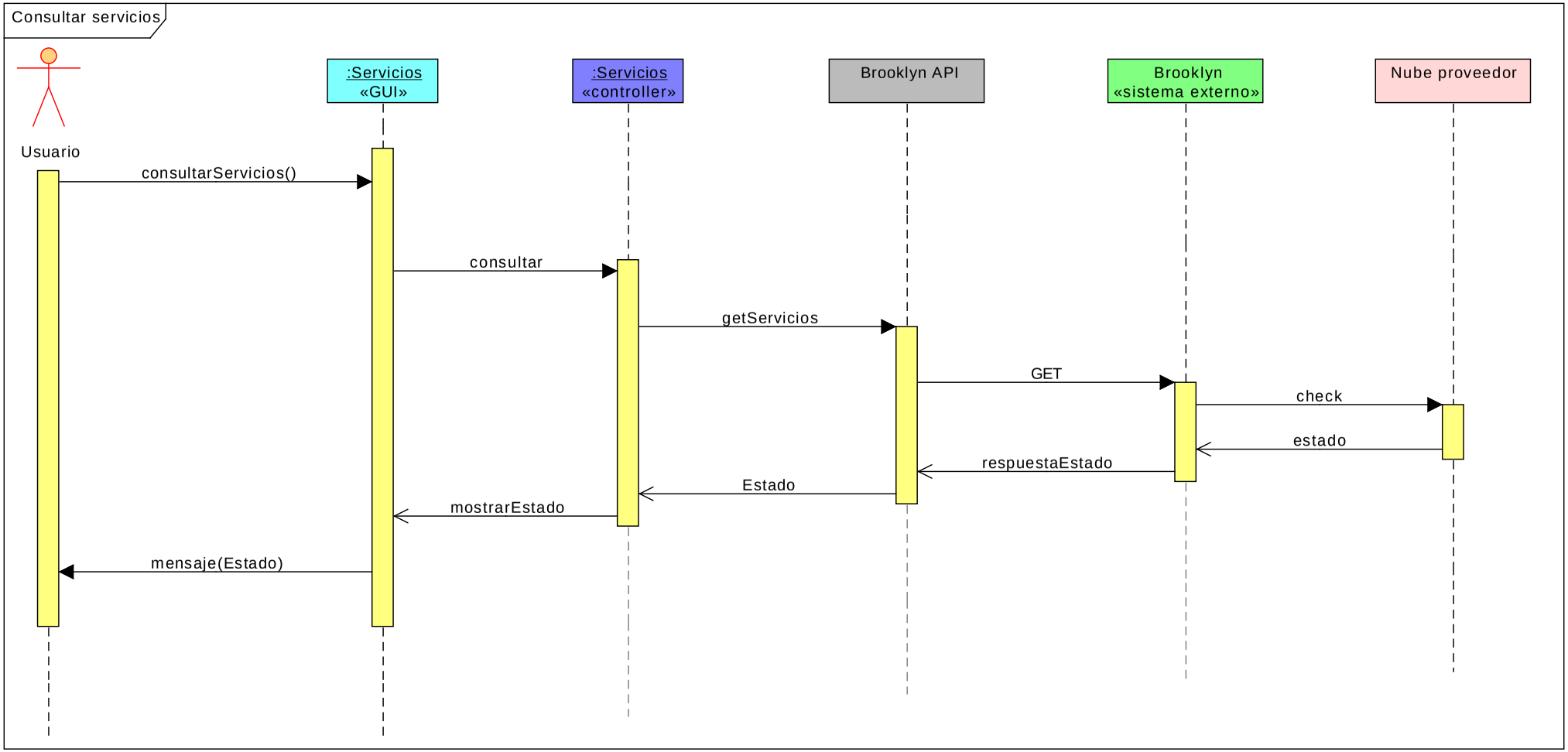


Figura 20: Diagrama de secuencia consultar servicios

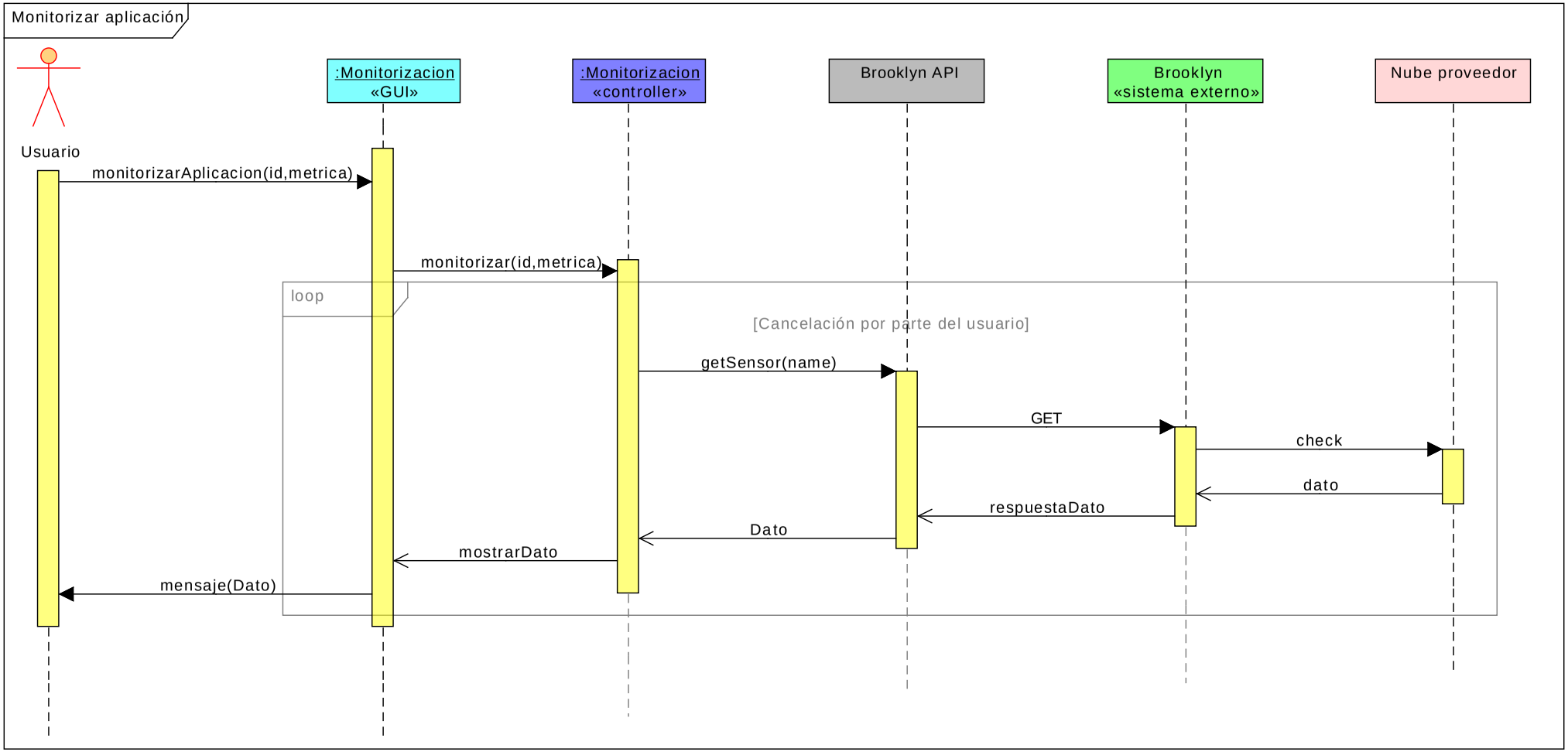


Figura 21: Diagrama de secuencia monitorizar aplicación

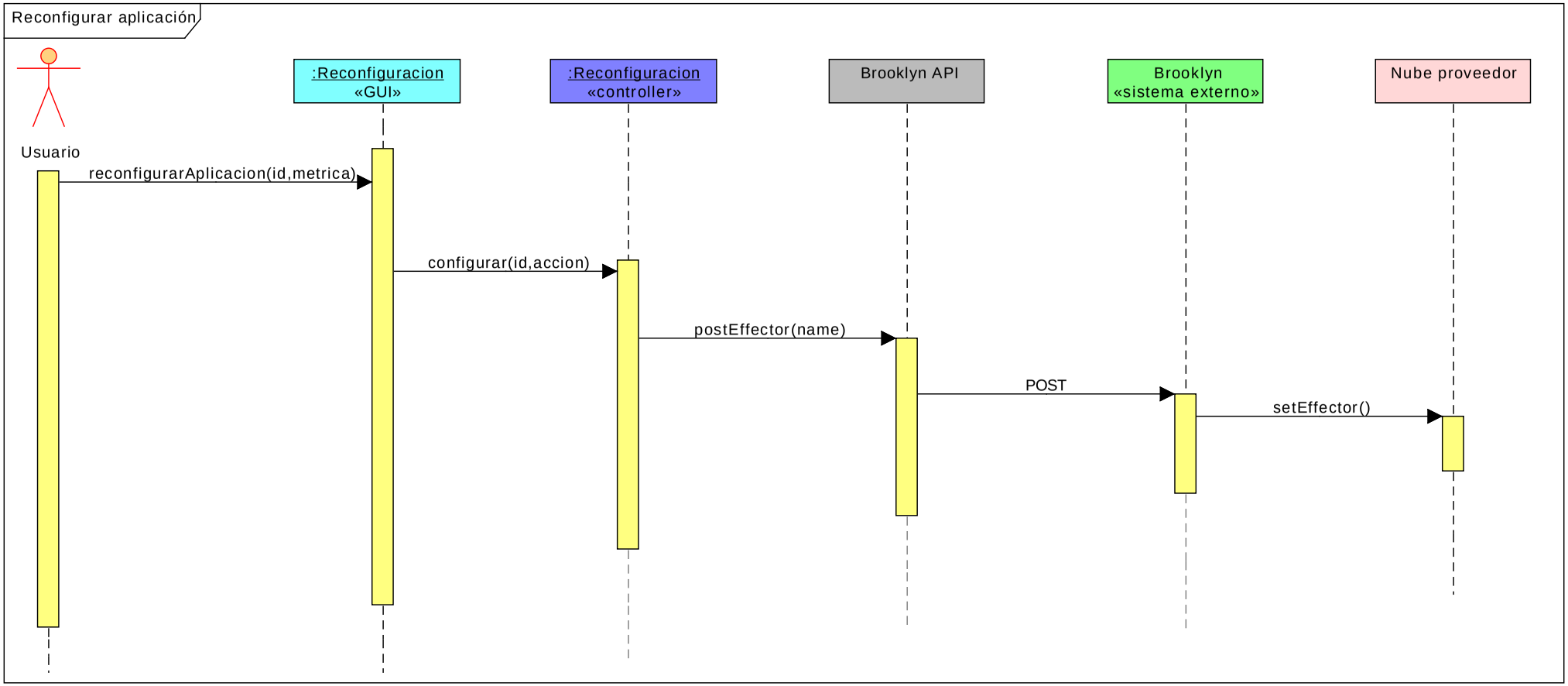


Figura 22: Diagrama de secuencia reconfigurar aplicación

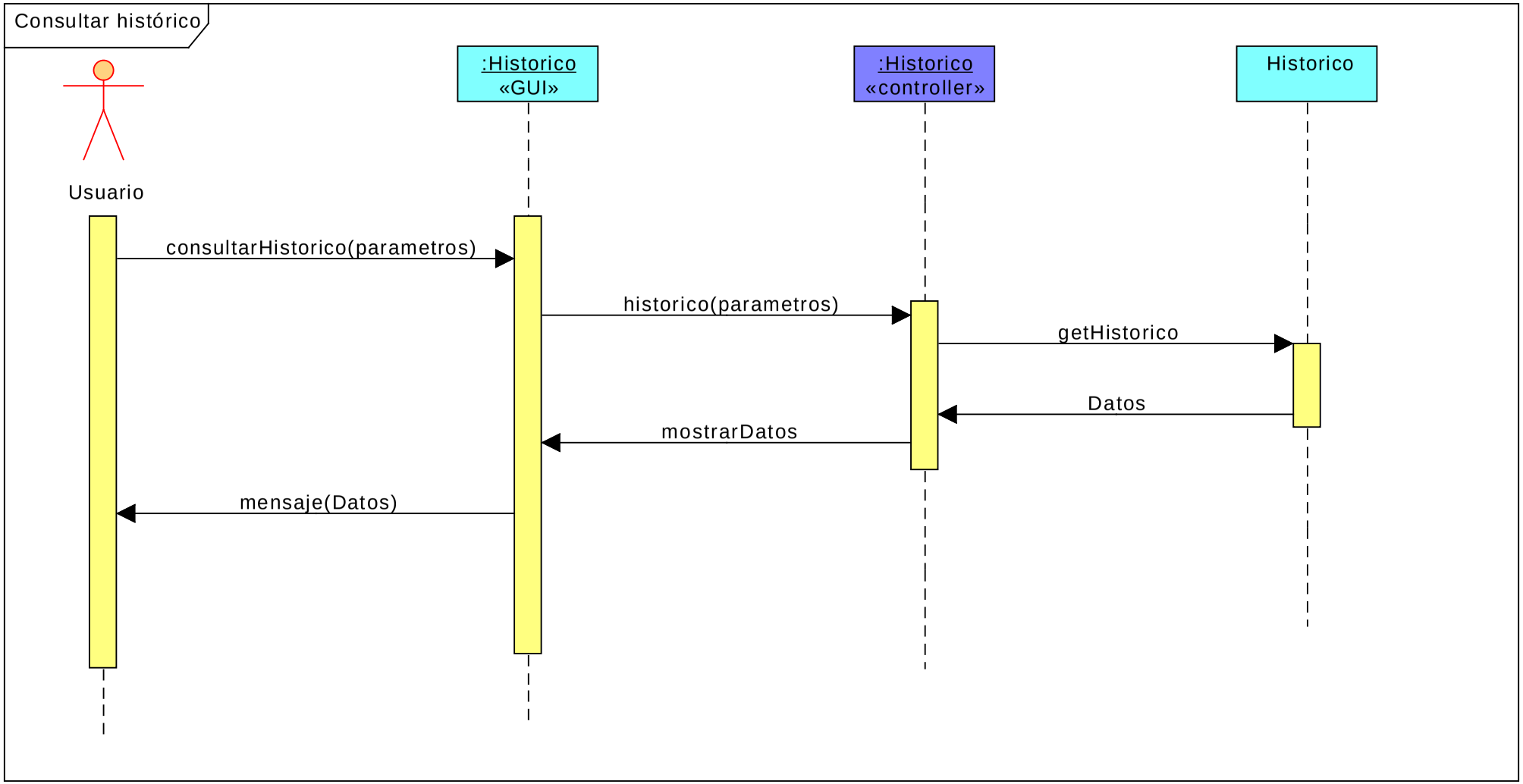


Figura 23: Diagrama de secuencia consultar histórico

3.3.3. Modelo de datos

En esta actividad vamos a definir la estructura física de los datos que utilizará el sistema. Para su definición, se ha de tener en cuenta el sistema de gestión de bases de datos (SGBD[II]) concreto a utilizar, los requisitos de nuestra infraestructura y las particularidades del entorno tecnológico.

