UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Mecanismos de visualización para la monitorización de servicios cloud

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Oleksandr Kilaru

Tutor: Silvia Abrahao Gonzales

Curso 2017-2018

Resumen

Actualmente existe la necesidad de mecanismos de monitorización flexibles que permitan tanto al cliente como al proveedor evaluar la calidad de los servicios ofertados en la nube con el fin de ofrecer una adecuada provisión de los mismos. El grupo de investigación ISSI del Departamento de Sistemas Informáticos y Computación (DSIC) de la UPV está trabajando en una infraestructura de monitorización de servicios cloud que utiliza modelos en tiempo de ejecución(models@runtime) para evaluar la calidad de los servicios, detectar incumplimientos en los acuerdos a nivel de servicio (SLA) y reconfigurar la arquitectura de los servicios dinámicamente para cumplir con los niveles de calidad esperados. El objetivo de este proyecto es diseñar e implementar un prototipo de herramienta de visualización de datos de monitorización que apoye la infraestructura de monitorización permitiendo a los usuarios crear dinámicamente, reordenar, buscar y explorar múltiples representaciones de datos de medición de calidad de los servicios de diversas fuentes de datos. Se presenta el diseño de la herramienta, su implementación y su aplicación a un escenario de uso con datos de monitorización obtenidos de un repositorio.

Palabras clave: Cloud Computing, Calidad de Servicios, Acuerdo a Nivel de Servicio, models@runtime, Visualización de Datos de Monitorización.

Abstract

Nowadays there is a need of flexible monitoring mechanisms that allow both the customer and the provider to evaluate the quality of cloud services in order to offer an adequate provision of them. The ISSI research group at the Department of Computer Systems and Computation (DSIC) from the UPV is working on a monitoring infrastructure for cloud services that explores the use of models at runtime (models@runtime) to support the quality evaluation of the services, the detection of Service Level Agreement (SLA) violations, and the dynamic reconfiguration of the service architecture in order to accomplish with the expected quality levels. The main objective of this project is the design and implementation of a prototype of a data visualization tool that supports the monitoring infrastructure allowing the users to dynamically create, reorder, search and explore multiple representations of services measurement data from various data sources. We present the design and the implementation of the tool as well as its application on a usage scenario with monitoring data obtained from a repository.

Keywords: Cloud Computing, Quality of Service, Service Level Agreement, models@runtime, Monitoring, Data Visualization.

Tabla de contenidos

Índice de tablas

Índice de figuras

1. Introducción

Este trabajo tiene a fin presentar el trabajo de desarrollo de una herramienta de visualización de datos de monitorización de servicios desplegados en plataformas cloud. A lo largo de este documento se expondrán los conceptos más relevantes para el proyecto, el desarrollo del prototipo y su aplicación a un caso práctico de monitorización de servicios.

1.1 Motivación

Actualmente los servicios en la nube representan una nueva corriente del desarrollo del software. Se trata de una tecnología que cobra cada vez más importancia en las áreas más relevantes de las empresas tecnológicas [2]. En la actualidad las plataformas en la nube agrupan sus servicios en distintos tipos: Infraestructura como Servicio (*Infrastructure as a Service, Iaas*), Plataforma como Servicio (*Platform as a Service, PaaS*), o Software como Servicio (*Software as a Service, SaaS*)

El modelo de negocio ofrecido por los servicios en la nube presenta una serie de ventajas tanto para proveedores como para los consumidores. Algunas de las ventajas son: gran escalabilidad ofrecida por el servicio, disponibilidad, elasticidad, un modelo de pago por uso, mejor aprovechamiento de los recursos, recuperación rápida en caso de avería, etc.

En este trabajo, nos centraremos en el modelo de servicio SaaS. SaaS ofrece una solución integral de software en la cual los datos se alojan en los servidores de las compañías TIC, a los cuales se accede mediante internet. La empresa TIC se ocupa del mantenimiento, soporte al cliente y garantiza la disponibilidad y la seguridad de la aplicación y de sus datos.

Un servicio SaaS, en ocasiones tiene un número de usuarios con picos de tráfico muy alto, por lo tanto, se debe soportar la carga de usuarios con un rendimiento escalable, adaptando sus recursos a las necesidades de los usuarios en el momento. Para garantizar el cumplimiento de los atributos de calidad de un servicio (disponibilidad, seguridad, escalabilidad, rendimiento, etc.), se crea un Acuerdo de Nivel de Servicio (*Service Level Agreement,* SLA) entre ambas partes. Un SLA es un contrato entre el proveedor del servicio y el usuario (consumidor del servicio). En este contrato se estipulan las garantías mínimas que un proveedor de servicios les ofrece a sus clientes. Habitualmente

el proveedor de servicios cloud ofrece algunas garantías de rendimiento para servicios de cómputos y deja la detección de violaciones del SLA para el cliente. Además, pocos proveedores de servicios recompensan a sus clientes de forma automática por esas violaciones, por lo tanto, el cliente debe demostrar, con evidencias, que se ha producido una violación del SLA.

De esta forma, es de gran utilidad tanto para el cliente como para el proveedor, que se disponga de una herramienta que permita conocer el comportamiento del servicio en tiempo real y comprobar si se cumple el acuerdo SLA acordado, para que en caso de que no se cumpla, poder aportar las evidencias necesarias para la reclamación. También es interesante por parte del proveedor ya que puede dar garantías de uso a sus clientes.

Por todo ello, es necesario una herramienta de monitorización que permita medir los atributos de calidad del servicio en la nube y generar informes de incumplimientos del SLA. Para solucionar este problema, Cedillo et al. [1] plantea el uso de los modelos en tiempo de ejecución (*models@runtime*)[1]. Estos modelos tienen el potencial de ser utilizados en tiempo de ejecución para supervisar y verificar aspectos particulares de los servicios cloud en tiempo de ejecución.

Esta tecnología especifica los datos que debe recoger la herramienta de monitorización, los cuales estarán basados en el SLA y en otros requisitos adicionales. Este modelo se produce de acuerdo a un modelo de calidad SaaS[2], que recoge las características, atributos y métricas que permitirán medir los atributos de calidad de los servicios en la nube. Utilizando el modelo de calidad SaaS junto al SLA, se puede crear un modelo con las características para la medición de los acuerdos y una estructura para su medición, el cual estará reflejado en el modelo en tiempo de ejecución (*models@runtime*).

Esta aproximación de monitorización de servicios cloud basada en modelos en tiempo de ejecución, así como la infraestructura software que le soporta se ha propuesto en trabajos anteriores [20]. Aunque se ha propuesto un Configurador y un Middleware de Monitorización de servicios cloud para las plataformas Microsoft Azure [20] y Google AppEngine [21], la infraestructura

[1] Un modelo en tiempo ejecución es una abstracción de un sistema ejecutándose que puede ser manipulado en tiempo de ejecución para un propósito en particular (Bencomo *et al.*, 2013).

[2] Un modelo de calidad SaaS describe las características de un producto software, como se relacionan, como pueden ser medidas y que interpretación se puede dar a estas medidas.

carece de un interfaz que permita a los usuarios visualizar los datos de monitorización y generar informes.

1.2 Objetivos

El propósito de este trabajo es diseñar e implementar un prototipo de herramienta de visualización de datos de monitorización que apoye la infraestructura de monitorización definida por Jiménez [20] permitiendo a los usuarios crear dinámicamente, reordenar, buscar y explorar múltiples representaciones de datos de medición de calidad de los servicios de diversas fuentes de datos.

Dicho objetivo general se descompone en los siguientes objetivos específicos:

- Creación de una herramienta de visualización capaz de leer el modelo en tiempo de ejecución (representado como un fichero xmi) procedente del Configurador y mostrar al usuario de forma gráfica los resultados obtenidos por la herramienta de monitorización de Jiménez [20].
- Permitir al usuario seleccionar qué atributos de calidad quiere visualizar y habilitar o deshabilitar el control de la violación del SLA.
- Permitir al usuario filtrar por fecha los valores a ser visualizados y seleccionar el tipo de gráfica a mostrar.
- Permitir al usuario generar informes en PDF en caso de que se produzca una violación del SLA.

Por lo tanto, la herramienta de visualización pretende facilitar el seguimiento y la toma de decisión por parte del usuario de aquellos atributos de calidad que desea monitorizar (ej., disponibilidad, rendimiento, tiempo de respuesta, etc.). En primer lugar, mostrando la información en diferentes formatos y mediante distintas gráficas interactivas dentro de un *dashboard*.

Además, también permitirá la generación de informes donde se reflejen las violaciones del acuerdo de nivel de servicio (SLA), el cual puede servir tanto para el cliente como para el proveedor de servicios como un referente claro de cómo los servicios están siendo provistos.

1.3 Contexto

En este trabajo se trata de definir e implementar una herramienta de visualización de datos de servicios cloud basada en el Trabajo de Fin de Grado de Jiménez [20] el cual define un configurador y un middleware de monitorización de servicios cloud para la plataforma Microsoft Azure.

La herramienta de visualización a desarrollar establecerá la conexión con la herramienta de monitorización de Jiménez mediante la lectura del modelo en tiempo de ejecución (fichero xmi). De esta forma, el modelo en tiempo de ejecución, que es la salida de la herramienta de Jimenez, es la entrada para la herramienta de visualización.

Asimismo, este trabajo forma parte del proyecto de investigación "Desarrollo Incremental de Servicios Cloud Dirigido por Modelos y Orientado al Valor del Cliente (Value@Cloud)" realizado por el grupo de investigación de Ingeniería del Software y Sistemas de Información (ISSI) del Departamento de Sistemas Informáticos y Computación (DSIC) y del Departamento de Organización de Empresas (DOE) de la Universitat Politécnica de Valencia. En particular, el trabajo desarrollado contribuye al paquete de trabajo "Definición y gestión de mecanismos de monitorización del valor en tiempo de ejecución".

1.4 Estructura del documento

En el primer capítulo se ha presentado la introducción al trabajo, se ha presentado la motivación, los objetivos que se pretende alcanzar y el contexto del trabajo. Los siguientes capítulos se estructuran de la siguiente forma:

En el Capítulo 2 se presenta el estado del arte acerca de los métodos relacionados con este trabajo, así como la evolución de las técnicas de monitorización de servicios cloud.

El Capítulo 3 presenta el contexto de este trabajo referente a *cloud computing.* Se definen los conceptos principales de la computación en la nube, los servicios que se ofrecen y se presenta el documento SLA que define las relaciones entre consumidores y proveedores de un servicio.

En el Capítulo 4 se presenta los antecedentes a este trabajo: el Configurador y el Middleware de monitorización independiente de la plataforma propuesto por Jiménez [20], presentando sus componentes, funciones y la relación que tienen entre ellos, así como la contribución de este proyecto que consiste en la implementación de una herramienta de visualización de datos de monitorización que se integre con las herramientas propuestas por Jiménez.

En el Capítulo 5 se presenta la implementación de la herramienta de monitorización.

En el Capítulo 6 se presenta el caso de estudio empleado para demonstrar el funcionamiento de la herramienta

Finalmente, en el Capítulo 7 se presentan las conclusiones generales, los problemas enfrentados en el desarrollo y los futuros trabajos.

2. Análisis del estado del arte

En esta sección se presentarán las tecnologías de monitorización halladas en la literatura. En primer lugar, se introducirá la monitorización de servicios en la nube. Posteriormente, se pondrá en contexto la monitorización de servicios en la nube a través de sus orígenes en la monitorización de sistemas distribuidos. Tras ello se procederá a exponer los distintos acercamientos actuales al problema encontrados en la literatura. Por último, se procederá a hacer una comparación entra las distintas herramientas de monitorización y se comentarán sus ventajas y desventajas.

2.1 Introducción a la monitorización en la nube

Actualmente la computación en la nube ha ido cobrando cada vez más importancia debido a que hoy en día se dispone de una conexión a internet de banda ancha y servidores rápidos y baratos que permiten acceder a los contenidos almacenados en la nube de forma rápida y segura. Utilizar estos servicios introduce un rango amplio de posibilidades para el ahorro de costos, mayor rendimiento y mayor seguridad de los datos. Para administrar estos servicios y hacer posible el cumplimiento de los términos establecidos en el SLA es necesario hacer el uso de una herramienta de monitorización para obtener información sobre el rendimiento y la calidad de los servicios desplegados.

Estas herramientas son esenciales para los clientes y los proveedores de servicios en la nube ya que, al utilizar el método de pago por uso, es necesario disponer de datos de rendimiento precisos para poder realizar la facturación de forma correcta. La monitorización es una parte importante para los procesos de provisión de recursos/servicios, planificación de la calidad, gestión de la configuración, comprobación del SLA, facturación y seguridad y privacidad.

Para estos fines hay una gran cantidad de herramientas disponibles las cuales utiliza distintos enfoques hacía la monitorización. Estas herramientas se clasifican según el tipo de datos que extraen y de cómo los obtienen.

2.1.1 Tipos de datos

Podemos clasificar los datos del servicio cloud en tres capas, relacionadas con las tres formas de provisión de los servicios cloud: IaaS, PassS y SaaS. La Infraestructura como Servicio (*Infrastructure as a Service*) se sitúa en la capa inferior y engloba la capacidad de computación de los servidores, estructuras de almacenamiento y elementos de conectividad y seguridad. Un ejemplo de IaaS más conocido es el Amazon Web Service (AWS) que nos provee una serie de servicios como EC2 que nos permite manejar máquinas virtuales o en la nube o S3 para usarlo como almacenamiento.

Una herramienta de monitorización que actúe sobre esta capa nos proporcionaría los datos sobre el uso de la CPU, el estado de la red, uso de los espacios de almacenamiento, uso de la memoria principal, etc.

En la capa del medio se encuentra la Plataforma como Servicio (*Platform as a Service)* la cual proporciona una combinación de hardware y software que requieren los desarrolladores para poner en marcha sus aplicaciones. Normalmente hay que seguir una serie de restricciones para poder desarrollar para un proveedor. Por ejemplo, para Google App Engine es necesario desarrollar en el lenguaje Java o Python para poder desplegar las aplicaciones en la infraestructura. En el caso de la plataforma Microsoft Azure se podrían utilizar los lenguajes C# y Visual Basic mediante el .NET Framewok, o implementar en C++, Java y otros lenguajes sin .NET.

Una herramienta de monitorización que actúe sobre esta capa nos proporcionaría la información sobre el número de peticiones realizadas, escrituras en el disco, espacio requerido, tiempo de procesamiento, etc.

Por otra parte, los datos de SaaS serán referentes a la implementación concreta del servicio y varían dependiendo del dominio del servicio y con la tecnología que se ha implementado. Por ejemplo, en un servicio de venta de coches se podrían obtener los datos sobre las ventas mensuales en cada concesionario, o en un servicio de *streaming online* se podría obtener un listado de películas o series más vistas.

2.2 Monitorización en sistemas distribuidos tradicionales

Las herramientas de monitorización de servicios en la nube y la computación en la nube tienen un origen común, los sistemas distribuidos de *clusters* y *grid.*

La computación en la nube se diferencia de un *cluster* en que el sistema es considerado como un conjunto de varios servidores que se construyen e instalan para trabajar como si fuesen uno solo y todos desarrollan la misma tarea. En cambio, en la computación en la nube, en cada nodo hay diferentes máquinas virtuales que comparten recursos para poder realizar tareas completamente distintas. La diferencia con los sistemas *grid* radica en la manera de computar las tareas. La computación en *grid* divide tareas grandes en partes más pequeñas y las ejecuta en varias máquinas. Sin embargo, la computación en la nube ofrece al usuario servicios y recursos a más alto nivel.

2.3 Herramientas de monitorización de servicios cloud

Actualmente existe una gran cantidad de herramientas de monitorización que ofrecen servicios básicos como la extracción de datos del servicio monitorizado o herramientas más completas que aparte de ofrecer la extracción de datos, también permiten la generación de informes y visualización de los resultados obtenidos.

Como se ha comentado en la sección 2.1 de este capítulo, se pueden clasificar estas herramientas según los tipos de datos que se pueden obtener del servicio monitorizado. Hay una gran cantidad de herramientas de monitorización de servicios en la nube como Stackdriver Monitoring [15], AppDynamics [5], Amazon Cloud Watch [3] y vRealize Hyperic [17] son capaces de monitorizar tanto a nivel de la infraestructura como a nivel de aplicación. Por otra parte, hay herramientas que son capaces de monitorizar a nivel de infraestructura, como Dynatrace [9], CopperEgg [8] y otras a nivel de plataforma, como SysDig [16].

La creciente variedad de plataformas y tecnologías que ofrecen servicios en la nube, han generado la especialización de las herramientas en plataformas concretas, dificultando de esta manera encontrar una solución multiplataforma.

Por otra parte, existe una gran cantidad de herramientas de monitorización ligadas a la plataforma de servicio debido a que los proveedores están interesado en ofrecer un servicio de confianza y de calidad a los consumidores. En el caso de Stackdriver Monitoring [15], monitor de Google Cloud Platform, incluye diagnóstico, supervisión y registro de las aplicaciones; Azure Diagnostics [7] en caso de Microsoft Azure, ofrece instancias de servicios de Windows Azure, bases de datos SQL de Azure y almacenamiento de Azure.

2.3.1 Monitorización basada en agentes

La monitorización basada en agentes consiste en instalar un agente en el servidor con el propósito de recopilar información sobre el funcionamiento del servicio. Cada agente se encarga de recopilar datos del servicio en el que se ejecutan y procesarlos para construir métricas a más alto nivel. Cada agente se encarga de enviar los datos de cada servicio a un segundo componente software llamado servidor de monitorización cuya tarea principal es recuperar los datos recolectados por los agentes, tratarlos si fuese necesario y almacenarlos para su análisis posterior. Este servidor suele encargarse también del ciclo de vida de los agentes, detectando agentes descontados y regulando el comportamiento del sistema en función del estado de los agentes, y además de lanzar alertas si fuese necesario.

2.3.2 Monitorización del SLA

Uno de los objetivos principales de la monitorización de servicios en la nube consiste en vigilar el cumplimiento del SLA. Un SLA es un contrato entre en proveedor de servicios y el consumidor en el cual se fijan los términos referentes a la calidad de provisión del servicio. En él se establecen los términos de nivel de calidad de servicio (Quality of Service - QoS) de todos los requisitos no funcionales. Por ejemplo, que el tiempo de respuesta del servidor no supere 50ms, que el servicio esté disponible las 24 horas, que el servidor pueda soportar 1000 consultas simultaneas.

Para poder comprobar el cumplimiento del SLA es necesario disponer de herramientas capaces de medir los atributos correspondientes a los requisitos no funcionales expresados en el documento. Para ellos es necesario disponer de un monitor capaz de extraer los datos a nivel de aplicación ya que los términos del SLA habitualmente asumen que serán garantizados a este nivel. Sin embargo, es un trabajo complicado para los monitores convencionales, ya que los datos extraídos del nivel de plataforma y la de infraestructura no pueden asociarse de manera obvia con las métricas requeridas en la capa de aplicación para medición de los requisitos no funcionales expuestos en el SLA.

Debido a ellos las técnicas de monitorización del SLA se centran en aspectos a más alto nivel. Su objetivo es establecerlas relaciones entre las especificaciones del SLA y en tipo de datos a obtener del servicio en la nube que permitan medirlos requisitos no funcionales. De manera que el posterior análisis de los datos obtenidos permita determinar el cumplimiento o no de los requisitos establecidos en el SLA.

2.3.3 Monitorización como Servicio (Monitoring as Service, MaaS)

La monitorización como servicio (*Monitoring as a Service, MaaS*) está marcando una nueva tendencia a la hora de ofrecer servicios en la nube. MaaS es un *framework* que ofrece el despliegue de varias funcionalidades de monitorización de servicios y aplicaciones en la nube.

MaaS ofrece herramientas de monitorización de estado que permite la monitorización de un componente en función de la métrica o el estándar elegidos. En la monitorización del estado, cierto aspecto de un componente es constantemente evaluado, el resultado de la monitorización se puede observar en tiempo real o se puede generar un informe periódicamente. Por ejemplo, el tiempo de respuesta del servidor a una petición puede ser analizado para determinar si se cumple el valor considerado como aceptable. Más tarde, los administradores del sistema pueden intervenir para solucionar el posible fallo o incluso solucionar el problema en tiempo real.

A continuación, se van a comentar algunas ventajas que ofrece MaaS:

El proveedor se encarga de configurar la infraestructura del hardware, la herramienta de monitorización, la configuración y detección de alertas. El cliente obtiene acceso al monitor de datos al cual puede acceder desde cualquier navegador de internet.

La alta disponibilidad del servicio permite al usuario acceder a la herramienta de monitorización en cualquier momento debido a que no se contemplan tiempos de inactividad en la herramienta.

El servicio permite disminuir el coste de propiedad de este tipo de herramientas debido a que el usuario del servicio ya no tiene que hacerse cargo de los costes que generan este tipo de aplicaciones a la hora de monitorizar los servicios.

El servicio MaaS se beneficia del modelo de pago por uso, igual que el SaaS. Este modelo de facturación reduce los costes iniciales del uso de estas herramientas, ajustándose así a proyectos más pequeños con presupuestos reducidos que podrán beneficiarse de este tipo de herramientas.

2.3.4 Comparación entre las herramientas de monitorización

Para la comparación de las herramientas de monitorización se han tomado un conjunto de soluciones comerciales y académicas populares, incluyendo las propias de los grandes proveedores de servicios en la nube. Se

ha comparado la capacidad de ser utilizada en distintas plataformas, tipo de datos que son capaces de extraer de los servicios, el tipo de tecnología que utilizan, si son capaces de incorporar métricas personalizadas, y si ofrecen detección de violaciones del SLA. La Tabla 1 presenta un resumen de la comparativa de estas herramientas.

	Métodos de almacenaje de datos	Estadísticas	Datos de platadorma	Alertas	Agentes	Gráficas	Informes SLA	Licencia
Nimsoft	SQL	Si	Si	Si	Si	Si	Si	Comercial
Nagios	SQL	Si	Si	Si	Si	Si	Si	GPL
Zenoss	MySQL	Si	Si	Si	Si	Si	NO	GPL
Amazon Cloud Watch	MySQL, Oracle, Microsoft SQL Server	Si	Si	Si	Si	Si	NO	Comercial
Azure Diagnostics	SQL	Si	Si	Si	*	Si	NO	Comercial
Google App Engine	SQL, NoSQL	Si	Si	Si	Si	Si	NO	Comercial

Tabla 1. Comparación entre sistemas de monitorización

2.3.5 Conclusiones

La mayoría de las herramientas analizadas se centran en la extracción de datos de los niveles de plataforma e infraestructura de bajo nivel y dejan en manos de otras aplicaciones o del usuario el análisis de estos datos con el fin de establecer correspondencias entre los atributos de calidad y los datos procedentes del servicio. Hay una gran cantidad de herramientas que disponen de la capacidad de incorporar métricas personalizadas al proceso de extracción de datos, sin embargo, en muchas ocasiones tienen que ser implementadas por el usuario el en propio servicio.

Además, la monitorización de los SLAs no se soporta en la mayoría de herramientas de monitorización comerciales, por lo tanto, el usuario debe interpretar los datos monitorizados con el fin de comprobar el cumplimiento o no del acuerdo a nivel de servicio.

3. Cloud computing

En esta sección se profundizará más sobre los conceptos relacionados con la computación en la nube o *cloud computing.* En primer lugar, se definirá el concepto de computación en la nube. En segundo lugar, se comentará las ventajas y desventajas del uso de la computación en la nube. En tercer lugar, se clasificarán los servicios cloud ofrecidos en la actualidad. Y en el último lugar, se explicará el significado del documento de acuerdo a nivel de servicio (*Service Level Agreement* - SLA) y su relevancia en la monitorización de los servicios en la nube.

3.1 Definición de la computación en la nube

La computación en la nube es un modelo de negocio que ofrece un sistema informático como servicio al cual pueden acceder los usuarios desde cualquier lugar mediante el uso de internet. La utilidad de la computación en la nube es ofrecer un servicio rápido, seguro y fácil de utilizar por los usuarios sin experiencia previa en ese tipo servicios.

Es un modelo que permite el acceso mediante la red a distintos recursos informáticos configurables los cuales se componen de las siguientes características esenciales:

- **Accesibilidad:** Gracias a internet, las aplicaciones están disponibles en la red y los usuarios pueden acceder a ellos desde cualquier lugar mediante un ordenador o un teléfono móvil.
- **Elasticidad y escalabilidad:** Las aplicaciones cloud son totalmente elásticas en cuanto a su rapidez de implementación y adaptabilidad. Además, son totalmente escalables debido a que varían en función de la demanda de los usuarios.
- **Seguridad:** Los datos son alojados en *data centers*, empresas específicamente dedicadas a la custodia y salvaguarda de los datos. Son empresas que cuenten con todas las medidas de seguridad necesarias para evitar pérdida de datos.
- **Supervisión del servicio:** Los sistemas cloud controlan y optimizan el uso de los recursos de manera automática, por lo que el uso de estos puede seguirse, controlarse y notificarse, lo que aporta transparencia tanto para el proveedor como para el consumidor del servicio utilizado.

- **QoS predefinidos:** Los términos de calidad van expresados en las cláusulas de los acuerdos de nivel de servicio (SLA) los cuales expresan los niveles de calidad que ofrece el servicio.
- **Asignación de recursos en modo multiusuario:** A diferencia de las aplicaciones de software tradicionales, en cloud computing el proveedor tiene una única aplicación que abre a todos los usuarios que desean utilizarla, estableciendo unos recursos de acceso y prestaciones distintos para cada uno de los usuarios.

3.2 Ventajas y riesgos del cloud computing

En esta sección se comentarán las ventajas y desventajas de los sistemas de computación en la nube, las cuales entran en ámbitos económicos, seguridad, rendimiento y muchos más.

3.2.1 Ventajas del cloud computing

Desde el punto de vista económico podemos destacar que una aplicación desplegada en la nube elimina los costes adicionales como la compra de licencias, la inversión en la infraestructura, el mantenimiento de los equipos y sistemas o la adaptación de los mismos a las nuevas necesidades. Cabe destacar además que este servicio se beneficia del modelo de pago por uso, en el cual el usuario solo paga una cuantía fija por el paquete de servicios que utiliza.

Otra de las ventajas que ofrece la computación en la nube es que los servicios ya no se enfrentan a la elección entre el software obsoleto o a los altos costes de las actualizaciones. En una aplicación basada en la web, las actualizaciones desde el punto de vista del usuario se hacen de forma automática y estarán disponibles la próxima vez que el usuario inicie sesión. El usuario siempre dispondrá de la última versión de la aplicación sin necesidad de pagar o instalar actualizaciones.

Los servicios en la nube proporcionan aplicaciones fáciles de utilizar, evitando configuraciones complejas para que los usuarios sin experiencia previa en ese tipo de servicios no tengan dificultad a la hora de utilizarlos.

Una de las principales ventajas que ofrece este tipo de servicios es la escalabilidad y la flexibilidad. Normalmente los servicios funcionan con picos de demanda variables, por lo tanto, es esencial que el servicio se mantenga estable y pueda atender un número de usuario variable sin perder calidad ni rendimiento ofrecidos. El usuario siempre percibe la misma calidad del servicio

en cualquier instante de tiempo independientemente de la carga a la que están sometidos los servidores.

Debido a que el servicio es accesible mediante internet, el tiempo de puesta en marcha de nuevos servicios es muy reducido puesto que no es necesario comprar e instalar recursos hardware y software adicionales, hacer la planificación ni lidiar con los problemas de las instalaciones y actualizaciones, ya que el proveedor se encargará de ello.

3.2.2 Riesgos del cloud computing

La principal desventaja de la computación en la nube es la necesidad de una conexión permanente a internet. Dado que se utiliza internet para conectarse a las aplicaciones y por lo tanto a sus datos y documentos, si no se dispone de una conexión a internet no se podrá acceder ningún servicio ofrecido en la nube.

Mal funcionamiento con las conexiones de baja velocidad. Del mismo modo, una conexión a internet de baja velocidad hace que la computación en la nube en muchos casos sea imposible. Las aplicaciones basadas en web requieren una gran cantidad de ancho de banda para poder funcionar de forma correcta.

Existe la posibilidad de que los datos almacenados se pierdan. La mayoría de las empresas que ofrecen servicios de computación en la nube toman las precauciones suficientes para que esto no ocurra, por ejemplo, instalando sistemas de alimentación eléctrica ininterrumpidos, almacenamientos tolerantes a fallos, servicios de copias de seguridad automáticos, almacenamiento de las copias de seguridad en ambientes protegidos físicamente (contra incendios o robos), pero, no obstante, al ser medios físicos nunca proporcionarán una seguridad al 100%.

Otro problema con el cual nos podemos encontrar es la privacidad de los datos. Una vez desplegados los datos en el servidor, dejan de ser privados y pasan a estar al cargo del proveedor del servicio. Aunque se firme un acuerdo legal muy detallado de confidencialidad que obligue al proveedor a seguir ciertos procedimientos para garantizar la privacidad de los datos, siempre puede haber problemas técnicos, de seguridad o de mantenimiento incorrecto que pueden causar que los datos internos de la empresa o de los clientes sean filtrados en internet y sean conocidos por todo el mundo.