La finalidad de este diseño es conseguir una mayor eficiencia en el tratamiento de los datos, que se traducirá, en una mejora en el rendimiento del sistema final.

En la figura 24 se representa el diagrama entidad-relación correspondiente al modelo físico de datos de la aplicación. Las tablas o documentos que aparecen en el modelo se detallan a continuación:

- User - información de perfil del usuario
- Session - información de la sesión del usuario
- Log - información de accesos de usuario para auditoría
- Sensor - especificación de sensor
- MonitoringData - datos obtenidos de la monitorización
- Entity - entidad o aplicación desplegada

[II]Sistema de gestión de bases de datos - conjunto de programas que permiten definir, construir y gestionar una base de datos

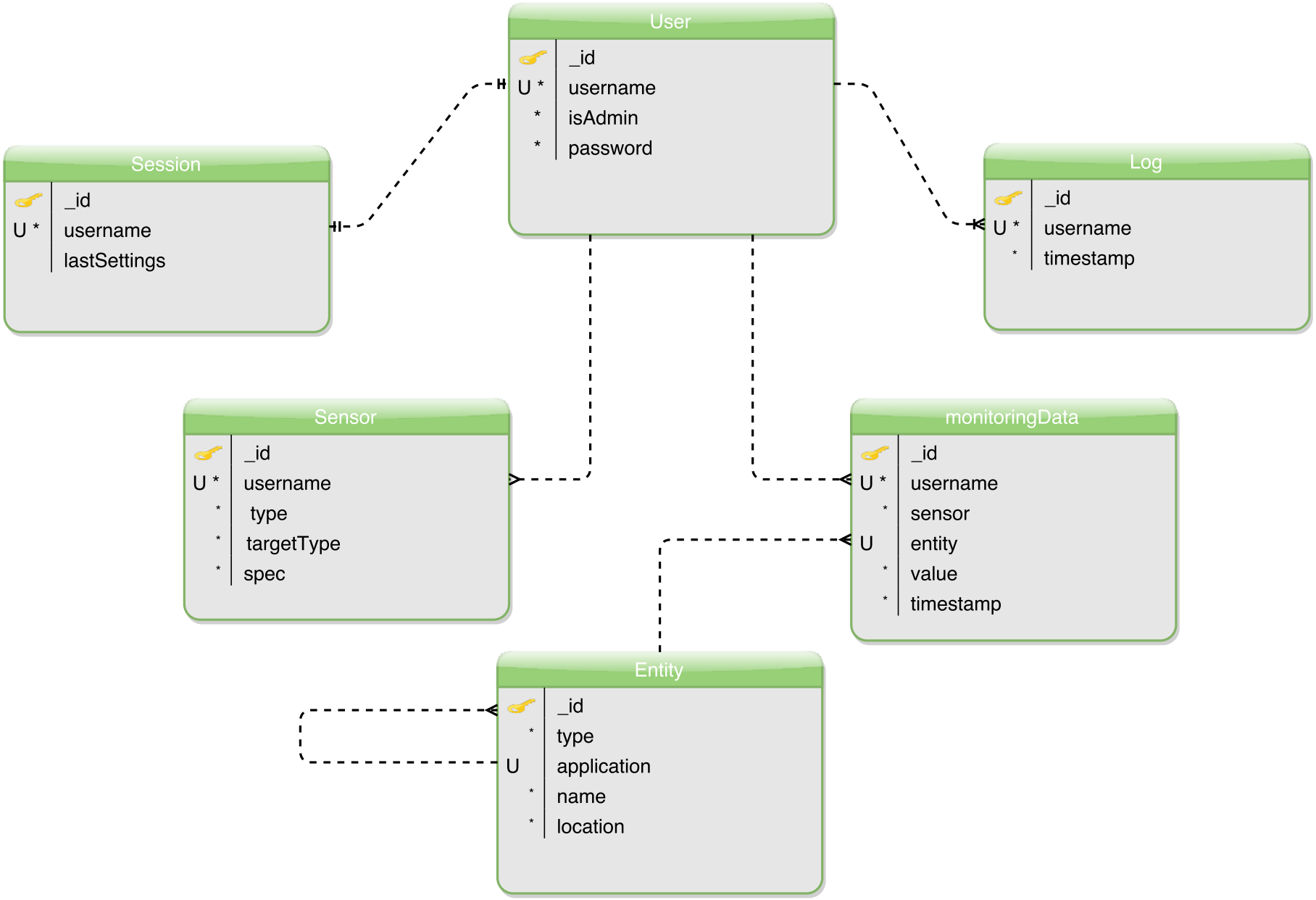


Figura 24: Diagrama de entidad-relación

3.4. Implementación de la infraestructura

En las secciones 3.1 a 3.3 se han documentado aquellas actividades de estudio, análisis, diseño y especificación necesarias para el desarrollo de la infraestructura de monitorización. El trabajo de implementación requiere asimismo la preparación de equipos físicos y lógicos para la puesta en marcha del desarrollo. En los siguientes apartados se definirán los medios empleados para el entorno de desarrollo así como las tecnologías involucradas en el mismo. Algunas de estas tecnologías ya han sido introducidas anteriormente, pero en este caso, las abordaremos haciendo hincapié en los aspectos técnicos que van a ser relevantes a lo largo del desarrollo.

3.4.1. Medios

IDE

Hoy en día para el desarrollo de software resulta imprescindible el uso de un IDE[III] para facilitar la edición de código fuente, la depuración y automatizar determinadas tareas. Tras la valoración de las alternativas disponibles, para nuestro desarrollo se ha optado por hacer uso de la herramienta *WebStorm* [50].

Sistema de control de versiones

Igual o mayor importancia para el desarrollo de proyectos, tienen los sistemas de control de versiones. *Git* [51], es sin duda unos de los software de control de versiones más populares. La herramienta *Bitbucket* [52], ofrece un repositorio que hace uso de Git. Además, incorpora herramientas para la gestión de incidencias que han sido de gran ayuda para la gestión del tiempo durante el desarrollo.

Equipos físicos

Para la puesta en marcha de la infraestructura final en producción se ha hecho uso de un máquina servidor propia basada en una distribución *Debian stable*. El conjunto de especificaciones técnicas sobre el equipo servidor se encuentran en el Anexo I.

[III] IDE - entorno de desarrollo integrado, en inglés Integrated development environment

3.4.2. Tecnologías

Para ayudar al lector a entender la implementación final, es necesario introducir las principales tecnologías involucradas en el desarrollo, así como sus características más relevantes, y aquellas por las cuales se han seleccionado de entre las alternativas existentes en cada ámbito.

Nodejs

Nodejs [53] es un entorno escalable y dirigido a eventos construido como una implementación de *Javascript* del lado del servidor que cambia la noción de cómo debería trabajar éste. Su meta es permitir a un programador construir aplicaciones altamente escalables y escribir código capaz de manejar decenas de miles de conexiones simultáneas en una sola máquina física. Su capacidad asíncrona para el procesamiento de peticiones de forma concurrente es una de sus grandes ventajas con respecto a otros lenguajes y frameworks de desarrollo de aplicaciones web.

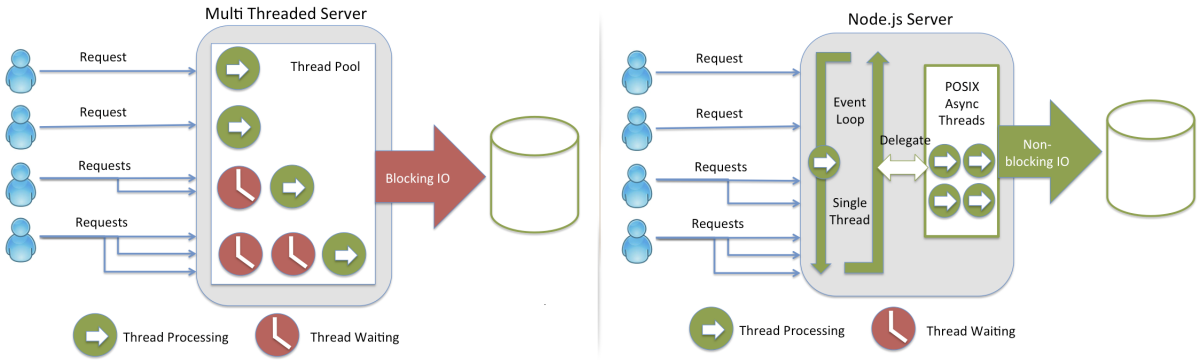


Figura 25: Filosofía síncrona Multi-thread frente la asíncrona de Nodejs

Nodejs unido al framework *Express* [54], proporciona un marco para el desarrollo de aplicaciones web basadas en el patrón MVC de una forma rápida y potente. Según sus creadores, Express es un framework de desarrollo de aplicaciones web minimalista y flexible para Nodejs. Entre sus principales características destacan gestión de cookies y sesiones, enrutamiento, y su extensa su colección de middleware[IV]. Todo esto hace que el desarrollo de servicios *REST* [V] con esta combinación de herramientas sea una elección acertada.

[IV] Middleware - software de computadora que asiste a a una aplicación para interactuar o comunicarse con otras para que puedan intercambiar datos entre éstas.

[V] REST - conjunto de principios para el desarrollo arquitecturas para servicios web que tienen como principios los recursos del sistema, cómo se accede a ellos y cómo se transfieren a través del protocolo *HTTP* hacía los clientes.

MongoDB

MongoDB [55] es una de las bases de datos *NoSQL*[VI] más populares en la actualidad que ha sido creada para proporcionar escalabilidad, rendimiento y gran disponibilidad. Su agilidad permite a los esquemas cambiar rápidamente cuando las aplicaciones evolucionan, proporcionando siempre la funcionalidad que los desarrolladores esperan de las bases de datos tradicionales, tales como índices secundarios, un lenguaje completo de búsquedas y consistencia estricta.

MongoDB brinda un elevado rendimiento, tanto para lectura como para escritura, potenciando la computación en memoria (in-memory). La replicación nativa y su tolerancia a fallos automática ofrece un entorno fiable a nivel empresarial y flexibilidad operativa.

Para el almacenamiento de sus estructuras de datos hace uso de documentos tipo *JSON*[VII] con un esquema dinámico (formato *BSON*), por lo que la integración de los datos en ciertas aplicaciones resultan más fáciles y rápidas.

Apache Brooklyn

Apache Brooklyn es una herramienta de la cuál ya se habló en la sección 3.1.3 dedicada al estudio y valoración de las alternativas disponibles. Corresponde a este apartado ahondar más acerca de sus detalles técnicos y su funcionalidad.

Las *entities* o entidades suponen el elemento más importante de Brooklyn. Estas se encuentran definidas mediante un catálogo [VIII]. Entre los distintos tipos de entidades podemos encontrar por ejemplo aquellas destinadas al despliegue de Servidores Web (Tomcat, JBoss, Jetty, ngingx, etc.) o Bases de Datos (MySQL, MongoDB, Redis, etc.). A través de ellas es posible modelar y desplegar aplicaciones que trabajen en el contexto de cualquier tecnología que se encuentre definida en el catálogo.

La descripción de una aplicación se realiza a través de la interconexión de entidades, las cuales gestionan su ciclo de vida a través de un componente denominado *Management Context*.

Una característica fundamental del modelo de las entidades es su capacidad para ser padres unas de otras. Así, se forman colecciones de entidades cuyo padre de nivel superior será la aplicación. La Figura 26 presenta de forma ilustrada esta jerarquía.

Las entidades contienen *sensors* o sensores que informan de su actividad y del estado de dicha entidad. Estos sensores se encuentran predefinidos para cada tipo de entidad,

[VI] NoSQL - es una clase de SGBD que difieren del modelo clásico del sistema de gestión de bases de datos relacionales en aspectos relevantes como su tipo de esquema y su organización. Este tipo de tecnologías tratan de resolver las cuestiones de escalabilidad, agilidad y rendimiento en el manejo de gran cantidad de datos.

[VII] JSON - http://json.org/

[VIII] Catálogo Brooklyn - https://brooklyn.incubator.apache.org/learnmore/catalog/index.html

siendo posible también su creación mediante especificaciones.

Los *effectors* o actuadores, constituyen las operaciones que pueden ser invocadas a través de una entidad. Estos elementos, van a permitir realizar reconfiguraciones sobre las entidades o aplicaciones.