3.3 Clasificación de servicios cloud

En esta sección se expondrán los distintos métodos de clasificación de los servicios en la nube, primero atendiendo al criterio de la forma de despliegue y en segundo lugar al modelo de servicio ofrecidos:

3.3.1 Modelos de despliegue

El despliegue de los servicios cloud incluye procesos mediante los cuales se procede a la puesta en marcha de las infraestructuras y del software necesario para que el servicio esté listo para su utilización. Según las características de la infraestructura de despliegue se puede clasificar a un servicio como:

- ***Public Cloud* (Nube Pública)*:*** La infraestructura de la nube está disponible al público en general. El usuario puede acceder al uso del software o hardware de forma libre o mediante el pago de una suma de dinero por el uso de los mismos.
- ***Private Cloud* (Nube Privada):** La infraestructura de la nube pertenece a una única organización, quien la ofrece como servicio a sus propios departamentos. Puede ser gestionada por la organización o por una tercera parte, y puede estar en los locales de la organización, o fuera de ellos.
- ***Community Cloud* (Nube comunitaria):** La infraestructura de la nube es compartida por varias organizaciones y da soporte a una comunidad específica que tenga asuntos compartidos.
- ***Hybryd Cloud* (Nube Híbrida):** La infraestructura de la nube está compuesta por dos o más tipos de nubes (privada, pública o comunitaria) que mantienen su propia identidad pero que son unidas por una tecnología propietaria o estándar para permitir la portabilidad de datos y aplicaciones.

3.3.2 Modelos de servicio

Teniendo en cuenta qué tipos de servicio ofrece el proveedor podemos encontrar:

- **Software como servicio (*Software as a Service,* SaaS):** SaaS representa el conjunto de aplicaciones accesibles 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 se suscriben

a un determinado servicio que se distribuye a través de la red, en lugar de desarrollarla o comprarla e instalar en un sistema físico.

- **Plataforma como servicio (*Platfom as a Service,* PaaS):** PaaS se destina 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.
- **Infraestructura como servicio (*Infrastructure as a Service,* PaaS):** PaaS ofrece a los usuarios procesamiento, almacenamiento, conexiones e interfaces de red, y otros recursos de cómputo mediante los cuales el cliente es capaz de desplegar y ejecutar software. Este software puede incluir tanto sistemas operativos como aplicaciones.

A continuación, la Figura 1 muestra la relación entre las tres capas y los posibles servicios y aplicaciones que pertenecen a cada una de ellas.

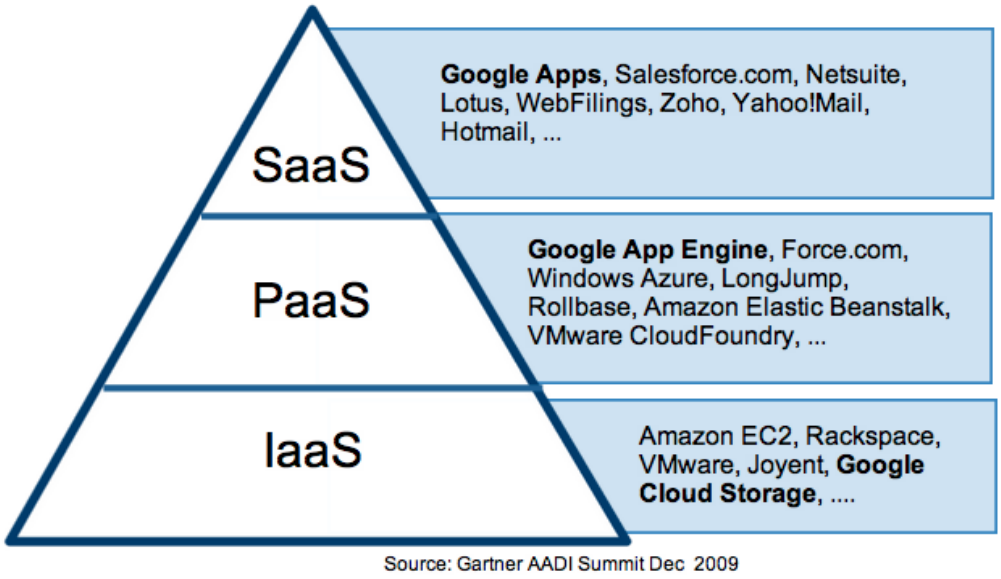

Figura 1. Pirámide de capas de abstracción en la nube

3.4 El acuerdo de nivel de servicio

El mayor problema actual de la computación en la nube es fa falta de confianza entre el proveedor del servicio y el consumidor debido a los riesgos que tiene y a su novedad. En esta sección se describirá el rol que juega el documento de acuerdo de nivel de servicio para regular entre el cliente y el proveedor para que cada una de las partes conozca sus obligaciones y derechos.

Los términos de un servicio en la nube contratado vienen definidos por dos partes que constituyen el mismo documento. El acuerdo de servicio y el Acuerdo de Nivel de Servicio (*Service Level Agreement,* SLA).

El acuerdo de servicio establece la relación legal entre el cliente y el proveedor del servicio cloud. En él se establecen las reglas legales de la relación entre consumidor y proveedor. Este documento es acompañado por el SLA, un documento que establece el acuerdo a cumplir para los clientes. Se establecen compromisos que son niveles de servicio acordados entre el proveedor de servicios y el cliente, así como el procedimiento a seguir en caso del incumplimiento de los acuerdos establecidos, especificando la recompensa que obtendrá el cliente en este caso.

El SLA define los niveles de calidad y debe contener una descripción clara del servicio a prestar, e irá acompañada de la cuantificación de algunos parámetros de carácter técnico como el tipo de servicio, soporte a clientes y asistencia, provisiones para seguridad y datos, disponibilidad del sistema, conectividad y multas por caída del sistema. En el contrato es imprescindible especificar las necesidades con claridad y precisión, así como fijar unos objetivos para que no haya futuros problemas entre el proveedor y el cliente. La elaboración y negociación del SLA lo suelen llevar perfiles técnicos/informáticos del cliente y del proveedor del servicio más que sus respectivos abogados, pero estos deben estar en un marco conceptual adecuado sino no se tiene eficacia por sí solo.

Cuando se trata de contratos largos, es conveniente que el contrato sea revisado periódicamente de forma que se puedan ir mejorando los acuerdos establecidos inicialmente. Una manera para poder cualificar cuándo no se está cumpliendo la calidad acordada es mediante un informe periódico, en el que se muestra qué indicadores se están cumpliendo y cuáles de los acordados no. Si se produce una situación donde los clientes no están satisfechos, hay que plantearse redefinir objetivos e implementar medidas correctivas.

En caso de incumplimiento del servicio, también existen una serie de penalizaciones que tienen transcendencia jurídica. La forma de solventarlas es a través de indemnizaciones al cliente por parte del proveedor. Algunas de ellas pueden ser una reducción del importe en la siguiente factura, algún tipo de compensación económica o bonus y concesión de créditos para consumir nuevos servicios por menos precio. También existe el derecho por parte del cliente de terminar el contrato por incumplimiento del SLA hasta un grado que debe fijarse específicamente. El SLA se considera un elemento complementario al contrato, ya se éste solo regula el servicio, pero, por el contrario, el contrato regula todo el marco legal de la relación.

Por último, los inconvenientes del SLA describen las excepciones a la hora de tratar las promesas del contrato. En estas limitaciones se pueden incluir las paradas de mantenimiento programadas, de las cuales se tiene que avisar con antelación; desastres naturales, brechas de seguridad, cambios en la facturación y cambios en la seguridad.

En la mayoría de los casos el cliente es el responsable de comprobar si se cumple el SLA. Para ello necesita disponer de herramientas capaces monitorizar los servicios utilizados y generar un informe cuando se produzca el incumplimiento del SLA para poder presentarlo ante el proveedor para reclamar la correspondiente indemnización.

3.4.1 Calidad del servicio (Quality of Service, QoS)

En la computación en la nube, QoS [14] 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 dependiente de la aplicación o servicio desplegado en la nube. Por, ejemplo, si un 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 de negocio, la época del año, o incluso la hora de día.

Un control de la QoS [14] 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 o bien a los asociados a las condiciones del proveedor. Dentro de este último caso, se encuentran aquellas situaciones asociadas a los cambios en las condiciones del servicio por parte del proveedor, ya se directamente QoS [14], tarificación, aparición de nuevos competidores o políticas de seguridad.

3.5 Plataformas Cloud

En esta sección se hace una breve introducción a las principales plataformas cloud existentes (Microsoft Azure, Google App Engine y Amazon Web Service) y a las herramientas de monitorización ofrecidas por estas plataformas.

3.5.1 Microsoft Azure

Microsoft Azure[3] es un servicio en la nube creado por Microsoft para desarrollar, testear, desplegar y administrar aplicaciones y servicios alojados en los *data centers* de Microsoft. La plataforma Microsoft Azure fue anunciada en 2008 ya lanzada en 2010 como Windows Azure y posteriormente en 2014 fue renombrada a Microsoft Azure.

La plataforma ofrece servicios SaaS, IaaS y PaaS y soporta distintos lenguajes de programación, herramientas y frameworks, incluidos los pertenecientes a terceros y los propios incluidos en Microsoft.

3.5.1.1 Microsoft Azure Diagnostics

Azure Diagnostics [10] es un software que se encarga de recolectar la información de las aplicaciones desplegadas en Microsoft Azure. La herramienta recopila datos de bajo nivel procedentes de los servicios y los almacena en una cuenta de almacenamiento elegida por el usuario.

Con este servicio, podemos recopilar información del rendimiento, monitorizar, ver los logs y los registros que deja una aplicación, incluso también, la información de seguimiento del servicio alojado en Azure.

Es posible también agregar diferentes contadores de rendimiento, pudiendo monitorizar el rendimiento de cada zona de una aplicación. Además, podríamos ver los archivos de registro, personalizarlos y realizar un seguimiento de cada uno.

[3] https://azure.microsoft.com

3.5.2 Google App Engine

Google App Engine[4] es una plataforma utilizada para crear aplicaciones web y *backends* móviles escalables. Cuenta con APIs y servicios integrados utilizados en la mayoría de aplicaciones, como bases de datos NoSQL, Memcache y una API de autenticación de usuarios.

La plataforma ofrece cuentas gratuitas de 500 megabytes de almacenamiento permanente y ancho de banda y capacidad de cómputo suficiente para cinco millones de visitas mensuales. Si se superan las cuotas, se pueden adquirir servicios adicionales abonando una determinada cuota.

3.5.2.1 Google Stackdriver

Google Stackdirver [3] es una plataforma de monitorización para los servicios en la nube de Google App Engine y Amazon We Services (AWS) [4] Esta herramienta ofrece a los usuarios funciones de control de tiempo de actividad, alertas, análisis del estado, registros de actividad, informes de errores y sistemas de depuración.

3.5.3 Amazon Web Services (AWS)

Amazon Web Services[5] es una plataforma cloud lanzada por Amazon. Los primeros servicios se lanzaron en 2006 y proporcionaba servicios online para páginas web y aplicaciones cliente.

Para minimizar el impacto de las interrupciones y asegurar la robustez del sistema, AWS está geográficamente clasificado en regiones. Cada región se compone de múltiples áreas más pequeñas denominadas áreas de disponibilidad.

Algunos de los servicios ofrecidos por AWS son:

- **CloudDrive:** permite a los usuarios subir y tener accesos a la música, videos, documentos y fotos desde dispositivos conectados a la red.
- **CloudSearch:** servicios de búsqueda escalable típicamente utilizado para integrar capacidades de búsqueda personalizadas en otras aplicaciones.
- **Dynamo Database:** servicios de bases de datos NoSQL conocido por su baja latencia y escalabilidad.

[4] https://cloud.google.com
[5] https://aws.amazon.com

- **Simple Storage Service:** servicio escalable, de alta velocidad y de bajo precio diseñado para realizar copias de seguridad y almacenar datos y aplicaciones.

3.5.3.1 Amazon CloudWatch

Amazon CloudWatch es un servicio de monitorización de los recursos de la nube de AWS. Puede recopilar y realizar seguimiento de métricas, logs, establecer alarmas y reaccionar automáticamente a los cambios en sus recursos AWS. Amazon CloudWatch puede monitorizar instancias de EC2, tablas de Amazon DynamoDB en instancias de Amazon RDS, así como métricas personalizadas generadas por las aplicaciones y servicios, y los logs generados por las aplicaciones. También se ofrece la posibilidad visualizar la utilización de recursos, el desempeño de las aplicaciones y el estado del funcionamiento.

4. Infraestructura de Monitorización

En este capítulo se expone los antecedentes de este trabajo: el proceso de monitorización de servicios cloud basado en modelos en tiempo de ejecución propuesto por Cedillo et al. [1] y las herramientas de monitorización (Configurador y Monitor) que soportan dicho proceso.

4.1 Proceso de Monitorización

El proceso de monitorización propuesto por Cedillo et al. [1] consiste en 3 tareas, la cual se subdividen en una serie de actividades. Este proceso se basa en la técnica de control de bucle autonómico (*autonomic control loop*). La idea de esta técnica es la de medir los parámetros de sistema, analizarlos, diseñar medidas correctivas en el caso de que sea necesario y ejecutar estas acciones en orden de mejorar el sistema.

El proceso de monitorización está compuesto por las siguientes actividades: (1) la configuración de la monitorización, (2) monitorización y (3) el análisis de los resultados. Como podemos ver en la Figura 2, el proceso de monitorización comienza con la configuración de la monitorización cuyo resultado es el modelo en tiempo de ejecución que va a ser utilizado por el proceso de monitorización y posteriormente para el análisis del resultado.

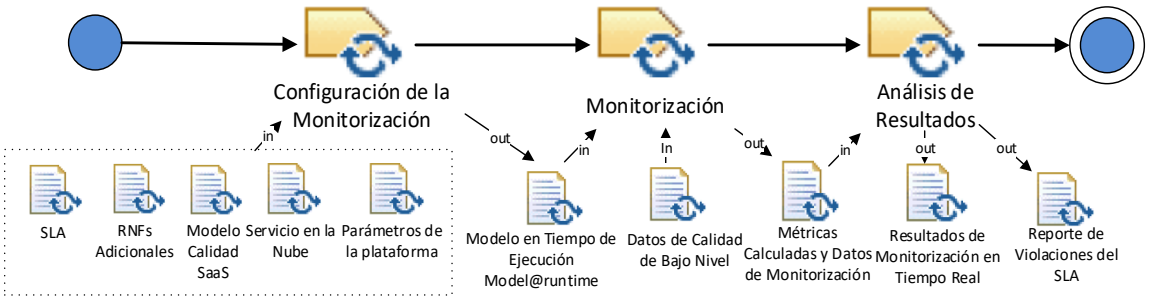

Figura 2. Proceso de monitorización

El objetivo de la actividad de Monitorización es capturar la información de bajo nivel de los servicios en tiempo de ejecución utilizando técnicas que transformarán los datos en información de más alto nivel los cuales permitirán a la tarea de Análisis de Resultados obtener un informe sobre la calidad del servicio.

La actividad de Análisis de Resultados utiliza los datos generados en el proceso de monitorización y los comparara con los requisitos no funcionales especificados en el SLA para generar un informe de violaciones del SLA.

En el punto siguiente explicaremos las características de cada una de estas actividades, así como sus entradas y sus resultados.

4.1.1. Configuración de la monitorización

La configuración de la monitorización es la actividad responsable de la preparación del modelo en tiempo de ejecución, que es el que contiene las directivas de monitorización. Ésta es una actividad en la que intervienen los usuarios directamente y en la que se pueden identificar dos actores principales: (1) el *Planificador de la Monitorización*, que se encargará de establecer los requisitos de monitorización y definir los requisitos no funcionales (RNF) adicionales, en el Modelo de Requisitos de Monitorización; (2) *Configurador de la Monitorización*, quien se encargará de realizar la configuración de la monitorización, hasta la obtención del Modelo de Calidad en Tiempo de Ejecución.

Las tareas de esta actividad son descritas en detalle a continuación, y están ilustradas en la Figura 3.

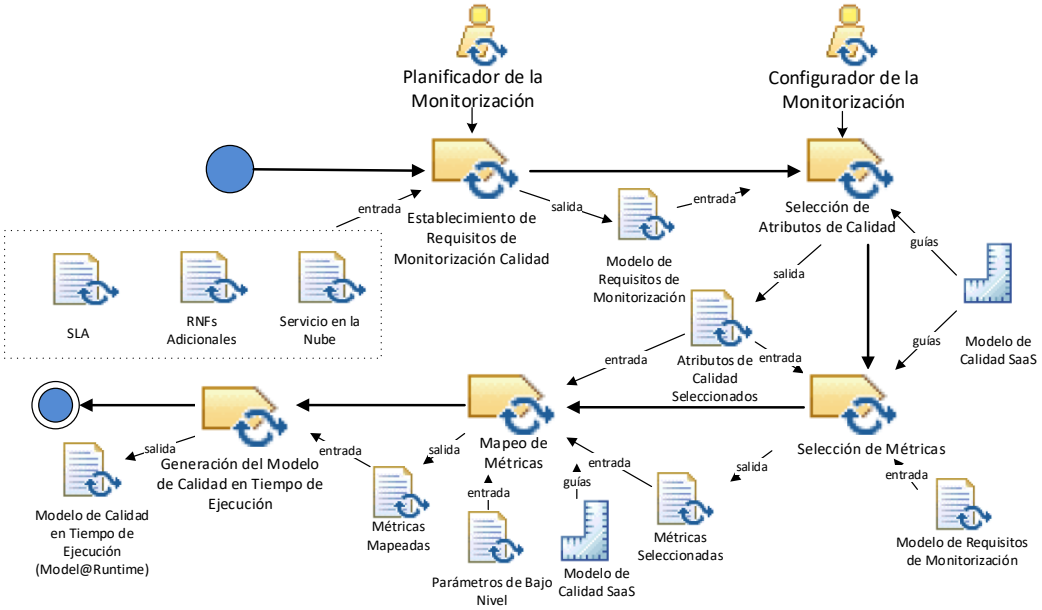

Figura 3. Diagrama SPEM de la Actividad de Configuración de la Monitorización

A continuación, se van a resumir los pasos que se realizan a la hora de configurar el monitor.

1. **Establecimiento de los requisitos de monitorización**

   Esta tarea es realizada por el planificador de la monitorización. Esta tarea recibe como entradas el Acuerdo de Nivel de Servicio, un documento con otros Requisitos Adicionales de Monitorización y los datos del Servicios en la Nube que será analizado en el proceso de monitorización. Toda esta información será recogida de manera estructurada en el documento de Especificación de Requisitos de Monitorización.

2. **Selección de los atributos de calidad**

   Se toma el documento de Especificación de Requisitos de Monitorización y utilizando como guía un Modelo de Calidad SaaS el Planificador de la Monitorización se encarga de relacionar cada Requisito de Monitorización con el atributo del modelo de calidad SaaS apropiado. Dando como lugar al artefacto de Atributos de Calidad Seleccionados.

3. **Selección de métricas de calidad**

   A los Atributos de Calidad Seleccionados en la tarea de Seleccionar Medidas, se los relacionará uno por uno con una métrica elegida, según los criterios del Configurador de la Monitorización, tomando en cuenta las indicaciones del Modelo de Calidad SaaS. La salida de esta tarea serán el conjunto de medidas asociadas a los atributos de calidad seleccionados identificadas como el artefacto Métricas Seleccionadas

4. **Mapeo de las métricas de calidad**

   Las Métricas Seleccionadas todavía se encuentran expresadas en términos independientes de la plataforma no válidos para su puesta en marcha dentro de la tarea de medición del Middleware, de manera que estas deberán ser asociadas con otras Métricas ahora sí expresadas en términos dependientes de la plataforma. Estas métricas expresan a nivel de datos crudos obtenibles por la plataforma de monitorización fórmulas equivalentes a la métrica seleccionada procedentes del modelo de calidad SaaS.

5. **Generación del Modelo de Calidad en Tiempo de Ejecución**

   Las métricas ya escritas de manera dependiente de la plataforma situadas en el artefacto Métricas Asociadas pasarán a transformarse en un modelo en tiempo de ejecución siendo este el producto final de la

primera tarea del proceso y que pasará al proceso del Middleware dónde será empleado para realizar la monitorización del servicio.

4.1.2. Monitorización

La actividad de *Monitorización* recibe como artefacto de entrada el *Modelo en Tiempo de Ejecución*, que contiene todos los parámetros necesarios para poder realizar la monitorización de la calidad de los servicios.

El modelo en tiempo de ejecución contiene una descripción de los requisitos no funcionales a monitorizar, de las características, sub-características, atributos y métricas a ser utilizadas en la fase de monitorización y los datos que se van a extraer del servicio en tiempo de ejecución para el cálculo de las métricas. Por lo tanto, este modelo está conformado por los requisitos de monitorización, atributos de calidad a ser monitorizados, métricas independientes e dependientes de la plataforma cloud, así como los parámetros necesarios de acceso y configuración de los servicios a ser monitorizados.

La monitorización utiliza métricas para la medición de las características de calidad de los servicios a ser monitorizados. Los datos de bajo nivel de los servicios son recogidos a través de varios mecanismos de recolección de datos (contadores de la plataforma, métricas directas, wrappers, etc.) y son transformados en datos más significativos mediante la aplicación de métricas indirectas.

4.1.2. Análisis de Resultados

En esta actividad se compara los valores obtenidos por la monitorización con los umbrales establecidos en el SLA y contenidos en el modelo de calidad en tiempo de ejecución. Esta comparación permite el análisis de los resultados y la detección de incumplimientos del SLA.

4.2 Arquitectura de la Infraestructura de Medición

La Figura 4 presenta la arquitectura general de la infraestructura de monitorización Cedillo et al. [1] con sus componentes principales.

La infraestructura de monitorización se divide en dos componentes principales, los cuales permiten:

1. **Configurador:** permite la especificación y configuración de RNFs a ser monitorizados.
2. **Monitor:** extrae los datos de los servicios cloud en tiempo de ejecución, calcula las métricas para evaluar la calidad de los servicios y genera informes del incumplimiento del SLA.

Esta infraestructura permite lograr estos objetivos mediante un conjunto de componentes y artefactos que conforman la infraestructura de monitorización y utilizan modelos en tiempo de ejecución.

El Configurador de la monitorización usa el *Modelo de Requisitos de Monitorización* y el *Modelo de Calidad SaaS* para configurar la monitorización de los servicios y obtener el *Modelo en Tiempo de Ejecución*.

El Monitor usa el modelo en tiempo de ejecución y contiene dos motores: el *Motor de Medición*, el cual permite la monitorización de servicios cloud a través del uso de datos de calidad tomados directamente desde los servicios y la respectiva realización de mediciones, y el *Motor de Análisis*, el cual compara los valores esperados con los valores monitorizados y permiten generar el informe de violaciones del SLA.

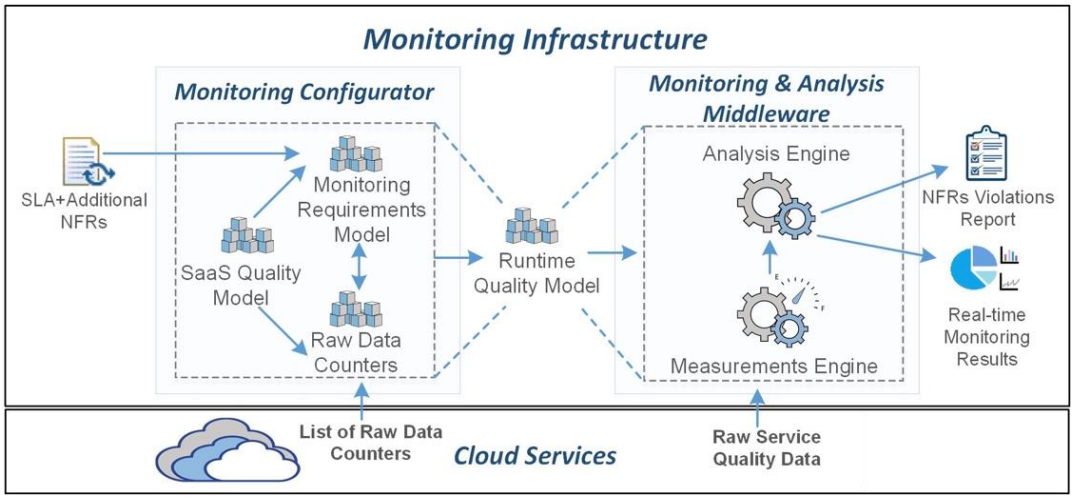

Figura 4. Infraestructura de Monitorización

4.3 Arquitectura del Monitor de Servicios Independiente de la Plataforma

La Figura 5 muestra la arquitectura del middleware de monitorización independiente de la plataforma propuesto por Jiménez [20]

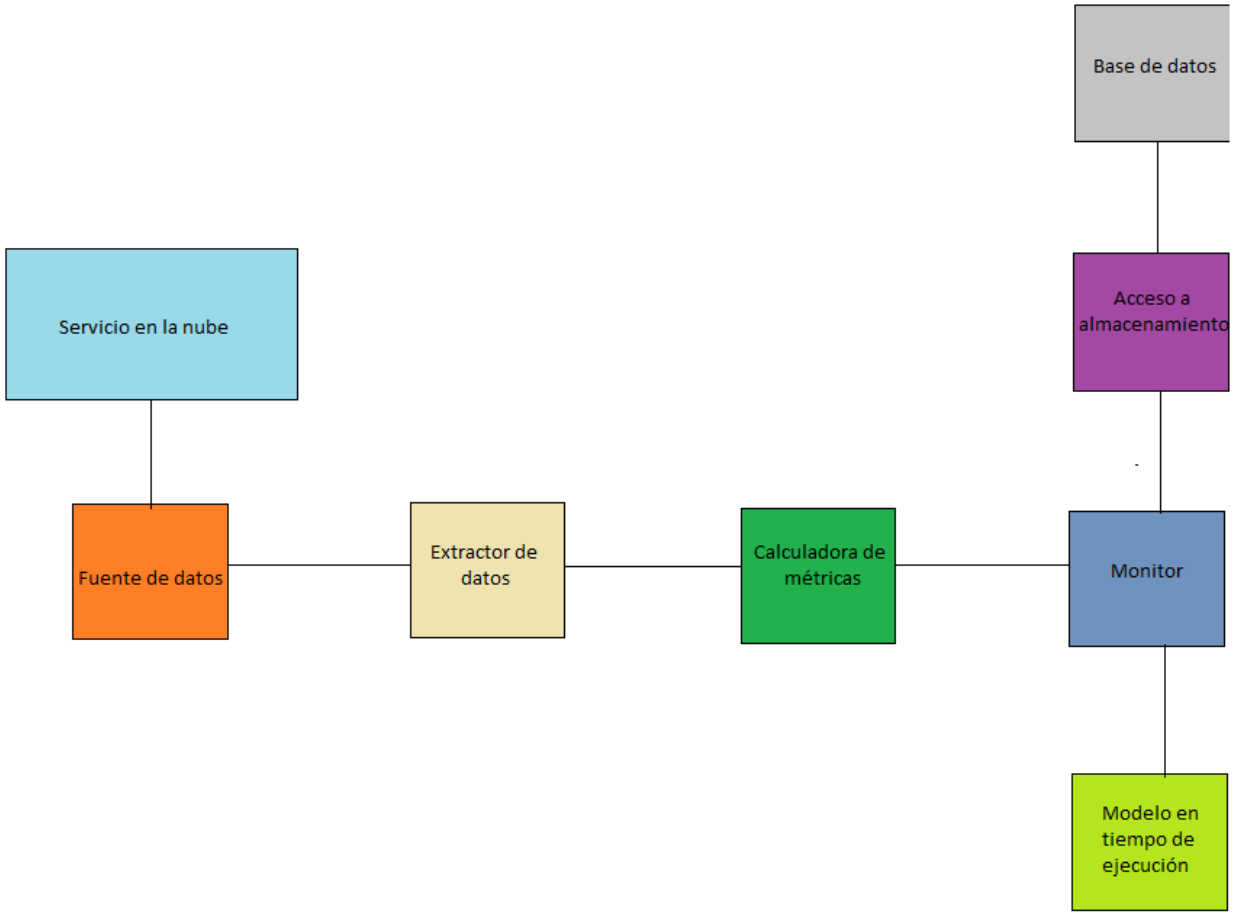

Figura 5. Arquitectura del middleware de monitorización de servicios

El middleware de monitorización de servicios cloud está compuesto de ocho componentes, los cuales se conectan entre sí para realizare a cabo las tareas de extracción, procesamiento y análisis de las métricas. A continuación, se explicarán los distintos elementos:

- **Modelo en tiempo de ejecución:** Es el elemento de entrada del middleware y es el resultado del configurador de la herramienta de monitorización
- **Extractor de datos:** Elemento encargado de extraer los datos del servicio cloud. Crea la conexión con el servicio en la nube y ejecuta las llamadas necesarias para extraer los datos.