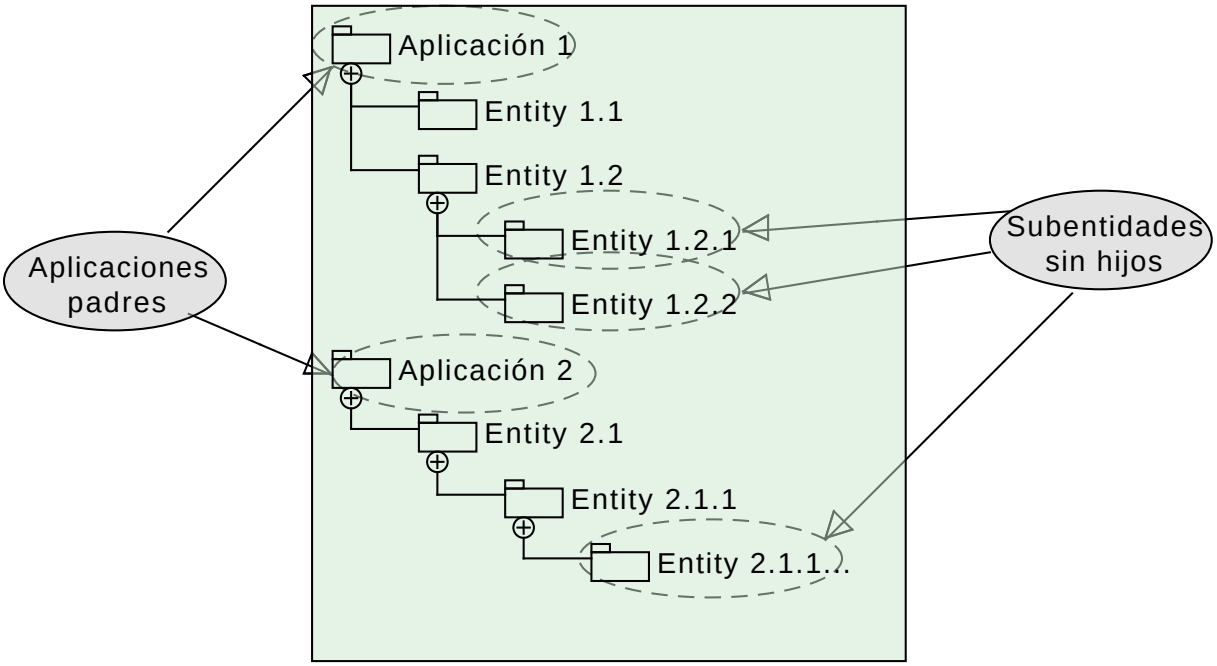


Figura 26: Representación jerárquica de aplicaciones y entidades en Brooklyn

Brooklyn, al estar basado en Java y en Groovy, permite realizar el modelado de aplicaciones con cualquiera de estas tecnologías. Además, también incorpora un mecanismo para la descripción de aplicaciones mediante uso de documentos YAML [59], alineándose así con el estándar de OASIS-CAMP [60]. Para nuestro sistema, se hará uso también del formato YAML para la especificación de sensores y aplicaciones.

Dygraphs

Dygraphs [56] es una librería JavaScript para la representación de gráficas. Tiene licencia de software libre y se pone a disposición de los desarrolladores como una herramienta rápida, flexible y que permite manejar conjuntos de datos de alta densidad.

Al trabajar de forma pura en el cliente navegador, Dygraphs no hace uso de ningún tipo de servidor para el procesamiento de datos. Sus características la hacen una alternativa potente y eficiente para la representación de gráficas con gran cantidad de datos y hacen que estos sean explorables e interpretables por los usuarios de forma interactiva.

JQuery

JQuery [57] podría definirse como un framework o una API de funciones, su potencia, su sencillez y su carácter de software libre han hecho crecer su popularidad enormemente en los últimos años.

Es uno de los complementos más esenciales para el desarrollo web, ya que nos facilita mucho el desarrollo de aplicaciones enriquecidas del lado del cliente, en lenguaje Javascript, lo que lo hace compatible con todos los navegadores.

Bootstrap

Bootstrap [58] es un framewrok front-end compuesto por un conjunto de herramientas para el diseño de aplicaciones web. Fue implementada por desarrolladores del equipo de Twitter, y liberada por este posteriormente en 2011 como una solución de código abierto.

Entre sus características principales, destacan su diseño modular, la flexibilidad de sus componentes, sus plantillas y su compatibilidad con los diferentes navegadores. Aunque destacan muchas de sus funcionalidades, el apartado visual es una de sus grandes ventajas, proporcionando mecanismos para adaptarse a cualquier tipo de pantalla, y por consiguiente a cualquier tipo de dispositivo (móvil, PC, tablet, etc.).

3.4.3. Descripción de la solución

La solución adoptada para la monitorización de la plataforma permite la creación y el despliegue de sensores para la obtención de métricas. Este conjunto de métricas van desde aquellos parámetros de la plataforma física (Máquina virtual), como de la plataforma software en la que se encuentre ejecutándose el servicio, como en este caso lo es una distribución linux, o una plataforma o entorno tecnológico concreto como puede ser una base de datos o un servidor web. Es importante señalar que esta infraestructura permite el despliegue de sensores a través de servicios IaaS, pero no por ello la monitorización se realiza únicamente a nivel físico del sistema. De esta forma, el sistema desarrollado es capaz de monitorizar a través de sensores a nivel de una plataforma concreta, ya sea hardware o software, a bajo o alto nivel.

Para la solución final, los escenarios que se han tenido en cuenta para el despliegue y distribución de aplicaciones son aquellos que nos permite la herramienta Brooklyn, junto con su librería para el despliegue. La infraestructura es capaz por tanto, de desplegar aplicaciones (y sensores) en cualquier servicio IaaS de los proveedores que implementa jclouds.

Mediante Brooklyn, se han realizado una serie especificaciones de sensores en formato YAML. Estos sensores se dividen en dos tipos, sensores *ssh* y sensores *http*. El primero realiza su función de recopilación de datos a través de comandos ssh y para su creación el usuario únicamente deberá introducir el tipo del mismo y el comando que desee ejecutar por consola. El segundo utiliza peticiones http a un servidor y para su especificación en

lugar del comando a ejecutar, se requiere la dirección y el path del servidor y recurso de un objeto JSON.

Estas especificaciones, y el conjunto de funcionalidades es proporcionada tal y como vemos en la Figura 9 mediante la *plataforma de monitorización* que constituye el elemento fundamental de la arquitectura del sistema. Esta plataforma ha sido desarrollada como aplicación web haciendo uso de Nodejs para la creación de la API REST que da soporte a las rutas que podemos ver en la Tabla 8. El usuario se comunica con esta API a través de las diferentes vistas haciendo uso de un navegador web.

La interfaz de la aplicación es accesible desde cualquier navegador. Con el uso de Bootstrap para proporciona una interfaz limpia, intuitiva y con elementos dinámicos que resultan de ayuda a la comprensión de las diferentes transiciones de cada operación. Además, la interfaz se encuentra diseñada para adaptarse a cualquier tipo de pantalla (monitor, móvil o tablet). La lógica de la aplicación del lado del cliente se encuentra desarrollada mediante JQuery, que es encargada de manejar los distintos flujos de información e intercambiarlos con el cliente mediante el uso de la tecnología *Ajax* [IX].

La librería Dygraphs, permite que podamos presentar los datos obtenidos de la monitorización de una forma adecuada e interactiva. Se ha desarrollado además un mecanismo para la anotación y marcado de elementos relevantes en gráficas, así como permitir indicar el conjunto de medias móviles con el objeto de suavizar y obtener una tendencia de la serie. Las gráficas, son obtenidas y alimentadas en tiempo real por el conjunto de datos obtenidos por los sensores encargados de recopilar datos a partir de métricas.

En la creación de sensores para la monitorización se han tomado como referencia de métricas aquellas proporcionadas por la herramienta collectl para los siguientes subsistemas: CPU, Disco, Sistema de ficheros, Memoria y Red. También se han desarrollado, entre otros, sensores para la monitorización de los parámetros de rendimiento de una base de datos MongoDB. Estos sensores se han especificado como de tipo ssh, tal y como se comento anteriormente, y por ello serán interpretables por la API de Brooklyn para su despliegue. Además de estos sensores proporcionados por el sistema, cada usuario puede especificar y almacenar su propia colección de sensores, que serán solo visibles desde su sesión.

La base de datos MongoDB que forma parte de la arquitectura del sistema, es la encargada del almacenamiento de todos los objetos de los cuales hará uso la aplicación. Entre ellos se incluyen aquellos especificados en la Figura 24. Para la puesta en marcha del sistema en producción, la base de datos contará con el conjunto de especificaciones de los sensores nombrados con anterioridad, que serán accesibles y seleccionables por cualquier usuario para su despliegue.

Como se puede observar en la Figura 9, la funcionalidad aportada va a permitir a un usuario avanzado (a la derecha de la imagen) realizar la especificación tanto de aplicaciones como de componentes para su posterior despliegue en diferentes proveedores de

[IX] Ajax - https://es.wikipedia.org/wiki/AJAX

Nubes (que serán descritos en la especificación) a través de la API de Brooklyn. Posteriormente, un usuario no técnico o directivo podría ser capaz de interpretar y analizar estos resultados (parte superior de la imagen) y tomar decisiones sobre la configuración de sus servicios en base a determinados requisitos. Estas decisiones, pueden ser notificadas al usuario avanzado con el objetivo de que realiza las reconfiguraciones necesarias, y tras esto, el usuario directivo puede observar mediante la presentación de los resultados de forma visual y gráfica el cambio en el estado de sus servicios según los diferentes sensores que desee observar. De igual forma, es posible que se desee notificar por parte del directivo al personal técnico la necesidad de observar otro tipo de métrica. Esto conllevará de nuevo la necesidad de especificación por parte del técnico de nuevas especificaciones de sensores que podrán proporcionar nuevas métricas para su visualización. Este ciclo se repite y proporciona la capacidad no sólo de observar y decidir en tiempo real, si no que cualquier usuario va a disponer de los datos monitorizados almacenados en el histórico para su consulta.

El conjunto de mensajes y operaciones intercambiadas entre los usuarios y los servicios se realiza a través de la API de Brooklyn y de forma transparente al usuario. El uso de interfaces gráficas adecuadas y una presentación de la información intuitiva permite a cada tipo de usuario comprender y seguir el flujo de los procesos y datos en el sistema.

Un ejemplo de especificación de una aplicación lo podemos ver en la Figura 27. El despliegue, monitorización y reconfiguración de servicios es también realizada a través de la interfaz gráfica como vemos en las Figuras 29, 31 y 37 de forma que esta información será enviada a través de la API de Brooklyn para que se ejecute la acción o proceso pertinente.

A continuación en la Tabla 8 se detallan las diferentes rutas REST de acceso a los recursos del sistema. En la descripción de cada ruta se especifica el tipo de recurso que es devuelto o se envía al servidor Nodejs por parte del cliente. En el caso de las vistas, es el usuario el que accede a ellas a través del menú. En el resto de los casos, estas rutas son manejadas por la lógica que se ejecuta en el cliente navegador y actúan de forma transparente al usuario conforme éste navega por los diferentes menús interactivos o realiza operaciones mediante formularios.

Método	URI	Descripción
GET	/	Acceso al sistema*
POST	/login	Inicio de sesión de usuario
GET	/user/profile	Vista Perfil del usuario
POST	/user/profile/modusr	Modificación de nombre de usuario
POST	/user/profile/modpass	Modificación de contraseña de usuario
POST	/user/new	Creación de un nuevo usuario (administrador**)
POST	/user/del/<id_usuario>	Eliminar usuario (administrador**)
GET	/user/settings	Vista Configuración del usuario
POST	/user/settings/brooklyn	Modificación de configuración de la máquina Brooklyn
GET	/user/brooklyn	Vista de despliegue
POST	/user/brooklyn/deploy	Despliegue de una especificación
GET	/user/sensors	Vista de sensores
POST	/user/sensors/search	Búsqueda de sensores
POST	/user/sensors/del/<id_sensor>	Eliminar un sensor
POST	/user/sensors/new	Creación de un sensor
GET	/user/monitoring	Vista de monitorización
GET	/user/brooklyn/apps	Lista de jerárquica de aplicaciones desplegadas
GET	/user/brooklyn/entity/<id_app>/<id_entity>	Devuelve la entidad concreta
GET	/user/brooklyn/sensorvalues/<id_app>/<id_entity>	Devuelve los valores de los sensores correspondientes a esa entidad
GET	/user/brooklyn/sensor/<id_app>/<id_entity>/<sensor>	Devuelve el valor de un sensor
GET	/user/brooklyn/effectors/<id_app>/<id_entity>/	Devuelve la lista de effectors de la entidad
POST	/user/brooklyn/sensor/<id_app>/<id_entity>/<effector>	Ejecuta un effector
GET	/user/historical	Vista de histórico
POST	/user/historical/search	Búsqueda en el histórico
POST	/user/brooklyn	Vista de monitorización
GET	/logout	Cierre de sesión de usuario

Tabla 8: conjunto de recursos de la REST API del sistema

*Acceso a la vista principal si el usuario se encuentra autenticado, en caso contrario se redirige a /login.

**En estos casos el sistema devuelve un error de Permiso Denegado (403) si el usuario no se encuentra autorizado para realizar esta acción.

4. Resultados y Pruebas de análisis

Con la puesta en marcha de la infraestructura en el servidor de producción, y los distintos elementos que la componen, se han realizado una serie de pruebas y casos de uso reales de despliegue y gestión de aplicaciones distribuidas para su monitorización.

Para las distintas operaciones de despliegue y pruebas se ha tomado como ejemplo una aplicación web que implementa un simple Chat sobre un servidor *JBoss*, junto con una base de datos *MySQL*. Su especificación en formato YAML puede verse en la Figura 27. Como puede observarse en esta figura, el servidor se encuentra sobre un *cluster* que hace uso de un balanceador de carga *nginx*.