- **Servicio en la nube:** Servicio a monitorizar desplegado en la plataforma.
- **Base de datos:** Base de datos en la cual se guardan los resultados del proceso de extracción de datos del servicio, datos del cálculo de las métricas y los datos del modelo en tiempo de ejecución.
- **Monitor:** Se encarga de gestionar los procesos de la aplicación. Recibe el modelo en tiempo de ejecución, especifica las métricas a extraer, gestiona la extracción de los datos del servicio para realizar el cálculo de las métricas del servicio y almacena los datos necesarios.
- **Calculadora de métricas:** Conjunto de procesos que definen las fórmulas necesarias para calcular el valor de las métricas a monitorizar.
- **Fuente de datos:** servicio que proporciona a las aplicaciones externas la posibilidad de hacer peticiones para extraer datos del servicio.
- **Acceso a almacenamiento:** Elemento encargado de gestionar la base de datos. Extrae los datos de la base de datos, realiza el mapeo de las tablas de las bases de datos con las clases de la aplicación, establece la conexión con la base de datos y ejecuta las tareas de creación, actualización y eliminación de los elementos de la base de datos.

El middleware independiente de plataforma propuesto por Jiménez ha sido implementado en la plataforma Microsoft Azure. Los detalles de la implementación pueden ser encontrados en [20].

4.4 Conclusiones

La herramienta de monitorización de servicios en la nube presentada por Jiménez [20] es una herramienta sólida y robusta que cumple a la perfección la finalidad para la cual se ha diseñado. Presenta una estructura algo compleja, pero está muy bien documentada para facilitar su comprensión y aplicación en la monitorización de servicios en la plataforma Azure.

Por otra parte, si se orienta de cara a un usuario inexperto en la materia, es probable que se le presenten dificultades a la hora de interpretar los datos monitorizados por la herramienta. Por lo tanto, añadirle a la estructura de la

herramienta de Jiménez [20] una parte adicional, la de visualización de los datos serviría de gran ayuda al usuario.

De esta forma, se pretende facilitar al usuario la interpretación de los datos monitorizados. Si se dispone de una herramienta capaz de visualizar los datos monitorizados y compararlos con el umbral de cada atributo de calidad especificado en el SLA, se conseguiría que los usuarios que no están familiarizados con los servicios en la nube tengan más facilidad a la hora de interpretar los datos recogidos por el monitor. Este será el objetivo principal de este trabajo.

5. Herramienta de Visualización de Datos de Monitorización

La herramienta de visualización a ser desarrollada en este trabajo pretende facilitar el seguimiento por parte del usuario de aquellos atributos de calidad que desea monitorizar (disponibilidad, rendimiento, seguridad, etc.), mostrando la información mediante distintas gráficas interactivas dentro de un *dashboard*. Además, también permitirá la generación de informes donde se reflejen con detalle las violaciones del acuerdo de nivel de servicio (SLA) entre cliente y proveedor.

A continuación, la Figura 6 muestra un diagrama UML de casos de uso que describe los requisitos de la herramienta (lo que puede realizar el usuario de la aplicación) y posteriormente se comentará cada caso de uso a desarrollar.

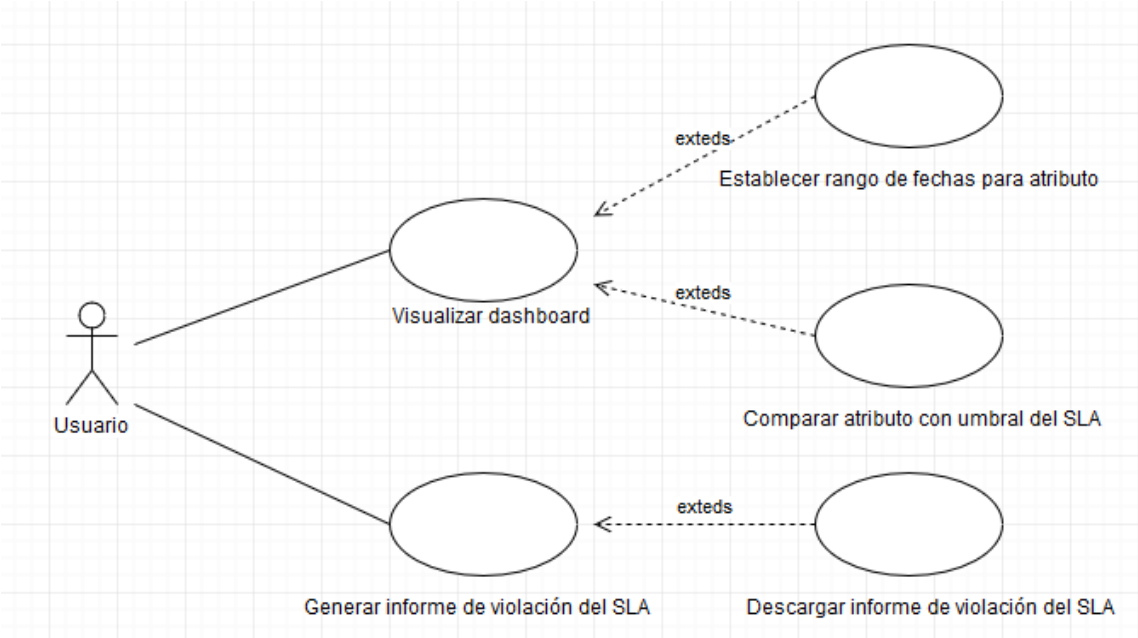

Figura 6. Diagrama UML de casos de uso

Dentro del cloud computing existen distintos tipos de usuario, en este proyecto hablaremos de los 3 más reconocidos. El consumidor, persona que contrata los servicios de un proveedor y utiliza sus servicios. El proveedor, entidad que proporciona los servicios cloud computing. Y, por último, el *bróker* o intermediario, entidad que media entre el consumidor y el proveedor, velando

por el cumplimiento del contrato entre ambos. La herramienta a desarrollar deberá ser de utilidad para cualquiera de estos usuarios.

En particular, el usuario de la aplicación y actor principal podrá ejecutar los siguientes casos de uso:

- **Visualizar dashboard**

Una vez seleccionados correctamente en el configurador los distintos atributos de calidad que se deseen monitorizar se pueden acceder a un *dashboard* donde se visualiza mediante gráficas interactivas el estado actual y la evolución de cada uno de ellos.

- **Comparar atributo con umbral del SLA**

Para cada atributo de manera independiente, puede elegirse compararlo con el umbral para cada atributo de calidad acordado en el SLA, visualizando el resultado en el *dashboard*. Esta funcionalidad mostrará al usuario la información de aquellos atributos de calidad que se encuentran fuera de los niveles de calidad esperado (violación del SLA).

- **Establecer rango de fechas para atributo**

Para cada atributo de manera independiente, puede establecerse un rango de fechas con el objetivo de ver en el *dashboard* la evolución dentro de dicho rango (en lugar del que existe por defecto). Esto permitirá realizar una configuración y seguimiento más preciso de los datos a visualizar.

- **Generar informe de violación del SLA**

El usuario podrá generar un informe con los datos de medición de los atributos de calidad visualizados en el dashboard.

- **Descargar informe de violación del SLA**

Se podrá descargar el informe de violación del SLA generado en un archivo en formato PDF.

5.1 Arquitectura de la aplicación

A continuación, se va a presentar el esquema de la arquitectura utilizada para el desarrollo de la herramienta de visualización. Posteriormente se explicará el funcionamiento de cada una de las partes.

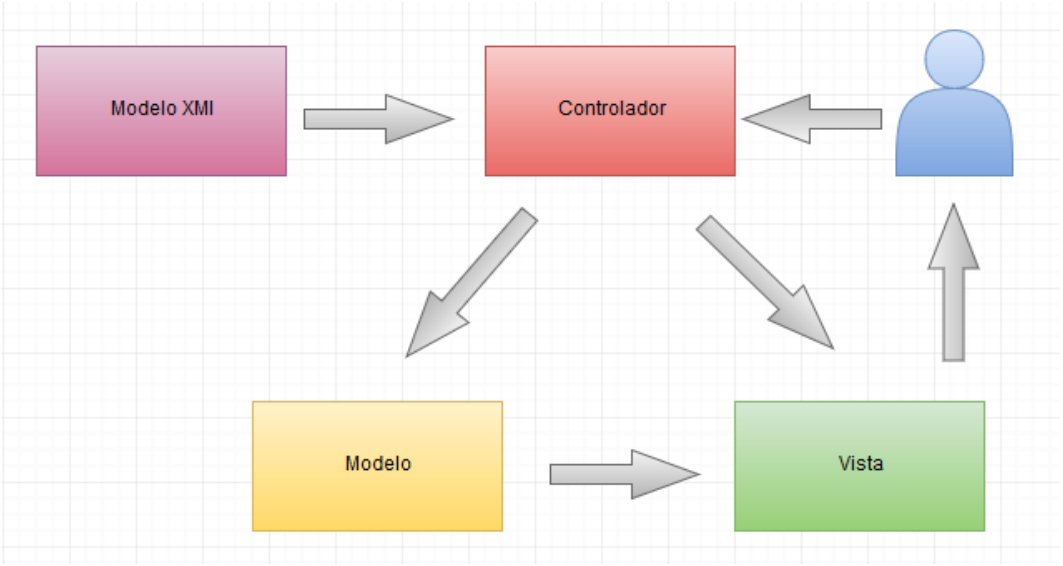

Figura 7. Arquitectura de la aplicación

Como punto de partida se ha utilizado el *Modelo en tiempo de ejecución* (archivo xmi) generado por la herramienta de monitorización de Jiménez [20]. Este archivo se es la salida de la herramienta Configurador y la entrada para la herramienta de visualización a desarrollar.

El archivo xmi contiene los nombres de los atributos de calidad que deben ser monitorizados, sus correspondientes umbrales descritos en el SLA, las métricas usadas para calcular la calidad del servicio y los contadores de la plataforma que serán usados para extraer la información del servicio en tiempo de ejecución. La herramienta Monitor de Jimenez utiliza el modelo en tiempo de ejecución (archivo xmi) para calcular las métricas y almacenar los valores de medición en una base de datos.

De esta manera, el archivo proporciona a la herramienta de visualización la información necesaria para acceder a la base de datos del monitor para extraer los valores de la monitorización y compararlos con sus respectivos umbrales especificados en el SLA. De esta forma la herramienta de visualización funciona como una extensión de las herramientas de monitorización Configurador y Monitor con las cuales establece la conexión mediante la interpretación de los datos proporcionados por el archivo xmi.

Para el desarrollo de la aplicación se ha utilizado la arquitectura Modelo-Vista-Controlador (MVC). Este modelo consiste en separar los datos y la lógica de negocio de una aplicación de la interfaz del usuario. A continuación, se explicará el funcionamiento de cada una de las capas que la componen:

- **Modelo:** Es la capa donde se trabaja con los datos, por lo tanto, dispone de mecanismos de acceso a la información y también para la actualización de su estado. Habitualmente la información se guarda en la base de datos, por lo cual los modelos disponen de funciones necesarias para acceder a las tablas.
- **Vista:** Esta capa contiene el código necesario para producir la visualización de las interfaces del usuario. En la vista utilizan los códigos HTML y ASP para poder mostrar la salida.
- **Controlador:** Esta capa contiene el código necesario para responder a las acciones que se solicitan en la aplicación, como la visualización de un elemento, solicitud de búsqueda, etc.

El uso de esta arquitectura permitirá varios beneficios. En primer lugar, la implementación se realiza de forma modular. Las vistas muestran siempre información actualizada. El programador no debe preocuparse de solicitar que las vistas se actualicen, ya que este proceso es realizado automáticamente por el modelo de la aplicación. Cualquier modificación como aumentar métodos o datos, implica una modificación sólo en el modelo y las interfaces del mismo con las vistas, no todo el mecanismo de comunicación y de actualización entre modelos.

Además, las modificaciones a las vistas no afectan al modelo de dominio, simplemente se modifica la representación de la información, no su tratamiento. En definitiva, la arquitectura MVC darán soporte a la extensibilidad y mantenibilidad de la herramienta, facilitando la incorporación de nuevas funcionalidades.

Como desventaja, para desarrollar la herramienta bajo el patrón de diseño MVC será necesario una mayor dedicación en los tiempos iniciales del desarrollo. Normalmente el patrón exige desarrollar un mayor número de clases que, en otros entornos de desarrollo, no son necesarias. Sin embargo, esta desventaja es muy relativa ya que posteriormente, en la etapa de mantenimiento la herramienta será mucho más mantenible, extensible y modificable.

5.2 Diseño de la herramienta de visualización

Esta sección describe el diseño del *dashboard* de la herramienta de visualización. En particular, se describe el tipo de gráficas soportadas por la herramienta, el tipo de datos recogidos por la herramienta de monitorización y la representación de los datos.

5.2.1 Diseño del Dashboard

Un dashboard es un medio de comunicación visual, donde se coloca la información más importante, requerida para llevar a cabo un o más objetivos; consolidada y organizada en una sola pantalla, de tal forma que la información, pueda ser monitoreada en una sola mirada. A continuación, se van a mostrar los mockups de los dashboards utilizados en la herramienta de monitorización:

En la Figura 8 se muestra el mockup del configurador de la herramienta. Es la primera página que ve el usuario al iniciar la aplicación. Para el diseño de esta parte se han seguido los siguientes principios de diseño de interfaces:

- **Contraste:** Consiste en evitar que todos los elementos de la página se vean iguales, si los elementos no corresponden a la misma clase, entonces hacerlos muy diferentes. En la Figura 8 el botón que se utiliza para subir el archivo es más grande que los demás para destacar entre los demás elementos debido a que es una parte esencial de la aplicación. Por otra parte, los botones y las listas desplegables tienen distinto tamaño para que se perciba que son elementos distintos.
- **Repetición:** Consiste en repetir elementos que pertenecen a la misma clase. Se pueden repetir colores, texturas, formas, tamaños, etc. En la Figura 8 se repite el mismo tamaño, color y textura para los elementos del mismo tipo. Los botones son un poco más grandes para que destaquen más, por otra parte, las listas desplegables son más pequeñas y están más próximas una de la otra.
- **Alineación:** Consiste en ubicar los elementos de forma clara y precisa. No puede haber elementos ubicados de forma arbitraria. Todos los elementos deben de tener una conexión visual con los elementos de la página. En la Figura 8 se sigue una estructura lineal que centra los elementos en una determinada parte de la página. De esta forma se aprecia que estos elementos tienen relación entre sí y forman parte del mismo proceso de configuración de la herramienta. Además, se ha añadido una serie de pasos numerados para que el usuario no se sienta desubicado al visualizar la página por primera vez.