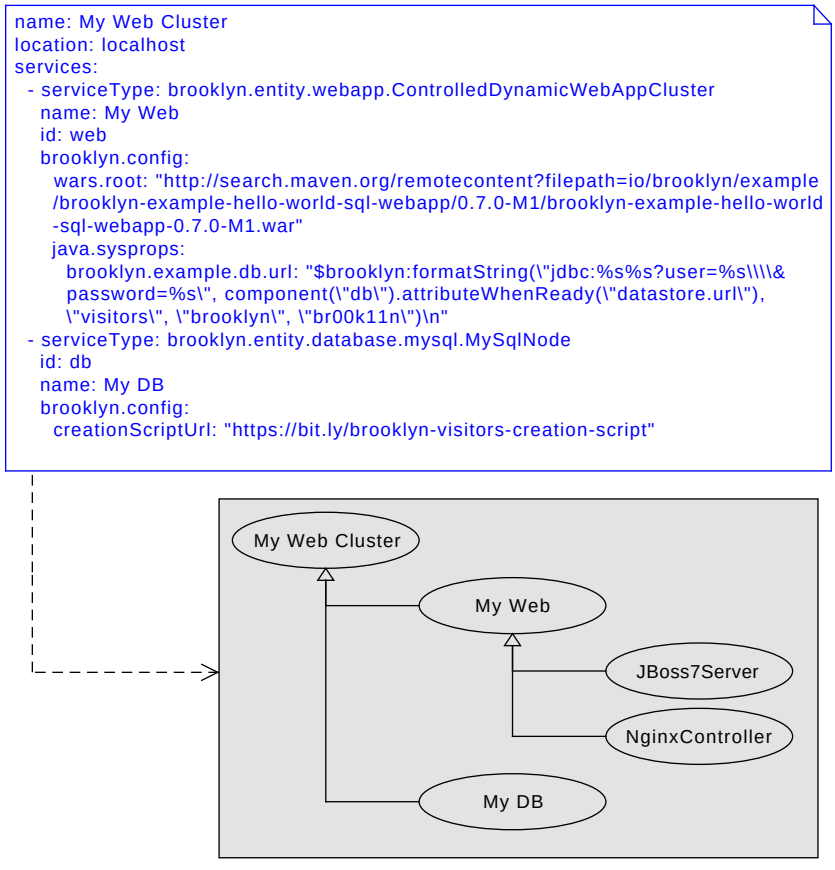


Figura 27: Especificación de aplicación web de Chat en formato YAML y la estructura generada en Brooklyn

La información sobre la localización de los elementos puede describirse tanto para cada uno de ellos como para la aplicación completa.

Para la selección y configuración de sensores, el sistema implementa un sistema interactivo que permite añadir aquellos sensores que deseemos para cada una de las localizaciones. De esta forma si se desea añadir una serie de sensores a una localización concreta que ya se encuentra ejecutando alguna aplicación que deseamos monitorizar, simplemente podemos añadir una especificación con su localización o localizaciones, que serán detectadas y mediante el menú podremos seleccionar los elementos que queramos desplegar. En la figura 28 podemos encontrar una descripción del procedimiento para la especificación de sensores en varias localizaciones.

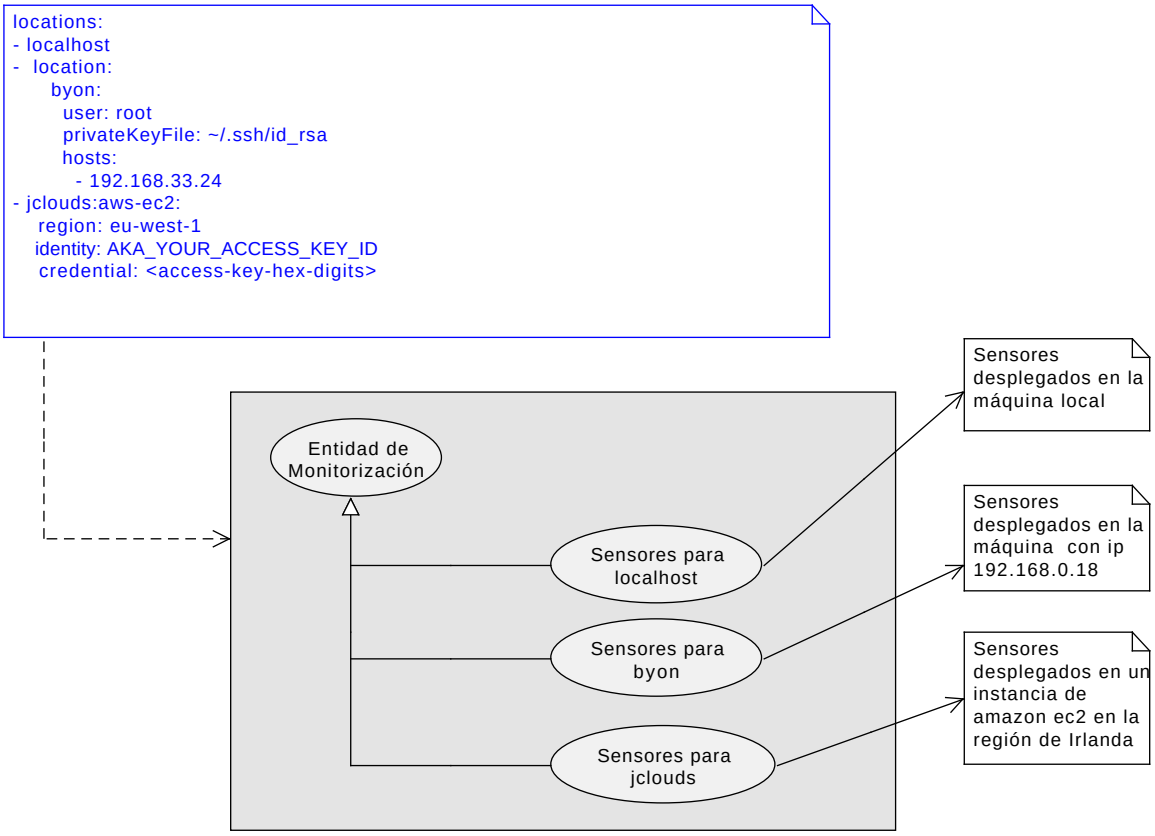


Figura 28: Especificación de localizaciones en formato YAML y despliegue de sensores

Para el caso de la aplicación web de chat, se ha realizado distribuyendo los componentes de la siguiente forma:

- Componente web: desplegado en máquina local
- Componente de base de datos: desplegado en una instancia de tipo t1.micro en el servicio *Amazon EC2* en la región de Irlanda (eu-west)

Las Figuras 29 a 34 muestran el procedimiento de despliegue del caso de uso, y los datos obtenidos por la monitorización de los parámetros de CPU y de peticiones a la base de datos y al servidor web. La aplicación, ha sido sometida a valores de estrés con el fin de mostrar su comportamiento en diferentes condiciones. Estas condiciones de estrés se

han realizado mediante la inserción continuada de mensajes en la aplicación de chat, cuya interfaz podemos observar en la Figura 35.

La Figura 36 muestra las instancias que se encuentran ejecutándose en la consola de Amazon EC2, para ello se ha creado previamente un usuario en los servicios de Amazon, cuyas credenciales han sido especificadas para el despliegue.

Por último se ilustran las consultas realizadas a posteriori sobre los datos de monitorización recogidos, estos valores se pueden filtrar según diferentes condiciones de fecha y hora, tipo de métrica, así como rango del propio valor. En las Figuras 38 a 40 se pueden comprobar los resultados obtenidos mediante consultas.

Los resultados obtenidos, tanto en tiempo real como a posteriori, nos muestran información acerca de como se está comportando nuestro servicio. La facilidad para especificar nuevos sensores para una localización y para definir esta recogida de datos para casi cualquier métrica existente constituye una garantía para poder tomar decisiones robustas.

Este sistema,da soporte tanto a la fase de desarrollo como a la de producción de servicios o aplicaciones distribuidas. De esta forma, una entidad u organización puede usar el entorno para realizar testing sobre el comportamiento de sus sistemas antes de realizar el despliegue. Posteriormente también puede monitorizar de forma robusta su aplicación y poder decidir que datos desea recopilar y en que momento, mediante el despliegue de estos sensores de forma dinámica.

Todos los datos recopilados pueden ser susceptibles de análisis en tiempo real mediante la representación gráfica, y a lo largo de un periodo de tiempo mediante la consulta al histórico. Estas dos modalidades de análisis, permiten la toma de decisiones a nivel de negocio y a nivel operativo tanto a en tiempo real, como a medio y largo plazo. La presentación de los datos resulta adecuada tanto para usuarios técnicos como para aquellos no especializados. La capacidad para la definición de sensores de alto nivel, ayuda a acercar este tipo de sistemas a lo largo del nivel directivo. Este acercamiento facilita la comprensión y la rotura de la brecha existente entre el personal más técnico y el personal de dirección encargado de la toma de decisiones.

Por lo tanto, con la adopción de esta infraestructura, las entidades pueden ver cubiertas sus necesidades operativas y de negocio a través de todo el ciclo de vida tanto de desarrollo, como producción y mantenimiento de sus aplicaciones en la Nube.

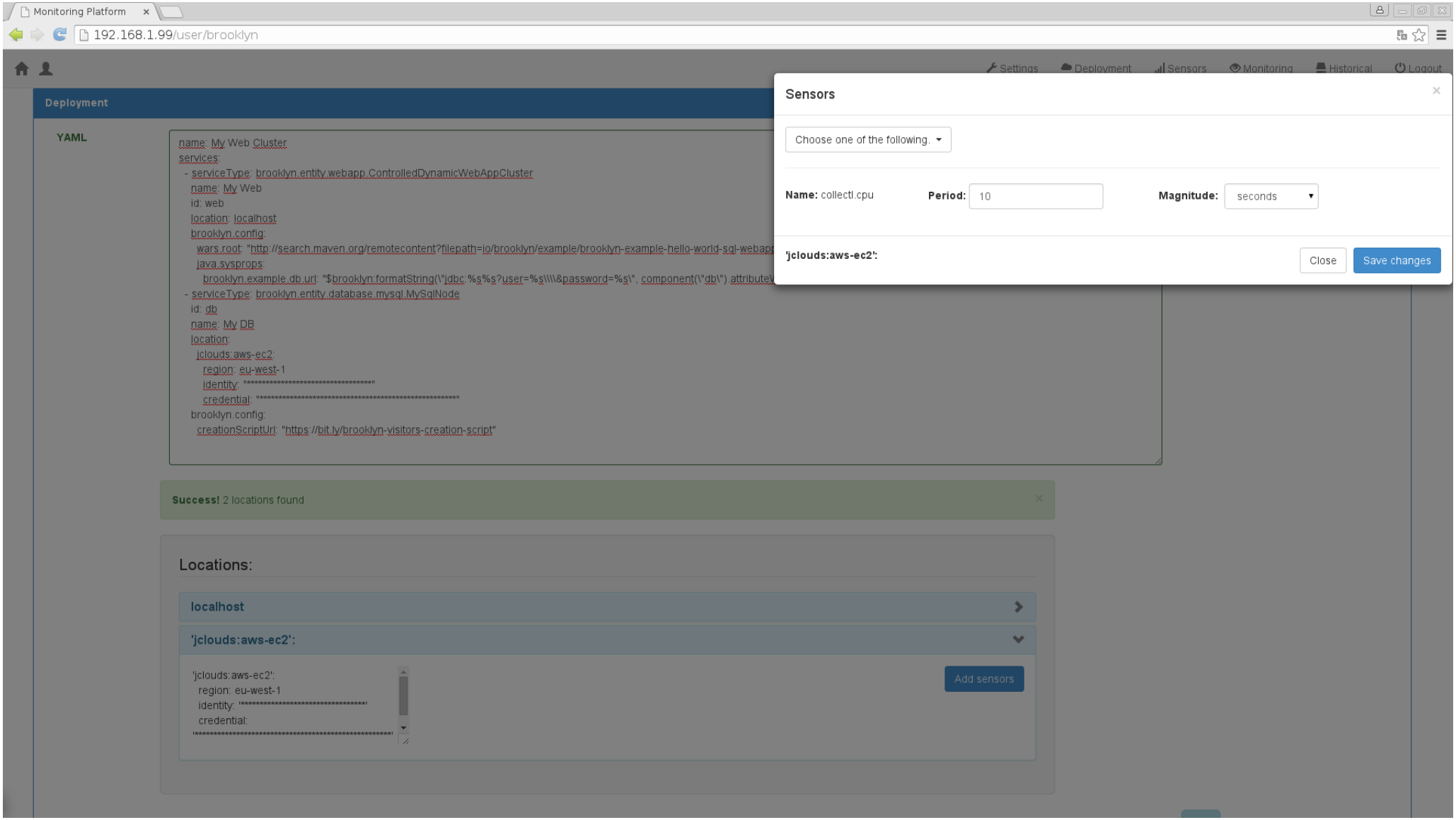


Figura 29: Especificación de servicios y selección de sensores para despliegue

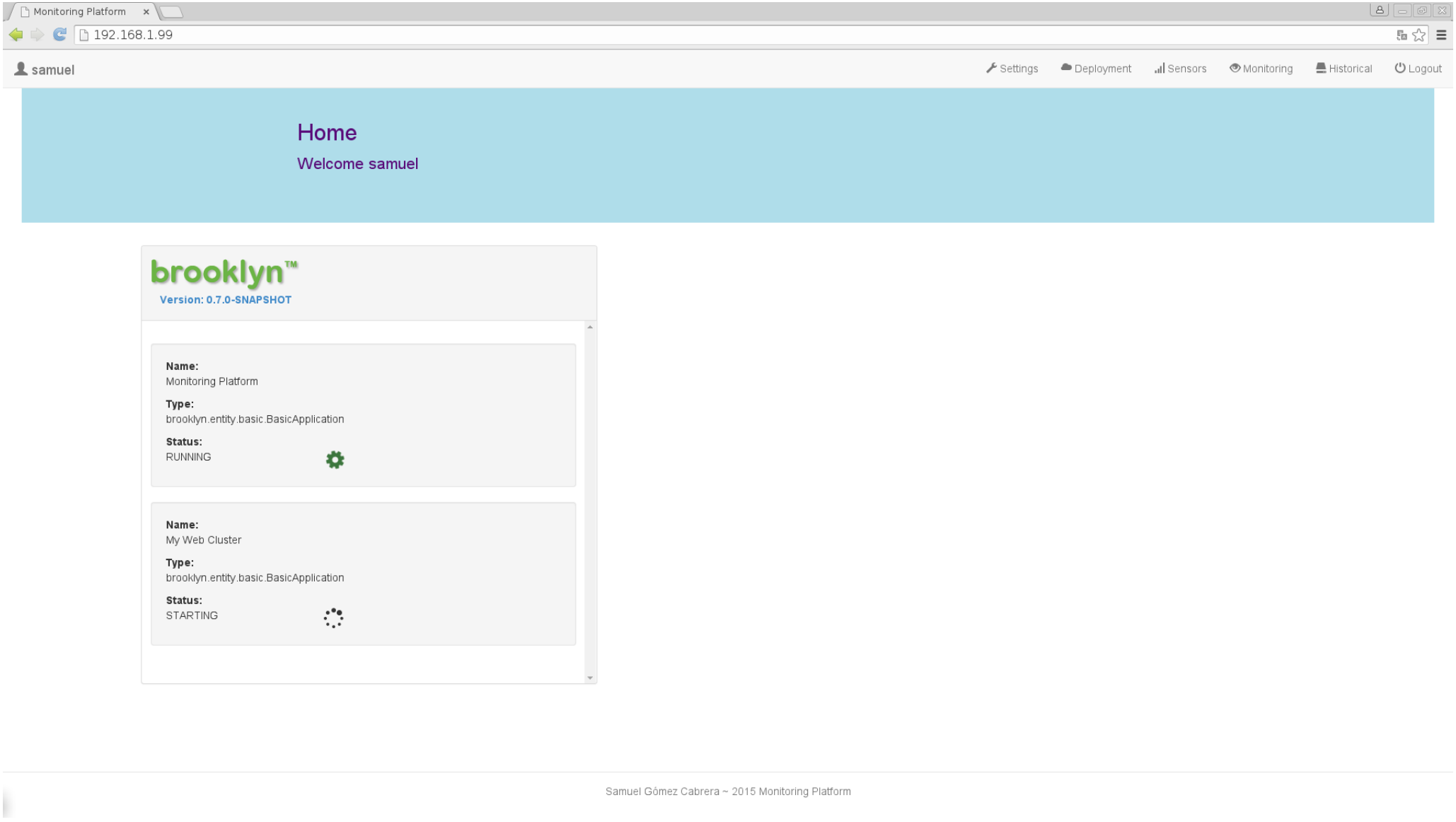


Figura 30: Información acerca del estado de los servicios desplegados

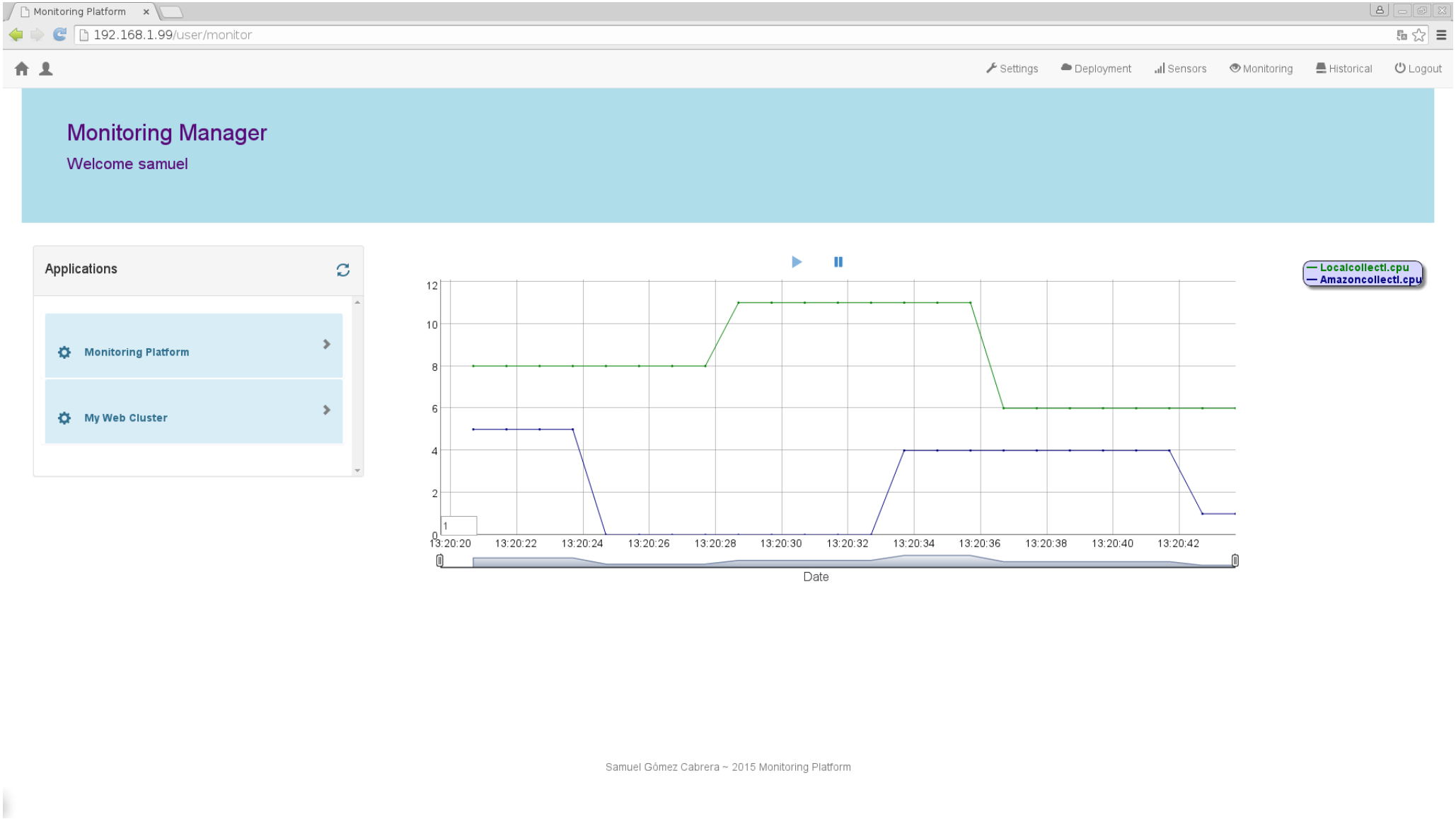


Figura 31: Monitorización de CPU en instancia de amazon y en instancia local

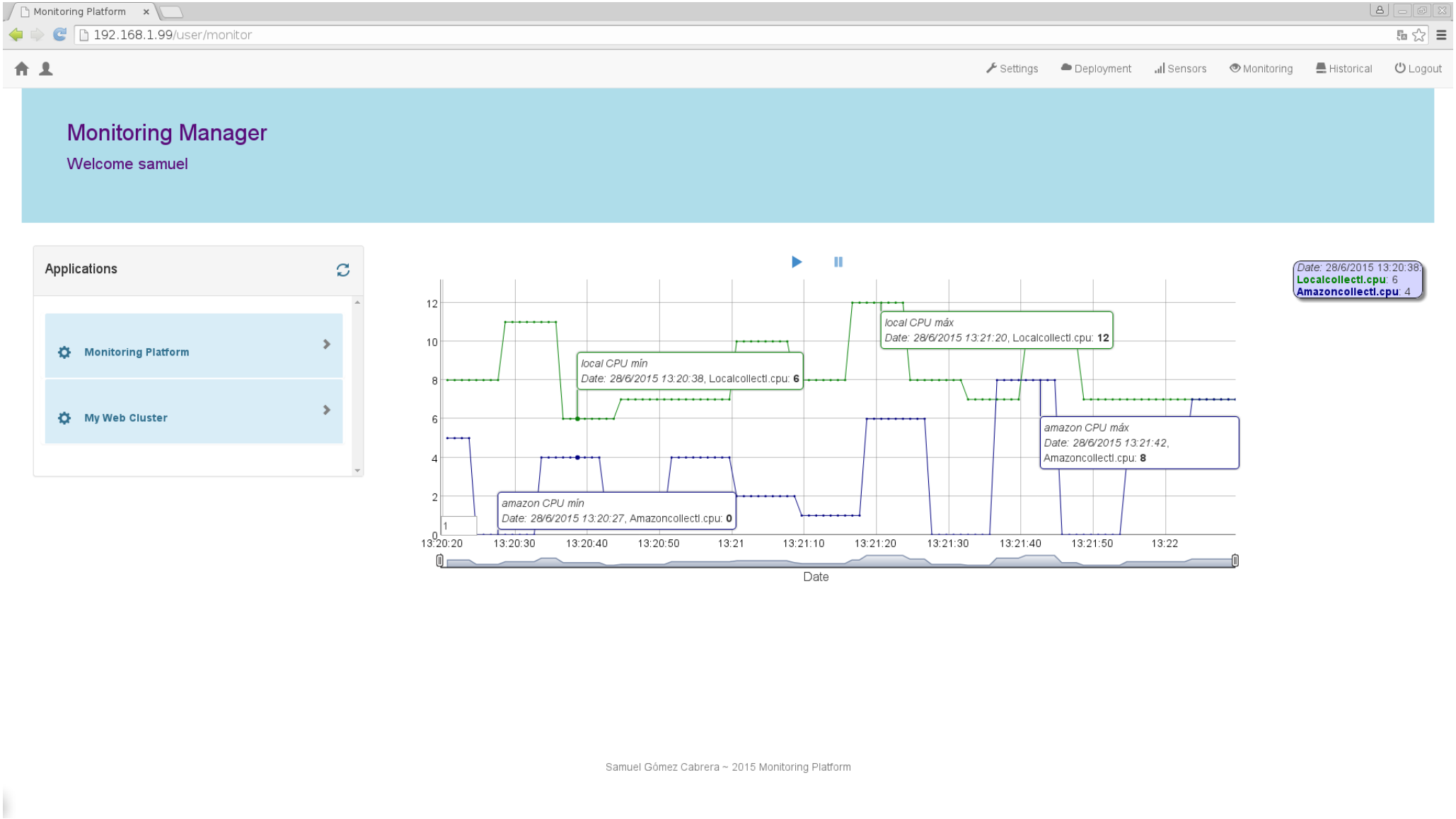


Figura 32: Monitorización de CPU en instancia de amazon y en instancia local, anotaciones relevantes

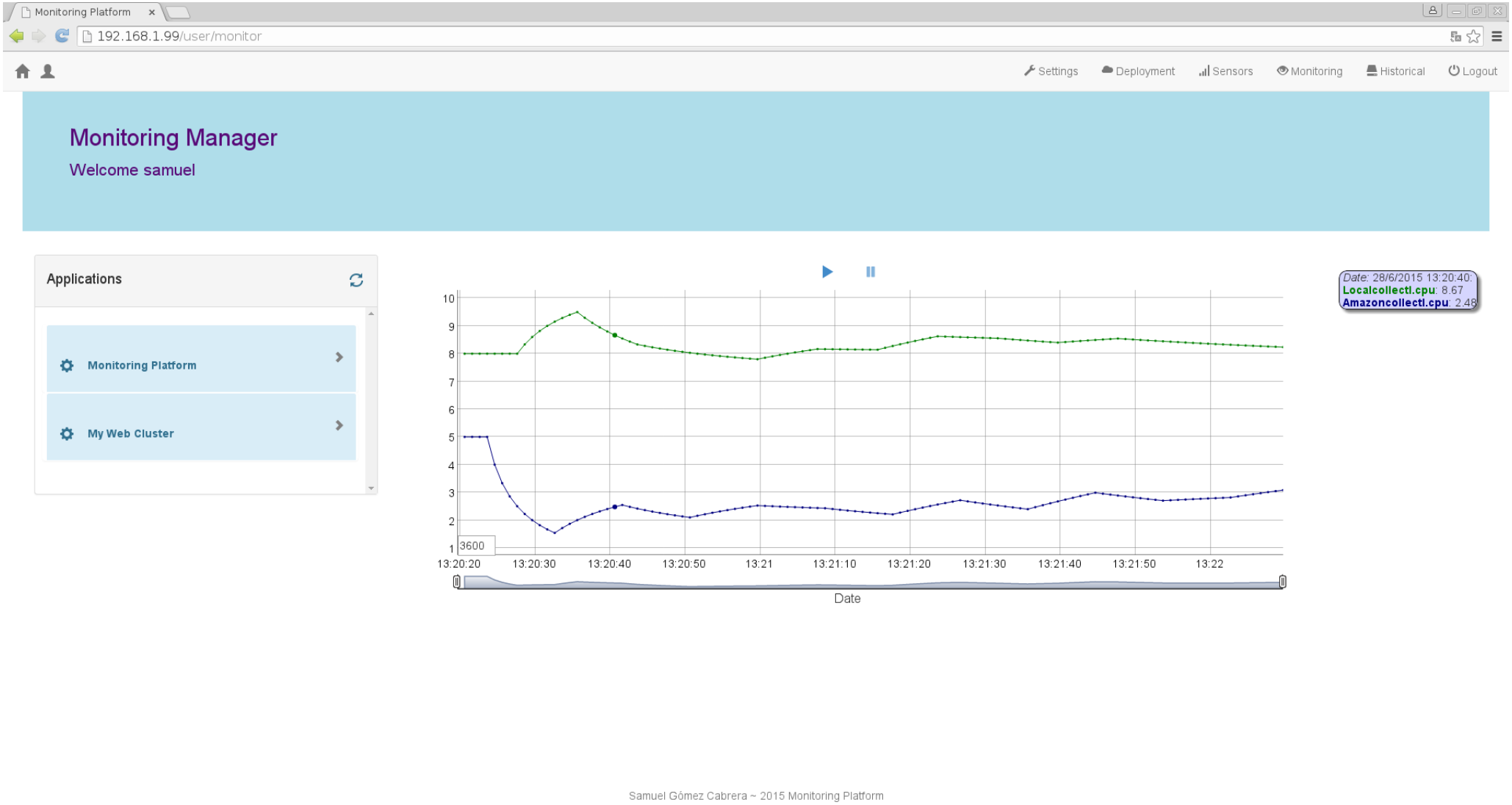


Figura 33: Monitorización de CPU en instancia de amazon y en instancia local, suavizado de valores