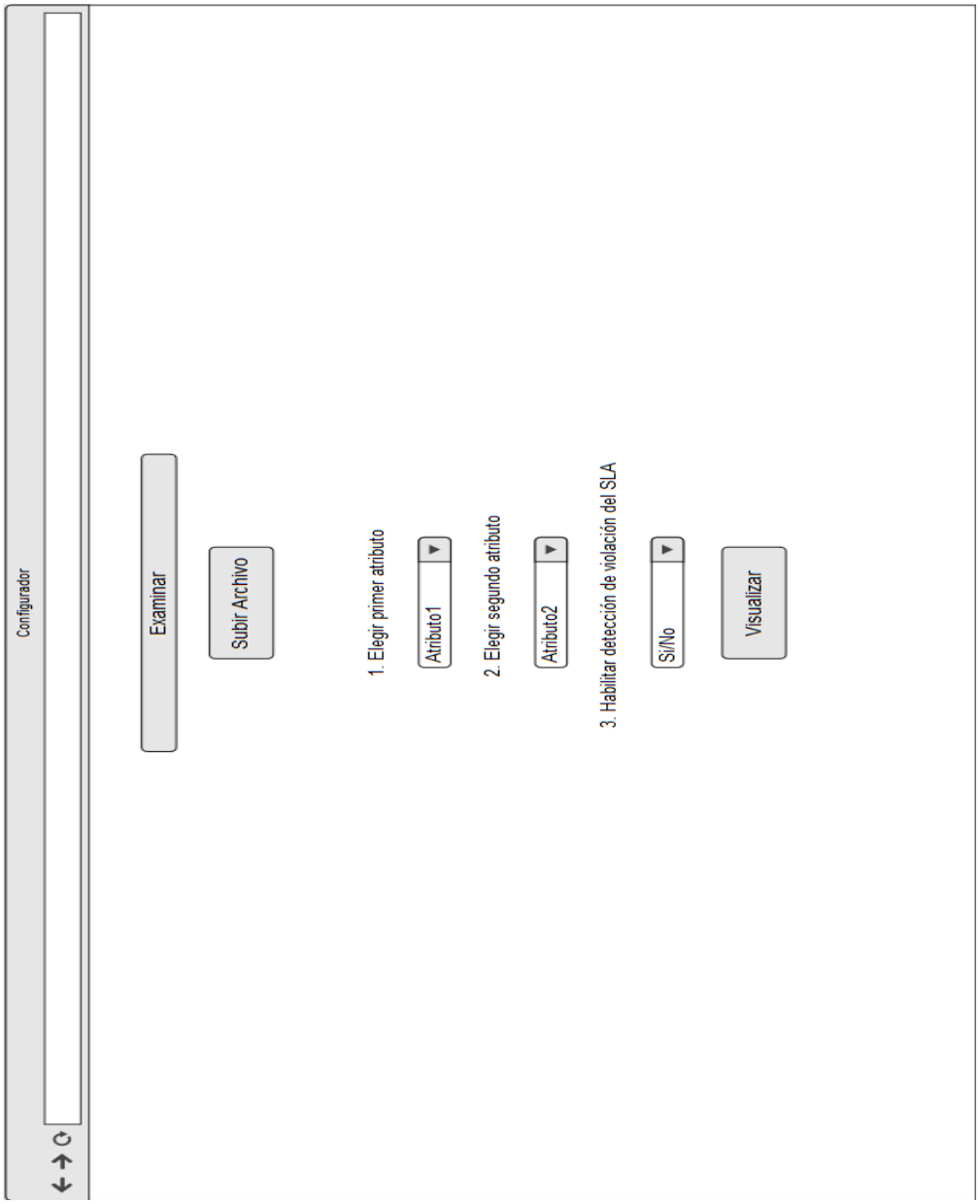

Figura 8. Mockup Configurador

- **Proximidad:** Consiste en agrupar los elementos relacionados de forma conjunta debido a que la proximidad implica relación. Al relacionar varios elementos, estos se convierten en una unidad visual. En la Figura 8 los dos primeros elementos que corresponde a la acción de seleccionar archivo y subirlo, se han situado cerca para que se aprecie que estos dos elementos tienen una relación directa. Los demás elementos tienen una separación similar entre ellos y forman otro grupo visual que se encarga de configurar el monitor.
- **Legibilidad:** Consiste en exhibir la información de una forma que sea fácil de ubicar y leer. El texto que aparece debe de tener un alto contraste, se deben de utilizar combinaciones de colores que favorezcan la lectura, el texto debe ser lo suficientemente grande para que pueda ser leído en distintos tipos de monitores. En la Figura 8 el texto que aparece tiene un tamaño adecuado para que pueda ser visualizado en distinto tipo de monitor, los contrastes de colores favorecen la localización del texto y su lectura y se muestra la cantidad de información justa para no sobrecargar la interfaz.
- **Sencillez:** Consiste en conseguir que los elementos de la interfaz no lleven al usuario a la confusión a la hora de su uso. Se evitan ambigüedades que hacen al usuario dudar sobre la funcionalidad de cierto elemento. En la Figura 8 se pretende mostrar una interfaz simple, intuitiva y con los pasos a seguir bien definidos para que en ningún momento se cree duda sobre qué es lo que realiza cada elemento de la página.

La Figura 9 presenta el mockup del *dashboard* de visualización. A continuación, se comentarán los correspondientes requisitos de diseño de interfaces que cumple:

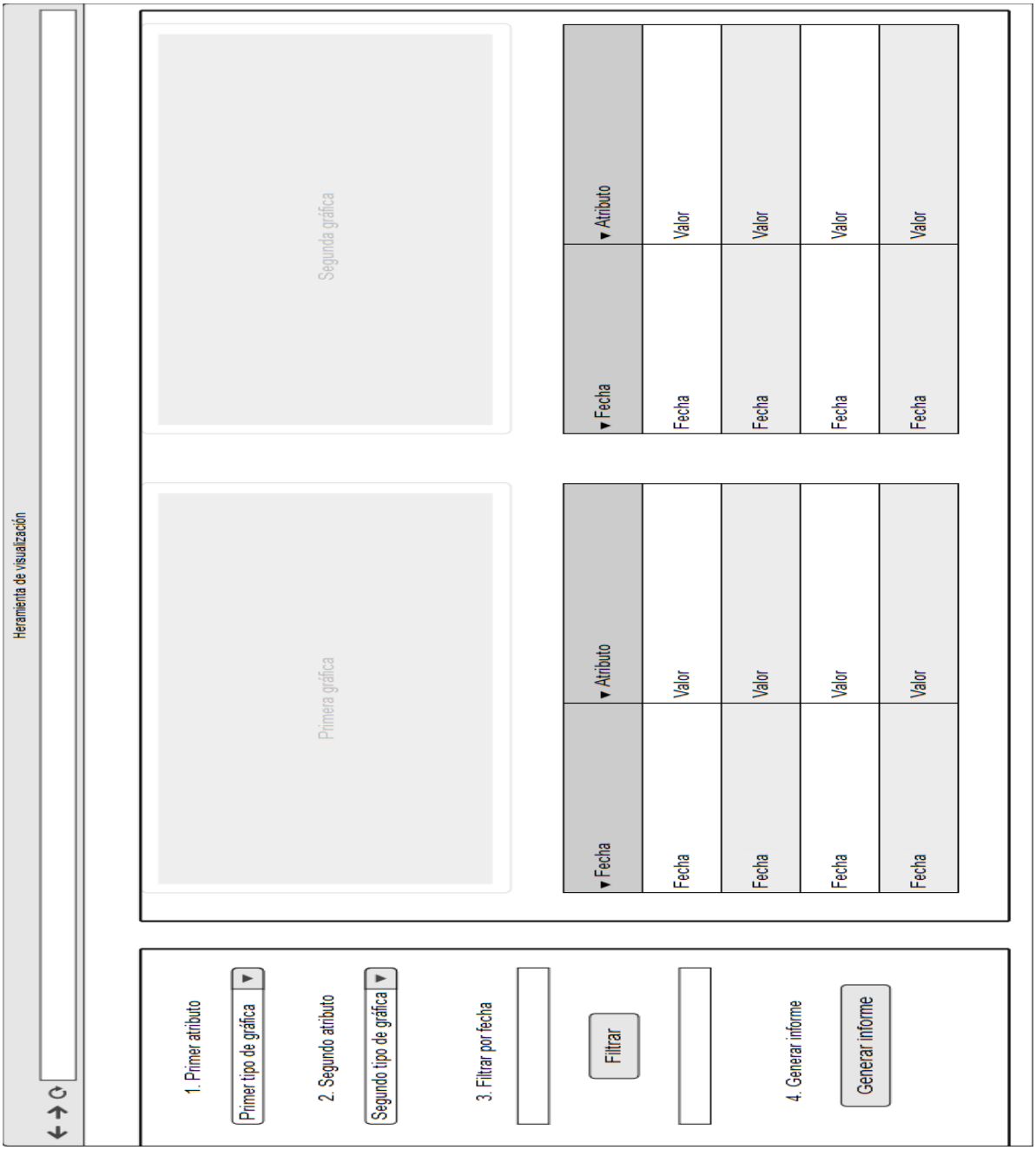

Figura 9. Mockup de la herramienta de visualización

- **Contraste:** En la Figura 9 se puede apreciar en la parte de la izquierda que pertenece al configurador, que los elementos de la misma clase como las listas desplegables, las áreas de texto o los botones tienen un diseño distinto para facilitar al usuario su localización y destacar que los elementos que contiene la página no son todos iguales. Por otro lado, las gráficas y las tablas de la derecha tienen el mismo diseño y color.
- **Repetición:** En la Figura 9 se repite el diseño y el color de los elementos de la misma clase. Las listas desplegables son un poco más grandes y destacan más, por otra parte, se sitúan los botones que tiene un tamaño inferior y una separación mayor para que se aprecie que no tienen relación entre sí. En el lado derecho donde se sitúan las gráficas y las tablas, se repiten los colores, el tamaño y las texturas para que se aprecie que pertenecen a la misma clase.
- **Alineación:** En la Figura 9 se puede observar una separación entre el bloque del configurador y la herramienta de visualización. Ambas partes están alineadas y tienen los elementos colocados de forma estructurada para facilitar el uso de la herramienta e indicar que elementos tienen relación directa entre sí.
- **Proximidad:** En la Figura 9, los elementos situados en la parte del configurador forman un grupo visual y los que están situados en la parte de la herramienta de visualización, forman otro grupo. Tanto los elementos del primer como del segundo grupo se sitúan cerca para que se puede identificar al instante cada uno de los grupos visuales.
- **Legibilidad:** En la Figura 9 se muestra un texto de un tamaño adecuado para distinto tipo de monitor, el contraste de colores permite leer el texto sin problemas y no hay demasiada información en la pantalla para no sobrecargar la interfaz.
- **Sencillez:** En la Figura 9 se observa una interfaz sencilla en intuitiva que facilita su uso. En la parte del configurador se ha añadido una serie de pasos para guiar al usuario a la hora de realizar la configuración de los elementos que quiere visualizar.

5.2.1 Estudio de las gráficas elegidas para visualización

En esta sección se va realizar un análisis sobre el uso de distintos tipos de gráficas y cuáles son las más apropiadas para cada tipo de atributo a visualizar.

5.2.1.1 Representación de los datos

Debido a la gran cantidad de datos que se maneja hoy en día, la necesidad de interpretarlos de una forma más rápida y precisa es cada vez mayor. La representación de datos de forma gráfica ayuda a presentar los datos de forma más sencilla donde las conclusiones son fáciles de entender. Evolución de la bolsa, mapas del tiempo, estudios estadísticos son ejemplos de datos representados con gráficos que difícilmente se pueden imaginar en otro formato.

El principal problema que tiene la representación de datos es su objetividad y comprensión. El procesado de la información requiere tomar decisiones sobre que ejes se tendrán en cuenta, periodo a mostrar, comparar o no, etc… Estas decisiones pueden hacer que la representación ofrecida no sea la que la persona que la visualice espera.

Una mala representación de los datos puede llevar a confusión a la hora de interpretarlos, no destacar de forma deseada la información visualizada o incluso convertir en inservible la información mostrada gráficamente.

5.2.1.2 Tipos de datos recogidos por el monitor

A continuación, se van a comentar los datos recogidos por el monitor de Jiménez [20] y cuál es la forma más adecuada de interpretarlos.

Los datos recogidos por el monitor de Jiménez [20] son de tipo cuantitativo. Los datos cuantitativos son datos que miden o calculan un algo para llegar a un punto en su investigación. Lo vital en estos tipos de datos es saber cómo interpretarlos y darles un análisis que tenga lógica para la investigación. Lo que se hace es buscar un instrumento de medición, que tenga validez y confiabilidad, así la investigación no se ve perjudicada por cómo se manipulan estos datos.

Aplicando la definición anterior al caso de estudio, la herramienta de Jiménez extrae los datos decimales para los atributos de fiabilidad, disponibilidad y estabilidad, y datos enteros para el atributo de latencia. El instrumento que se utiliza para generarlos son las distintas métricas aplicadas a cada uno de los atributos mencionados anteriormente y el umbral SLA establece un límite con el cual poder comparar si los datos cumplen o no los requisitos establecidos. De esta manera, al limitar la variación de los datos a un determinado umbral, se establece una forma de interpretar los datos de forma gráfica. El umbral ayuda a la hora de interpretar si los datos están dentro del rango establecido y facilita la localización de los datos que no lo están.

5.2.1.3 Representación de los datos de forma gráfica

En este apartado se va a comentar que tipos de gráfica son más adecuados para representar los datos recogidos que se han mencionado en el apartado anterior.

Debido a que los datos a visualizar son de tipo cuantitativo, se ha optado por el uso de las siguientes gráficas:

- **Gráfico de barras:** Un gráfico de barras es la manera más frecuente de presentar datos cuantitativos. Son fáciles de hacer y de entender. Interpretar las columnas es muy sencillo. Generalmente se otorga dos valores (usando ejes x y y). Para el eje x, tenemos las fechas en las cuales se ha realizado la monitorización de datos; para el eje y, los valores de los atributos monitorizados. En nuestro caso es un tipo de gráfico sencillo debido a que solo tiene una serie de datos y de esta forma se compara la evolución en el tiempo de un mismo atributo. A continuación, en la Figura 10 se muestra un ejemplo de gráfico de barras.