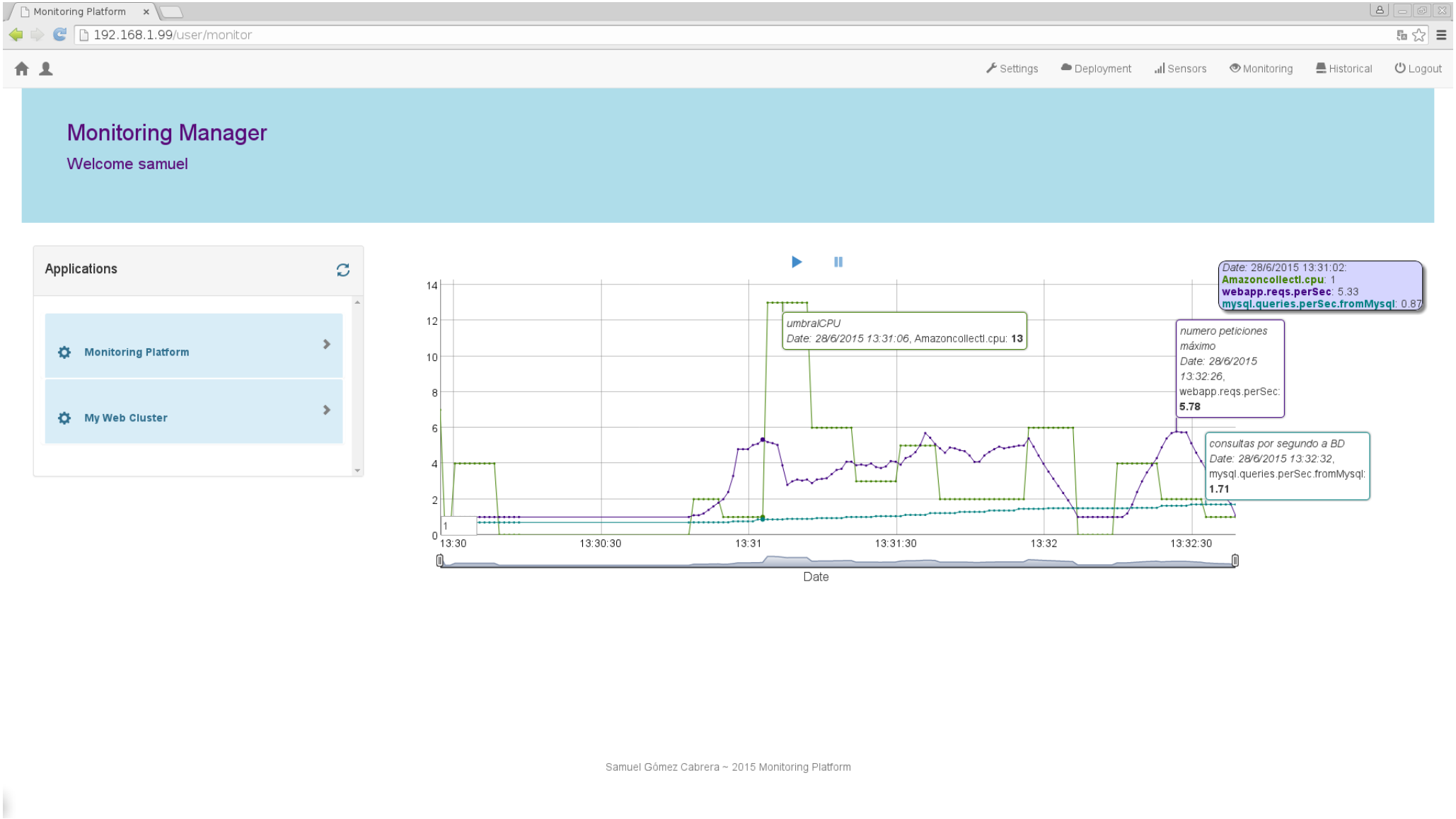


Figura 34: Monitorización de peticiones por segundo al servidor web, peticiones a la base de datos y tiempo de CPU de la instancia de amazon

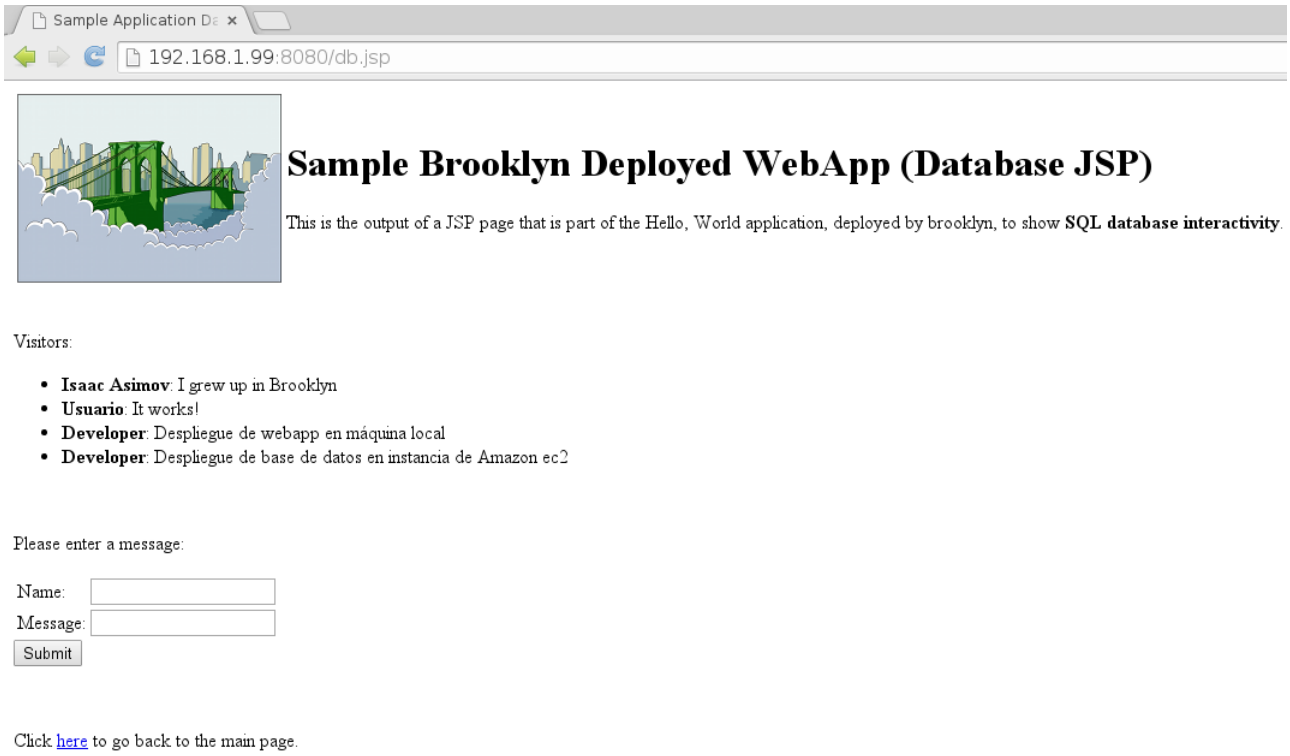


Figura 35: Interfaz de la aplicación web desplegada en la máquina local y base de datos en una instancia de Amazon

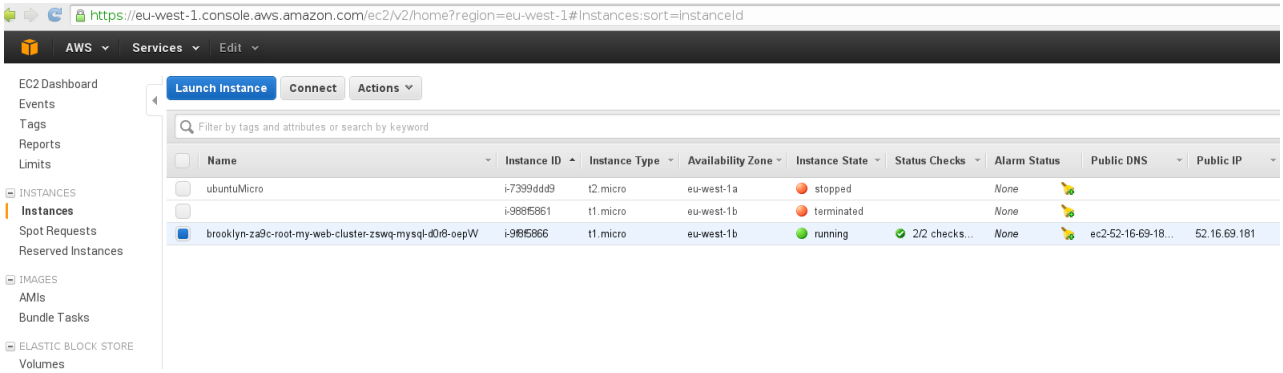


Figura 36: Interfaz de la consola de amazon ec2 donde se muestra la instancia desplegada que contiene la base de datos MySQL

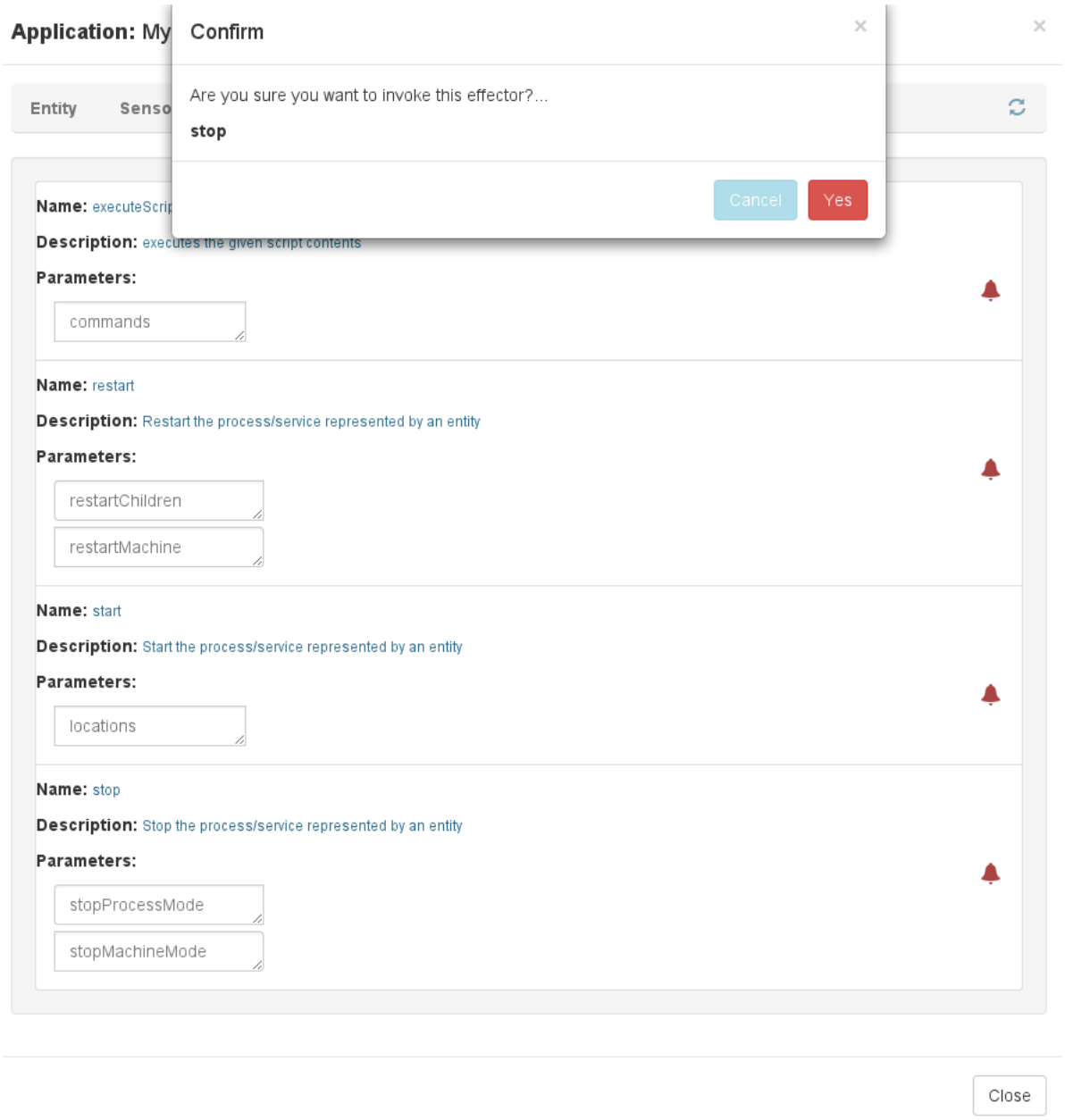


Figura 37: Interfaz de reconfiguración de My Web Cluster. Acción de parado del componente.

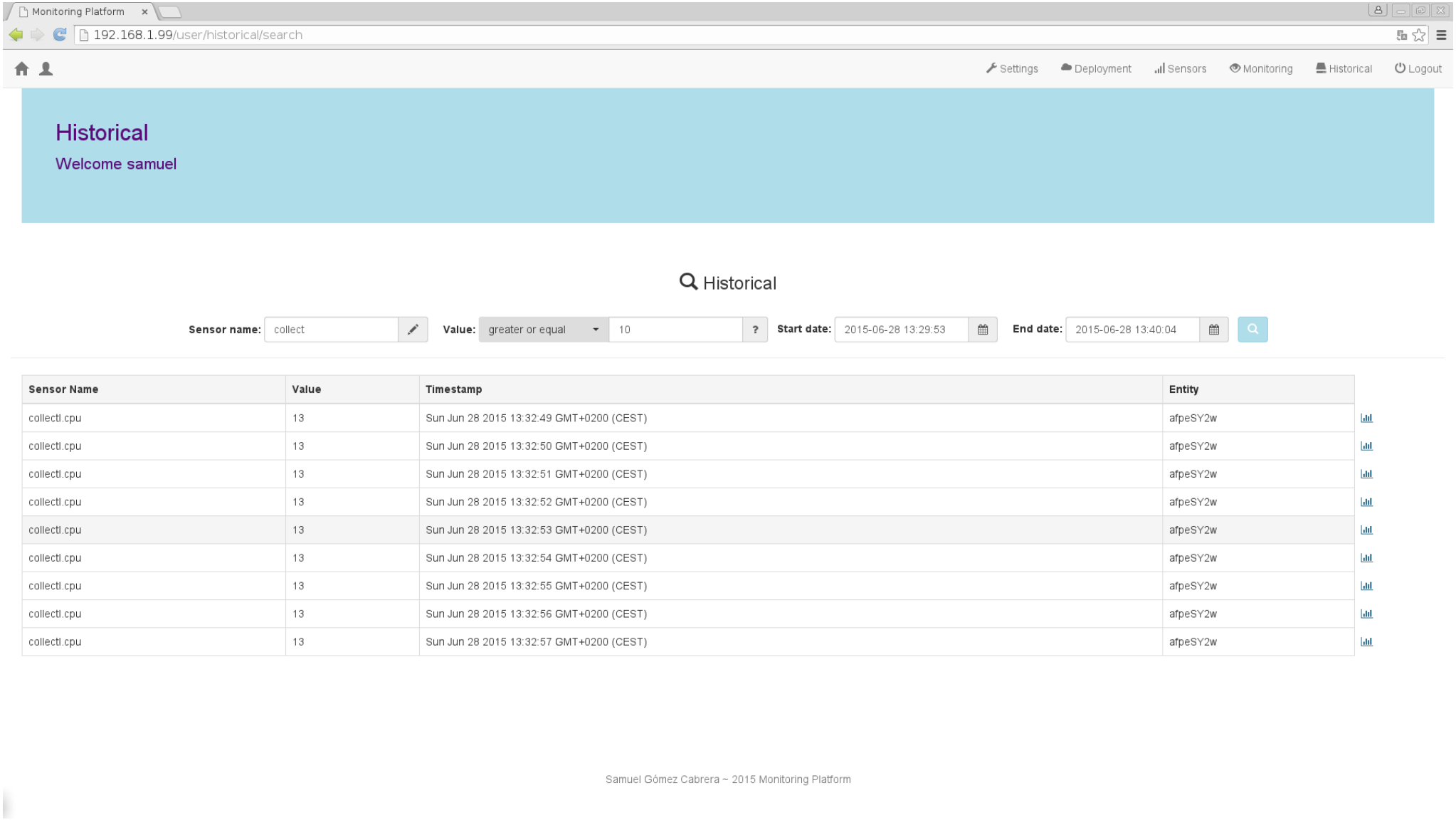


Figura 38: Consulta de datos en histórico sobre los valores recogidos de CPU

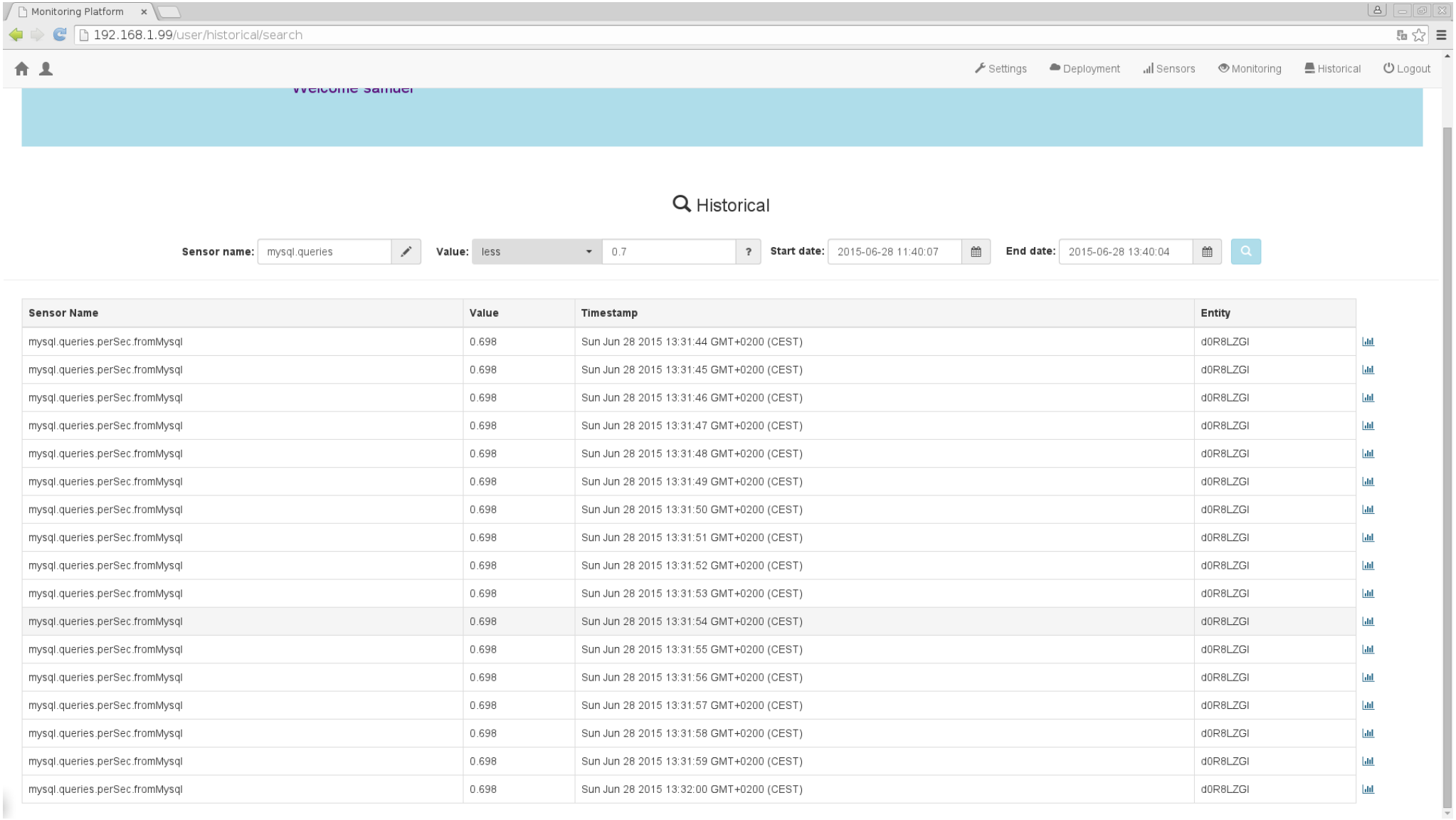
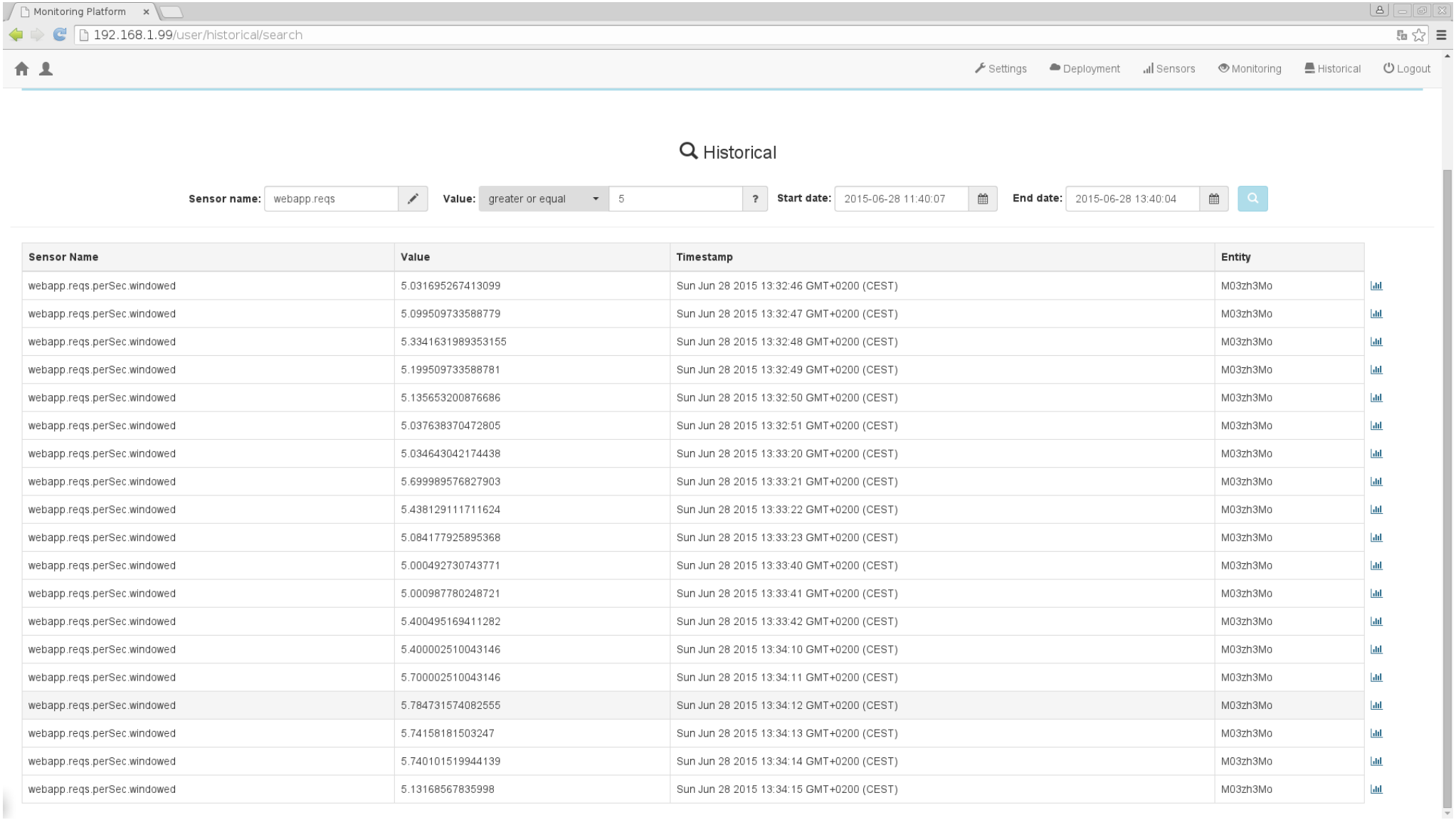


Figura 39: Consulta de datos en histórico sobre los valores recogidos de consultas a la base de datos



Sensor Name	Value	Timestamp	Entity
webapp.reqs.perSec.windowed	5.031695267413099	Sun Jun 28 2015 13:32:46 GMT+0200 (CEST)	M03zh3Mo
webapp.reqs.perSec.windowed	5.099509733588779	Sun Jun 28 2015 13:32:47 GMT+0200 (CEST)	M03zh3Mo
webapp.reqs.perSec.windowed	5.3341631989353155	Sun Jun 28 2015 13:32:48 GMT+0200 (CEST)	M03zh3Mo
webapp.reqs.perSec.windowed	5.199509733588781	Sun Jun 28 2015 13:32:49 GMT+0200 (CEST)	M03zh3Mo
webapp.reqs.perSec.windowed	5.135653200876686	Sun Jun 28 2015 13:32:50 GMT+0200 (CEST)	M03zh3Mo
webapp.reqs.perSec.windowed	5.037638370472805	Sun Jun 28 2015 13:32:51 GMT+0200 (CEST)	M03zh3Mo
webapp.reqs.perSec.windowed	5.034643042174438	Sun Jun 28 2015 13:33:20 GMT+0200 (CEST)	M03zh3Mo
webapp.reqs.perSec.windowed	5.699989576827903	Sun Jun 28 2015 13:33:21 GMT+0200 (CEST)	M03zh3Mo
webapp.reqs.perSec.windowed	5.438129111711624	Sun Jun 28 2015 13:33:22 GMT+0200 (CEST)	M03zh3Mo
webapp.reqs.perSec.windowed	5.084177925895368	Sun Jun 28 2015 13:33:23 GMT+0200 (CEST)	M03zh3Mo
webapp.reqs.perSec.windowed	5.000492730743771	Sun Jun 28 2015 13:33:40 GMT+0200 (CEST)	M03zh3Mo
webapp.reqs.perSec.windowed	5.000987780248721	Sun Jun 28 2015 13:33:41 GMT+0200 (CEST)	M03zh3Mo
webapp.reqs.perSec.windowed	5.400495169411282	Sun Jun 28 2015 13:33:42 GMT+0200 (CEST)	M03zh3Mo
webapp.reqs.perSec.windowed	5.400002510043146	Sun Jun 28 2015 13:34:10 GMT+0200 (CEST)	M03zh3Mo
webapp.reqs.perSec.windowed	5.700002510043146	Sun Jun 28 2015 13:34:11 GMT+0200 (CEST)	M03zh3Mo
webapp.reqs.perSec.windowed	5.784731574082555	Sun Jun 28 2015 13:34:12 GMT+0200 (CEST)	M03zh3Mo
webapp.reqs.perSec.windowed	5.74158181503247	Sun Jun 28 2015 13:34:13 GMT+0200 (CEST)	M03zh3Mo
webapp.reqs.perSec.windowed	5.740101519944139	Sun Jun 28 2015 13:34:14 GMT+0200 (CEST)	M03zh3Mo
webapp.reqs.perSec.windowed	5.13168567835998	Sun Jun 28 2015 13:34:15 GMT+0200 (CEST)	M03zh3Mo

Figura 40: Consulta de datos en histórico sobre los valores recogidos de peticiones al servidor web

5. Conclusiones

El carácter novedoso de la computación en la Nube hace que cualquier trabajo que se desarrolle en su ámbito tenga una carga importante de estudio e investigación. A lo largo de esta memoria se ha tratado de dar una visión lo más amplia posible del abanico de posibilidades, tecnologías y estudios sobre la computación en la Nube centrada principalmente en monitorización de aplicaciones distribuidas y gestión e interoperabilidad en entornos multi-cloud.

Dados los avances, oportunidades de negocio y oportunidades de investigación que ofrecen estos elementos no resulta fácil saber cuál será la línea futura. Lo que sí parece claro es que el uso de la Nube irá incorporándose paulatinamente con más fuerza en el mundo empresarial. El avance de las redes de comunicaciones y el grado de acogida a estándares por parte de los proveedores, van a determinar en qué medida las empresas y organizaciones pueden delegar sus servicios y confiar en la Nube.