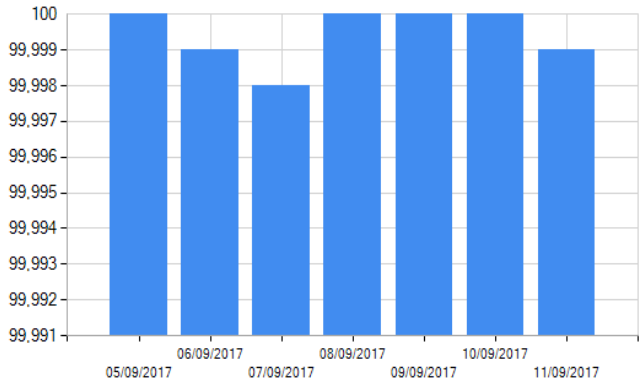

Figura 10. Ejemplo de gráfica de barras

- **Gráfico de líneas:** Un gráfico de líneas es una representación gráfica en un eje cartesiano de la relación que existe entre dos variables reflejando con claridad los cambios producidos**.** Se suelen usar para presentar tendencias temporales. En el eje horizontal se ha de posicionar la variable que indica las unidades de tiempo y en el vertical se introduce la escala de la variable cuya variación en el tiempo queremos ver. A continuación, en la Figura 11 se muestra un ejemplo de gráfico de líneas.

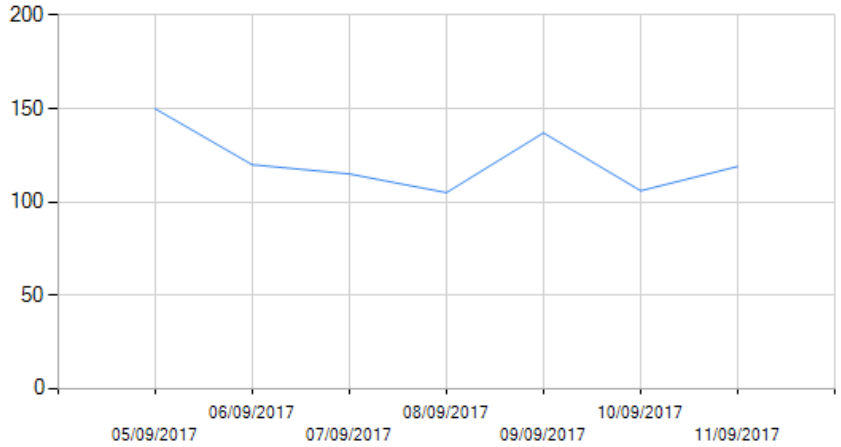

Figura 11. Ejemplo de gráfica de líneas

- **Gráfico de puntos de dispersión:** En este tipo de diagramas, los datos son representados por puntos delimitados por los ejes X e Y, dando la posibilidad de unir los puntos. En este caso tendríamos un gráfico de líneas o líneas curvas. Se utilizan para dejar constancia de los valores concretos dentro del gráfico y se pueden relacionar con los demás valores uniendo los puntos mediante líneas. Es un gráfico muy similar al de líneas. A continuación, la Figura 12 se muestra un ejemplo de gráfico de dispersión.

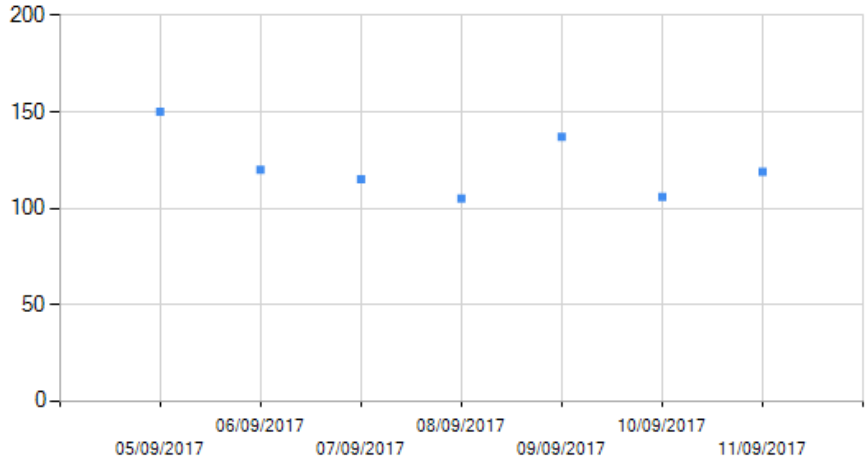

Figura 12. Ejemplo de gráfica de dispersión

- **Gráfico de área:** Los gráficos de áreas son útiles para resaltar la magnitud del cambio a lo largo del tiempo. Los gráficos de áreas también se usan para mostrar la relación de las partes con todo el conjunto. Los gráficos de áreas son como gráficos de líneas que tienen áreas por debajo de las líneas rellenas de colores o modelos. A continuación, en la Figura 13 se muestra un ejemplo de gráfico de área.

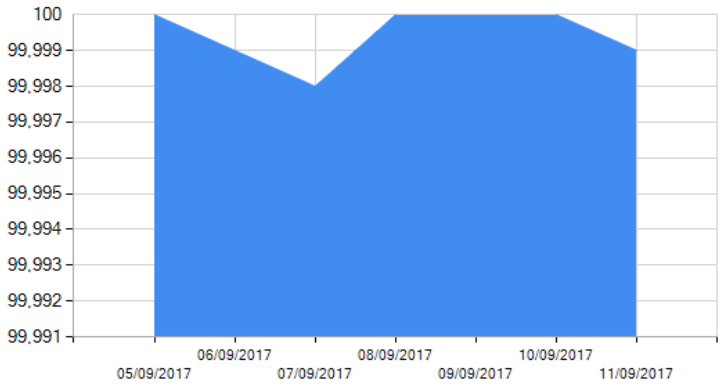

Figura 13. Ejemplo de gráfica de área

5.3 Estructura del Modelo en tiempo de ejecución

En este apartado se muestra una parte del modelo xmi que se toma como entrada a la aplicación y se comentarán las partes que son necesarias para el funcionamiento de la herramienta de visualización.

En la Figura 14 se muestra una parte del fichero xmi de entrada. De este fichero es necesario extraer el nombre del atributo monitorizado y su correspondiente umbral SLA, los cuales se encuentran en las líneas 2, 8, 14 y 20. Los nombres de los atributos de calidad se insertan en las listas desplegables de la figura 8 para que el usuario elija el que desee visualizar.

Los valores de los umbrales especificados en el SLA correspondiente se utilizan cuando el usuario habilita la opción de detección de violación del SLA. Cuando el usuario acaba de establecer la configuración deseada, los valores seleccionados pasan a la siguiente capa de la aplicación para que puedan ser mostrados de forma gráfica.

```
<hasSLO
    nameGuarantee="Reliability 99.999%"
    isOfferedBy="//@hasParties.0/@HasSignP.0"
    corresponds="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperation.0/@hasSLAP.0"
    validityStart="01-05-2015"
    validityEnd="01-05-2016"/>
<hasSLO
    nameGuarantee="Availability 99.995%"
    isOfferedBy="//@hasParties.0/@HasSignP.0"
    corresponds="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperation.0/@hasSLAP.1"
    validityStart="2016-05-01"
    validityEnd="2015-05-01"/>
<hasSLO
    nameGuarantee="Latency &lt;130ms"
    isOfferedBy="//@hasParties.0/@HasSignP.0"
    corresponds="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperation.0/@hasSLAP.2"
    validityStart="2016-05-01"
    validityEnd="2015-05-01"/>
<hasSLO
    nameGuarantee="Stability 99.999%"
    isOfferedBy="//@hasParties.0/@HasSignP.1"
    corresponds="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperation.0/@hasSLAP.3"
    validityStart="2016-05-01"
    validityEnd="2015-05-01"/>
```

Figura 14. Modelo XMI de entrada

5.4 Diseño de la Base de Datos

En este apartado se va a explicar la base de datos que se ha utilizado, cuál es su estructura y cuáles son los datos que almacena.

A la hora de desarrollar la herramienta de visualización, se ha optado por utilizar una base de datos de SQL Server debido a que no se dispone de una licencia de Microsoft Azure. La base de datos local que se ha utilizado simula en gran parte a la base de datos de Microsoft Azure. De hecho, contiene datos de ejemplos de monitorización de servicios en la plataforma Azure recogidos por la herramienta de Jiménez [20].

Al no disponer de licencias necesarias, no ha sido posible hacer una copia exacta de la base de datos de Azure porque no se ha tenido accesos a su estructura interna. Sin embargo, consideramos que esta es una solución viable para realizar pruebas de la herramienta de visualización. Una de las características de los servicios de monitorización en la nube es la capacidad de generar grandes cantidades de datos de manera periódica, lo que puede aumentar mucho los costes de almacenamiento de estos datos en la nube.

La base de datos local de SQL Server proporciona la información necesaria para demostrar el correcto funcionamiento de la herramienta. A continuación, se va a mostrar una imagen de la base de datos utilizada y se van a comentar cada una de sus partes:

	TimeStamp	Value	Name	Service
1	2017-09-05	150	Latency	CoCoMe
2	2017-09-06	120	Latency	CoCoMe
3	2017-09-07	115	Latency	CoCoMe
4	2017-09-08	105	Latency	CoCoMe
5	2017-09-09	137	Latency	CoCoMe
6	2017-09-10	106	Latency	CoCoMe
7	2017-09-11	119	Latency	CoCoMe
8	2017-09-12	129	Latency	CoCoMe
9	2017-09-13	102	Latency	CoCoMe
10	2017-09-14	122	Latency	CoCoMe
11	2017-09-05	100	Reliability	CoCoMe
12	2017-09-06	99,999	Reliability	CoCoMe
13	2017-09-07	100	Reliability	CoCoMe
14	2017-09-08	99,997	Reliability	CoCoMe
15	2017-09-09	99,999	Reliability	CoCoMe
16	2017-09-10	100	Reliability	CoCoMe
17	2017-09-11	99,999	Reliability	CoCoMe
18	2017-09-12	99,995	Reliability	CoCoMe
19	2017-09-13	100	Reliability	CoCoMe
20	2017-09-14	100	Reliability	CoCoMe
21	2017-09-05	100	Stability	CoCoMe
22	2017-09-06	99,999	Stability	CoCoMe
23	2017-09-07	99,998	Stability	CoCoMe
24	2017-09-08	100	Stability	CoCoMe
25	2017-09-09	100	Stability	CoCoMe
26	2017-09-10	100	Stability	CoCoMe
27	2017-09-11	99,999	Stability	CoCoMe

Figura 15. Estructura de la base de datos

La base de datos contiene una tabla llamada MetricsTable que dispone de las siguientes columnas:

- **Timestamp:** Guarda la fecha en la que se ha tomado la medida del atributo. Se mostraría el día, mes y año de la medida. Como futura mejora de la herramienta, también se podría mostrar la hora en la que se ha tomado la medida para aumentar la precisión.
- **Value:** Esta columna guarda el valor que se ha monitorizado en un determinado instante de tiempo.
- **Name:** Nombre del atributo monitorizado.
- **Service:** Nombre del servicio monitorizado.

Como se ha comentado anteriormente, la base de datos puede no ser una copia exacta de la base de datos de Azure debido a que pueda haber elementos internos añadidos por el propio Azure para poder gestionar mejor su base de datos.

El objetivo principal es demostrar que la herramienta es capaz de realizar los casos de uso establecidos en la Figura 15 y para ello la base de datos local proporciona la información necesaria poder comprobar el funcionamiento correcto de la herramienta.

Como futuros trabajos se podría plantear el hecho de establecer una conexión con la base de datos de Microsoft Azure en lugar de la base de datos local.

5.5 Implementación de la herramienta de visualización

En esta sección se explicará el desarrollo de la herramienta de visualización. Se hará una introducción a las herramientas, tecnologías utilizadas en el desarrollo y las librerías que se han utilizado.

5.5.1 Introducción a la herramienta y tecnologías utilizadas

En este apartado se van a definir las tecnologías utilizadas para llevar a cabo el desarrollo de la herramienta de monitorización. A continuación, se van a definir las siguientes categorías según su nivel de detalle y uso:

- Despliegue
- Entorno de desarrollo
- Lenguaje de programación
- Bases de datos
- Frontend

5.5.1.1 Despliegue

Inicialmente se ha elegido la plataforma de despliegue Microsoft Azure, pero debido a que no se dispone de licencias necesarias para realizar el despliegue y posterior uso de la aplicación, finalmente se ha optado por realizar pruebas del funcionamiento en un entorno local. El funcionamiento en local simula completamente el funcionamiento de la aplicación. Se ha replicado la base de datos que se utilizaría en un servidor en la nube y se han añadido datos para poder mostrar el funcionamiento de la aplicación. Posteriormente si se consiguen las licencias necesarias, la aplicación puede ser desplegada en la plataforma de Azure.

5.5.1.2 Entorno de desarrollo

Un entorno de desarrollo integrado o *Integrated Development Environment* (IDE), es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica de usuario. Para

el desarrollo de la herramienta se ha elegido Visual Studio. Soporta múltiples lenguajes de programación y permite el desarrollo en la plataforma web ASP.NET, la cual se va a utilizar para desarrollar la herramienta.

5.5.1.3 Lenguajes de programación

El lenguaje de programación utilizado es C#. Es un lenguaje orientado a objetos desarrollado y estandarizado por Microsoft como parte de la plataforma .NET. Se ha elegido este lenguaje porque se pretende conseguir una integración más fácil con la herramienta desarrollada por Javier Jiménez [18], la cual también utiliza C#.

5.5.1.4 Bases de datos

La base datos utilizada es SQL Server. Es una base de datos local utilizada por el entorno de desarrollo de Visual Studio. Debido a que no se ha obtenido acceso a las licencias de la plataforma de Microsoft Azure, se ha optado por simular la base de datos de la plataforma de despliegue en un entorno local.

5.5.1.5 Frontend

Para la creación de la estructura de la interfaz de usuario se ha utilizado el lenguaje HTML, ASP.NET para el tratamiento de los datos y las funcionalidades de interacción con la aplicación y CSS para modificar la interfaz de la página web.

5.5.1.5.1 HTML

El Lenguaje de Marcas de Hipertexto (*HyperText Markup Language,*HTML) es un lenguaje es se utiliza comúnmente para establecer la estructura y contenido de un sitio web, tanto de texto, objetos e imágenes. Los archivos desarrollados en HTML usan la extensión .htm o .html [11].