Mediante el desarrollo de esta infraestructura, se ha proporcionado un mecanismo de monitorización y gestión de servicios distribuidos en un entorno multi-cloud. La solución adoptada para su implementación permite la adaptación a diferentes entornos que pueden cubrir un gran número de necesidades, que van desde el número de proveedores o tecnologías soportadas por la API de despliegue, la capacidad para definición nuevos sensores, reconfiguración de servicios, o la capacidad de análisis a través de las herramientas gráficas proporcionadas. Así, una empresa que desee desplegar sus servicios en la nube, puede lograr obtener una solución integrada a través de éste sistema ahorrando en el uso de recursos de computación pudiendo controlar y monitorizar el estado de sus sistemas.

El fruto de este trabajo pretende aportar una ventaja añadida al despliegue de aplicaciones y servicios en la Nube, y servir como base para el desarrollo de nuevas iniciativas que traten de acercar este nuevo paradigma de la Computación en la Nube como un servicio cercano y del cual se pueden obtener numerosos beneficios, que van más allá de los recursos económicos. El uso del sistema puede ser de gran ayuda para asistir al ciclo de vida del desarrollo y mantenimiento de software, usando la Nube como infraestructura para el despliegue de sus servicios a través de nuestra infraestructura pudiendo beneficiarse de la retroalimentación obtenida en base a análisis de los datos monitorizados, y disponer así de fundamentos robustos para verificar el cumplimiento de requisitos y apoyar la toma de decisiones.

La monitorización juega, y va a seguir jugando, un papel clave en este entorno. La pérdida de control sobre la infraestructura por parte del cliente debe equilibrarse correctamente con estas medidas de supervisión que permitan comprobar la QoS y ser una garantía de cumplimiento de los SLA. Este hecho cobra aún mayor relevancia cuanto más avanzamos en la pirámide de abstracción y menos sabemos sobre la infraestructura donde operan nuestros servicios. Los proveedores deben no sólo garantizar el cumplimento de los SLA, sino ofrecer interfaces para el acceso a servicios de monitorización con el objetivo de poder obtener valores reales y fiables. Todo ello con independencia de la capa servicio de la Nube en la que nos encontremos.

Lograr la confianza de las empresas y usuarios no vinculados al sector de las TIC's, es otro motivo para el fomento de este tipo de medidas. Fortalecer esta confianza va a pasar no solo por asegurar con garantías las condiciones de los servicios, sino también por poner a disposición de los usuarios herramientas de alto nivel accesibles a cualquier tipo de público. De esta forma, serán capaces de verificar, gestionar y comprobar el correcto funcionamiento de sus sistemas.

5.1. Trabajos futuros

El enfoque adoptado para el desarrollo de la infraestructura, hace que se encuentre en un grado de madurez en el cuál es posible su explotación en la actualidad. Tanto como elemento para el despliegue y gestión, como herramienta para monitorización de aplicaciones y servicios en la Nube, puede proporcionar una solución integrada.

Además, el hecho de contar con tecnologías en pleno auge como Brooklyn, jclouds, Nodejs, o MongoDB, hace que con un mantenimiento evolutivo las mejoras que se implementaran en estas tecnologías se vieran reflejadas en un aumento de las capacidades y rendimiento de nuestro sistema.

Lograr una mayor abstracción en el diseño, podría ser una posible área en la que mejorar el sistema y así acercarlo a un público menos especializado. De igual forma, su integración con otras herramientas existentes en el desarrollo de aplicaciones, tales como IDEs supondría un gran avance y una forma de potenciar su uso.

Otra posible forma de mejorar su potencial, sería mediante la creación de componentes para predecir los cambios en los servicios desplegados a través de los datos recopilados. En este sentido, un área aún por explotar y con mucho futuro como la inteligencia empresarial o business intelligence, es un marco de trabajo que resulta muy interesante y en el que podrían converger sistemas monitorización y análisis formando una estructura completa que aporte "inteligencia" al negocio.

Finalmente, también se debe mencionar que la creación e inserción de nuevos sensores en nuestro sistema, se puede traducir en un aumento de las prestaciones del sistema, disponiendo así de un mayor número de métricas disponibles que pueden dar cobertura a los requisitos de los potenciales usuarios.

Referencias

[1] Real Academia Española. (2014). Diccionario de la lengua española (23.a ed.). Consultado en http://www.rae.es/

[2] HarperCollins. (2011). Collins English Dictionary. Consultado en http://www.collinsdictionary.com/

[3] Cambridge University Press. (2015). Cambridge dictionaries online. Consultado en http://dictionary.cambridge.org

[4] J.McCarthy, "Reminiscences on the History of Time Sharing", Standford University, 1983.

[5] R. Buyya, et al., Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility, Future Generation Computer Systems, 2009.

[6] Wikipedia, Virtual Storage Personal Computing. Wikimedia Foundation, Inc. 21 January 2008.

[7] Qusay F. Hassan, "Desmystifying Cloud Computing", Faculty of Computers and Information, Mansoura University, Egypt , 2011.

[8] P. Mell and T. Grance, The NIST Definition of Cloud Computing, Special Publication 800-145, National Institute of Standards and Technology, Gaithersburg, Maryland, 2011.

[9] Srinivas Rao V,Nageswara Rao N K ,E Kusum Kumari, Cloud Computing : An Overview, JATIT, Nov 2009, www.jatit.org

[10] Toby Velte, Anthony Velte, Robert Elsenpeter, Cloud Computing, A Practical Approach, McGraw-Hill, Inc., New York, NY, 2009

[11] Open source software for creating private and public clouds. https://www.openstack.org/

[12] NYSE Technologies Launches Global CloudBased 'Capital Markets Community Platform" http://www.vmware.com/files/pdf/customers/VMware-NYSE-Technologies-12Q2-EN-Case-Study.pdf

[13] Dimitris Zeginis, Francesco D'Andria, Stefano Bocconi, Jesus Gorronogoitia Cruz, Oriol Collell Martin, Panagiotis Gouvas, Giannis Ledakis, Konstantinos, A. Tarabanis. A user-centric multi-PaaS application management solution for hybrid multi-Cloud scenarios. *Scalable Computing: Practice and Experience*, 2013.

[14] N. Grozev, R. Buyya, "Inter-Cloud architectures and application brokering: Taxonomy and survey", *Software Practice and Experience*, vol. 44, pp. 369–390, 2014.

[15] B. Kezia Rani, B. Padmaja Rani, Dr., A. Vinaya Babu, Dr., "Cloud Computing and Inter-Clouds – Types, Topologies and Research Issues", *Procedia Computer Science*, vol. 50, pp. 24–29, 2015.

[16] Contrail Project - A European Cloud Federation, IaaS, PaaS project. http://contrail-project.eu/

[17] Optimis Project - Optimized Infrastructure Services. http://www.optimis-project.eu/

[18] mOSAIC Project - Open-Source API and Platform for Multiple Clouds. http://www.mosaic-project.eu/

[19] Apache Stratos - Open source polyglot Platform as a Service (PaaS) framework. http://stratos.apache.org/

[20] OpenTOSCA - Open Source TOSCA Ecosystem. http://www.iaas.uni-stuttgart.de/OpenTOSCA/

[21] MODAClouds - MOdel-Driven Approach for design and execution of applications on multiple Clouds. http://www.modaclouds.eu/

[22] SeaClouds - Seamless adaptive multi-cloud management of service applications. http://www.seaclouds-project.eu/

[23] Apache Brooklyn open source project. http://brooklyn.incubator.apache.org/

[24] Apache jclouds - The Java Multi-Cloud Toolkit. http://jclouds.apache.org/

[25] Apache Libcloud - Python library for interacting with many of the popular cloud service providers using a unified API. https://libcloud.apache.org/

[26] Apache DeltaCloud https://deltacloud.apache.org/

[27] SimpleCloud API https://en.wikipedia.org/wiki/Simple_Cloud_API

[28] Apache Nuvem http://wiki.apache.org/incubator/Nuvem

[29] Zend_Cloud http://framework.zend.com/manual/1.12/en/zend.cloud.html

[30] K. Oberle, G. Gallizo, R. Kuebert, E. Oliveros, Enhancing the SLA Framework of a Virtualized Service Platform by dynamic renegotiation, eChallenges2010, Warsaw, Poland, October 2010.

[31] Cloud Computing Use Cases Group, Cloud computing use cases white paper, July 2010 http://www.cloud-council.org/Cloud_Computing_Use_Cases_Whitepaper-4_0.pdf

[32] D Ardagna, G Casale, M Ciavotta, JF Pérez, W Wang, Quality-of-service in cloud computing: modeling techniques and their applications, *Journal of Internet Services and Applications*, 5(1), 1-17, 2014.

[33] Jeffrey J.P.Tsai, Steve J.H.Yan, Monitoring and Debugging of Distributed and Real-Time Systems, *IEEE Computer Society Press*, 1995.

[34] M. Zuñiga-Prieto, P. Cedillo, J. González-Huerta, E. Insfrán, and S. Abrahão, "Monitoring Services Quality in the Cloud", *ERCIM News 201*4, Vol. 99, Special Theme on Software Quality, 2014.

[35] Giuseppe Cicotti , Luigi Coppolino , Rosario Cristaldi , Salvatore DÁntonio , Luigi Romano, QoS monitoring in a cloud services environment: the SRT-15 approach, Euro-Par Workshops (1), volume 7155 of *Lecture Notes in Computer Science*, page 15-24. Springer, 2011.

[36] VC Emeakaroha, I Brandic, M Maurer, S Dustdar, Low Level Metrics to High Level SLAs-LoM2HiS framework: Bridging the gap between monitored metrics and SLA parameters in Cloud environments, The 2010 High Performance Computing and Simulation Conference (HPCS), 2010.

[37] Amazon CloudWatch http://aws.amazon.com/es/cloudwatch/

[38] AzureWatch https://www.paraleap.com/AzureWatch

[39] CloudKick https://en.wikipedia.org/wiki/Cloudkick

[40] Google Cloud Monitoring https://cloud.google.com/monitoring/

[41] HP Cloud Monitoring http://www.hpcloud.com/products-services/monitoring

[42] Zenoss Cloud http://www.zenoss.com/solution/cloud-monitoring

[43] vRealize Hyperic http://www.vmware.com/es/products/vrealize-hyperic

[44] Ganglia Monitoring System http://ganglia.sourceforge.net/

[45] Nagios https://www.nagios.com/products

[46] collectl http://collectl.sourceforge.net/

[47] collectd https://collectd.org/

[48] OpenNMS http://www.opennms.org/

[49] Zabbix http://www.zabbix.com/

[50] WebStorm https://www.jetbrains.com/webstorm/

[51] Git http://git-scm.com/

[52] Bitbucket https://bitbucket.org/

[53] Nodejs https://Nodejs.org/

[54] Express http://expressjs.com/

[55] MongoDB https://www.mongodb.org//

[56] Dygraphs http://dygraphs.com/

[57] JQuery https://jquery.com/

[58] Bootstrap http://getbootstrap.com/

[59] YAML: YAML Ain't Markup Language http://yaml.org/

[60] Cloud Application Management for Platforms Version 1.1 Committee Specification 01, OASIS, 2014,http://docs.oasis-open.org/camp/camp-spec/v1.1/camp-spec-v1.1.html

Anexo I. Descripción técnica del equipo servidor

```
Información del equipo:


   Procesador : 4x Intel(R) Core(TM) i3-4130 CPU @ 3.40GHz
   Memoria    : 3928MB
   Disco      :

      Familia:          Seagate Barracuda 7200.14 (AF)
      Modelo:           ST2000DM001-1CH164
      Capacidad:        2,00 TB
      Verión SATA:      SATA 3.1, 6.0 Gb/s (current: 6.0 Gb/s)


Información del sistema operativo:

   ID Distribuidor: Debian
   Descripción:    Debian GNU/Linux 8.1 (jessie)
   Release:        8.1
   Codename:       jessie
   Detalles del Kernel:
   Linux servidor 3.16.0-4-amd64 #1 SMP Debian 3.16.7-ckt11-1 (2015-05-24)
   x86_64 GNU/Linux
```

Anexo II. Instrucciones de instalación

Para la puesta en producción de la infraestructura será necesario en primer lugar la instalación de los siguientes elementos:

- Apache Brooklyn
- Nodejs
- MongoDB

Se detallarán los pasos necesarios para la instalación en distribuciones linux basadas en Debian.

Instalación de Apache Brooklyn

Para la instalación y configuración, y puesta en marcha de Apache Brooklyn, se puede consultar la página web del proyecto [23], en la que se detalla de forma adecuada los pasos necesarios.

Nodejs

Para la instalación de Nodejs, abre una consola y ejecuta los siguientes comandos:

```
sudo apt-get install python-software-properties
sudo apt-get update
sudo apt-get install nodejs npm
```

MongoDB

Para la instalación de MongoDB, abre una consola y ejecuta los siguientes comandos:

```
curl -O https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-3.0.4.tgz
tar -zxvf mongodb-linux-x86_64-3.0.4.tgz
cd /opt
mkdir -p mongodb
cp -R -n mongodb-linux-x86_64-3.0.4/ mongodb
echo "export PATH=/opt/mongodb/bin:$PATH" >> /home/<usuario>/.bashrc
source /home/<usuario>/.bashrc
#crear directorio de datos
mkdir -p /data/db
#iniciar mongo
mongod
```

Tras esto, descomprimimos el archivo de la aplicación en cualquier directorio de y debemos importar la base de datos antes de arrancar la aplicación. Estos pasos los realizamos de la siguiente forma:

```
tar -zxvf monitoring-platform.tar.gz
cd db/
#restauramos los datos de base de datos
mongorestore --dbpath /var/lib/mongo --db mp_usr mp_usr/
cd ../monitoring-platform/
#arrancamos la app nodejs
nodejs bin/www
```

Finalmente podremos acceder a nuestra aplicación a través del navegador en

http://localhost:3000