5.5.1.5.2 ASP.NET

ASP.NET es un modelo de desarrollo Web unificado que incluye los servicios necesarios para crear aplicaciones Web con el código mínimo. ASP.NET forma parte de .NET Framework y al codificar las aplicaciones ASP.NET tiene acceso a las clases en .NET Framework. El código de las aplicaciones puede escribirse en cualquier lenguaje compatible con el Common Language Runtime (CLR), entre ellos Microsoft Visual Basic, C#, JScript .NET y J#. Estos lenguajes permiten desarrollar aplicaciones ASP.NET que se benefician del Common Language Runtime, seguridad de tipos, herencia, etc [6].

5.5.1.5.3 CSS

El lenguaje de Hojas de Estilo en Cascada (*Cascading Stylesheets,*CSS) es un lenguaje de diseño gráfico utilizado para crear la presentación de un documento estructurado escrito en un lenguaje marcado. Es usado generalmete para establecer el diseño visual de las páginas web, e interfaces de usuario escritas en HTML o XHTML.

5.5.2 Librerías utilizadas en el desarrollo.

En este apartado se van a comentar las librerías que se han utilizado durante el desarrollo de la herramienta.

- **Jquery [13]:** jQuery es una biblioteca multiplataforma de JavaScript, creada, que permite simplificar la manera de interactuar con los documentos HTML, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. jQuery es la biblioteca de JavaScript más utilizada. jQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privados. jQuery ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código.
- **Bootstrap [12]:** Bootstrap es un *Framework* desarrollado por Twitter que simplifica el proceso de creación de diseños web. Esta ofrece una serie de plantillas CSS y ficheros JavaScripts. De la misma forma este *Framework* permite usar estilos y plugins integrados con Jquery.
- **Bootstrap-DatePicker [12]:** Es una librería perteneciente a Bootstrap que permite generar un calendario. Se ha utilizado para crear el calendario en el cual es usuario pueda filtrar por fechas los atributos visualizados.

6. Caso de estudio

En este capítulo se presentará un escenario de uso de la herramienta de visualización propuesta. En primer lugar, se presentará el contexto de este caso, a continuación, las tareas de configuración llevadas a cabo, y para finalizar, el funcionamiento de la herramienta de visualización.

6.1 Presentación del caso

Una empresa de supermercados Market SL ha decidido contratar un servicio de venta en la nube. En él los clientes pueden realizar compras de los productos del supermercado, incluyendo aquellos ítems que deseen en el carrito de la compra y realizando el pago de manera electrónica para más tarde recogerlos en el establecimiento a su elección.

Este servicio se encuentra desplegado en Microsoft Azure por un tercero en la forma de una aplicación web, cuyo servicio se denomina CoCoMe. Se ha decidido contratar este servicio debido a que este modelo ofrece ventajas como la elevada disponibilidad, facilidad de acceso para los clientes, externalización de costes de mantenimiento, posibilidad de proveer con más recursos en temporadas de alta actividad como la navidad, etc.

El supermercado considera críticas para el servicio la *eficiencia* y la *precisión*. Market SL está interesado en que los *tiempos de respuesta del servicio* sean lo suficientemente bajos para que al cliente no le resulte molesto realizar sus compras ya que el objetivo es que el proceso de compra se haga más rápida y cómodamente de lo que sería al realizarlo en un establecimiento físico. A su vez, también es vital que durante el proceso de compra se generen los mínimos errores posibles para evitar frustraciones en el usuario y también errores que puedan suponer un coste a la empresa (pedidos enviados por duplicado, pedidos no finalizados, cobros incorrectos…). Por ello la empresa proveedora del servicio y Market SL incluyen estas condiciones en el acuerdo de servicio.

Con mayor detalle en el acuerdo de nivel de servicio (SLA) se acuerdan los requisitos no funcionales siguientes:

1. La disponibilidad (*Avalilability*) del servicio será superior a 99.995% y será calculada en base a la métrica robustez del servicio (*Robustness of a Service* - ROS) calculada con la siguiente fórmula:

$$ROS = \frac{Avaiable\ Time\ for\ Invocking\ SaaS}{Total\ Time\ for\ Operating\ SaaS} * 10^{6}$$

2. La confiabilidad (Reliability) será medida a través de las operaciones defectuosas por millón. En este caso, el servicio tendrá un máximo de 10 operaciones defectuosas por millón. (99.999% de confiabilidad de servicio). Este requisito será calculado utilizando la métrica correspondiente (*Defective Operations per Million-DPM*).

$$DPM = \frac{Operations\ Attempted - OperartionsSuccesful}{Operations\ Attempted} * 10^{6}$$

3. La latencia (*Latency*) máxima del servicio será de 130 milisegundos y será calculada con la siguiente fórmula:

$$Latency = Request\ Execution\ Time + Response\ Time$$

4. La estabilidad(*Stability*): la estabilidad del servicio es la tolerancia a riesgos de efectos inesperados por modificaciones. No tiene que ser menor que 99.999%.

Estos requisitos han sido monitorizados por la herramienta de Jiménez [20] y serán mostrados de forma gráfica al usuario por la herramienta de visualización desarrollada. Como se ha explicado en el apartado anterior, debido al problema con las licencias de Microsoft Azure, los resultados serán simulados en una base de datos local, de esta forma se podrá mostrar la funcionalidad de la herramienta de visualización.

6.2 Configuración de herramienta de visualización

Para la creación del modelo en tiempo de ejecución se utilizará el configurador de monitorización implementado en el trabajo final de grado de Jiménez [20], la cual seguirá los siguientes pasos, los cuales no vamos a entrar en detalle ya que se trata de una implementación externa a este trabajo:

Primero. El usuario introduce los datos de autentificación para monitorizar el servicio CoCoMe. Los cuales se detallan a continuación:

- **Plataforma de los servicios a monitorizar**: Microsoft Azure
- **Nombre de la instancia**: CoCoMe

- **Deployment Id**: fdc45005307044e0a00073e118ab71b0
- **ConnectionString:** Eby8vdM02xNOcqFlqUwJPLlmEtlCDXJ1OUzFT50uSRZ6IFsuFq2UVErCz4I6tq/K1SZFPTOtr/KBHBeksoGMGw==

Segundo. El usuario introduce el archivo correspondiente al Acuerdo de Nivel de Servicio, cuyos requisitos no funcionales (NFR) y sus umbrales se exponen en la tabla siguiente:

NFR	Límite
Confiabilidad	>99.999
Tiempo de resupuesta del servicio	<130 ms

Tabla 2. Caso de Uso - métricas SLA

Tercero. El usuario escogerá los requisitos no funcionales del SLA y les asignará una métrica independiente de la plataforma. Para ello escogerá una característica presente en el modelo de calidad SaaS, en el caso de que esta tenga una subcaracterística escogerá una, se escogerá un atributo de calidad y una métrica independiente de la plataforma que permita medir dicho atributo de calidad. En concreto, el usuario ha realizado la clasificación especificada en la siguiente tabla:

NFR	Característica	Subcaracterística	Atributo	Métrica	Operacionalización
Latencia	Eficiencia del rendimiento	Comportamiento del tiempo	Tiempo de Respuesta	Latencia	Tiempo de Solicitud + Tiempo de Solicitud de respuesta
Confiabilidad	Confiabilidad	-	Tolerancia a fallos	Millones de Operaciones fallidas (DP;)	Operaciones Fallidas / Nº Total de Operaciones

Tabla 3. Caso de Estudio - Asignación independiente de plataforma

Cuarto. El usuario asigna los contadores de la plataforma Azure a cada uno de los operandos de las métricas independientes de plataforma. Esto permitirá extraer los datos del servicio en tiempo de ejecución, calcular la métrica y proporcionar una medida para el atributo de calidad de interés. Las asignaciones realizadas para cada uno de los requisitos no funcionales se describen en la tabla siguiente:

NFR	Función dependiente de plataforma
Latencia	$Latencia = Request\ Execution\ Time + Response\ Time$
Confiabilidad	$DPM = \frac{Operations\ Atempted - Succesful}{Operations\ Atempted} * 10^6$

Tabla 4. Caso de Estudio - Asignación Requisitos no funcionales con funciones dependientes de plataforma

Al finalizar los cuatro pasos anteriores se creará el modelo de calidad en tiempo de ejecución (fichero xmi) el cual será la entrada de la herramienta de visualización. Podemos ver el modelo en el apartado 1 del Anexo.

Una vez la herramienta de Jiménez [20] termine el proceso de monitorización de los servicios cloud, se procederá ejecutar la herramienta de visualización para poder mostrar de forma gráfica los datos recopilados. El usuario ejecutará la aplicación y en la pantalla principal escogerá la configuración deseada.

6.2.1 Configuración inicial

En la página principal mostrada en la Figura 16, una vez el usuario haya subido el correspondiente modelo en tiempo de ejecución a visualizar (archivo xmi), se procederá a elegir la configuración deseada.

En el paso 1 el usuario selecciona el modelo de entrada en formato xmi que desee importar a la herramienta.

En el paso 2 se elige el primer atributo de calidad a visualizar. Según el caso de estudio presentado en el apartado anterior, el usuario podría elegir las siguientes opciones: *Availability, Latency, Reliability y Stability.*

En el paso 3 se elige el segundo atributo a visualizar que contiene las mismas opciones que en el paso 2.

En el paso 4 el usuario tiene la opción de configurar una Alerta para que le avise en caso de producirse una violación del SLA.

Finalmente, después de elegir la configuración deseada, se pulsará el botón Visualizar. El cual redirigirá al usuario a la siguiente página en la que se mostrarán los datos monitorizados de forma gráfica.

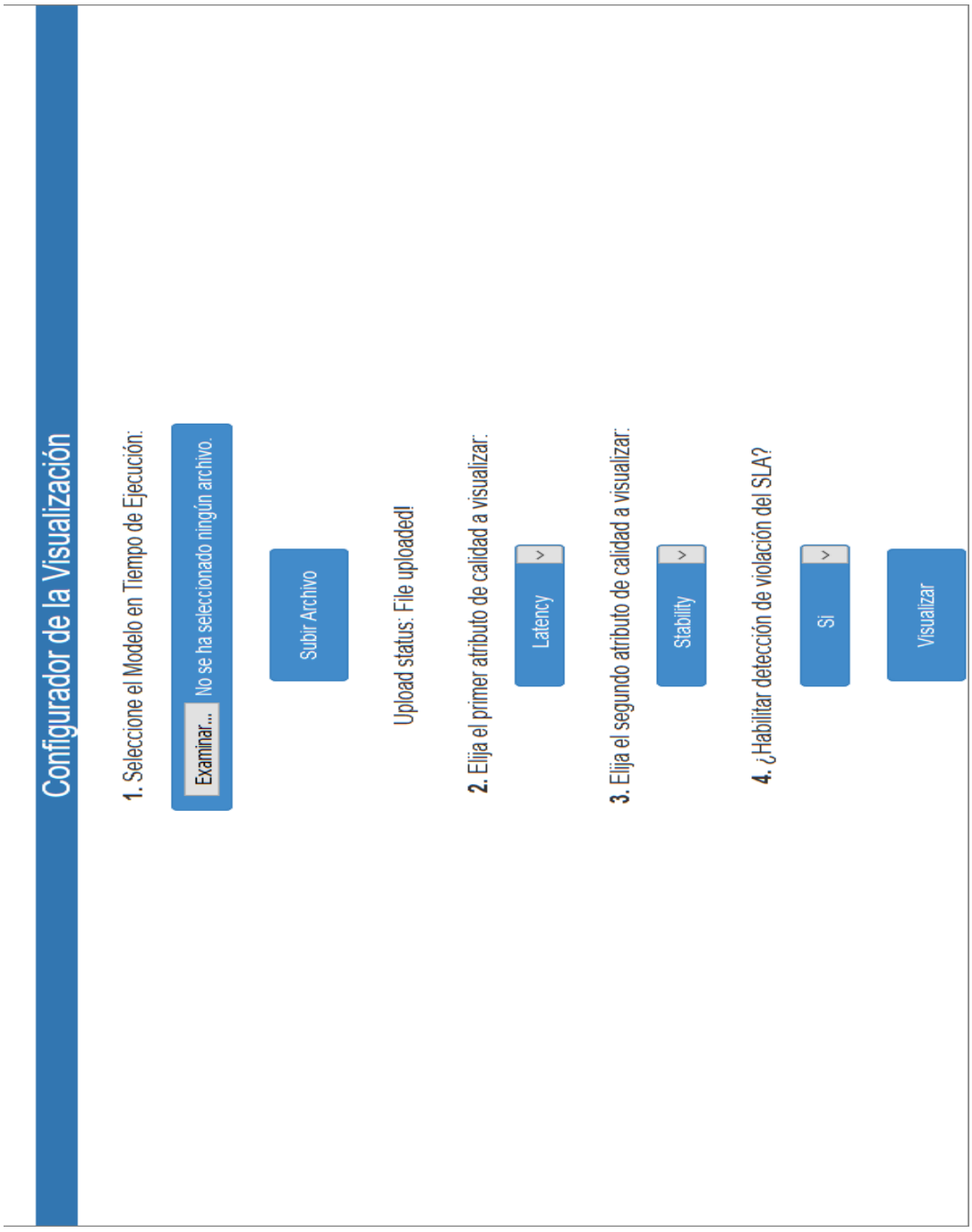

Figura 16. Configurador de la visualización

Después presionar el botón Visualizar de la página principal mostrada en la Figura 16, la aplicación muestra una nueva página mostrando los datos gráficamente.

La Figura 17 muestra los datos recopilados por la herramienta de monitorización de Jiménez [20]. En este caso no se ha habilitado la opción de detección de violación del SLA, por lo tanto, la herramienta no avisará al usuario de que se ha incumplido el SLA.

En el apartado configurador, el usuario puede seleccionar el tipo de gráfica que quiere visualizar, filtrar los datos por fecha y generar un informe.

En el paso 1 el usuario puede seleccionar la gráfica a visualizar correspondiente al nombre del atributo de calidad elegido. Dentro del desplegable de elección de gráfica, el usuario dispone de las siguientes opciones: *Line Chart, Column Chart, Point Chart, Area Chart.*

En el paso 2 el usuario puede elegir las gráficas correspondientes al segundo atributo de caliddad a visualizar. Dispone de las mismas opciones que en el paso 1.

En el paso 3 el usuario puede filtrar por fecha los datos mostrados en las gráficas y en las tablas. Como máximo es posible visualizar 7 datos tanto en la gráfica como en la tabla.

Esta limitación se debe a que en la base de datos puede haber una gran cantidad de información a visualizar. Si toda esta información se muestra de forma simultánea, puede llegar a confundir al usuario a la hora de interpretar la información y reduciría de forma considerable la usabilidad de la herramienta.

Finalmente, en el paso 4 el usuario podrá generar un informe en formato PDF de los elementos visualizados.

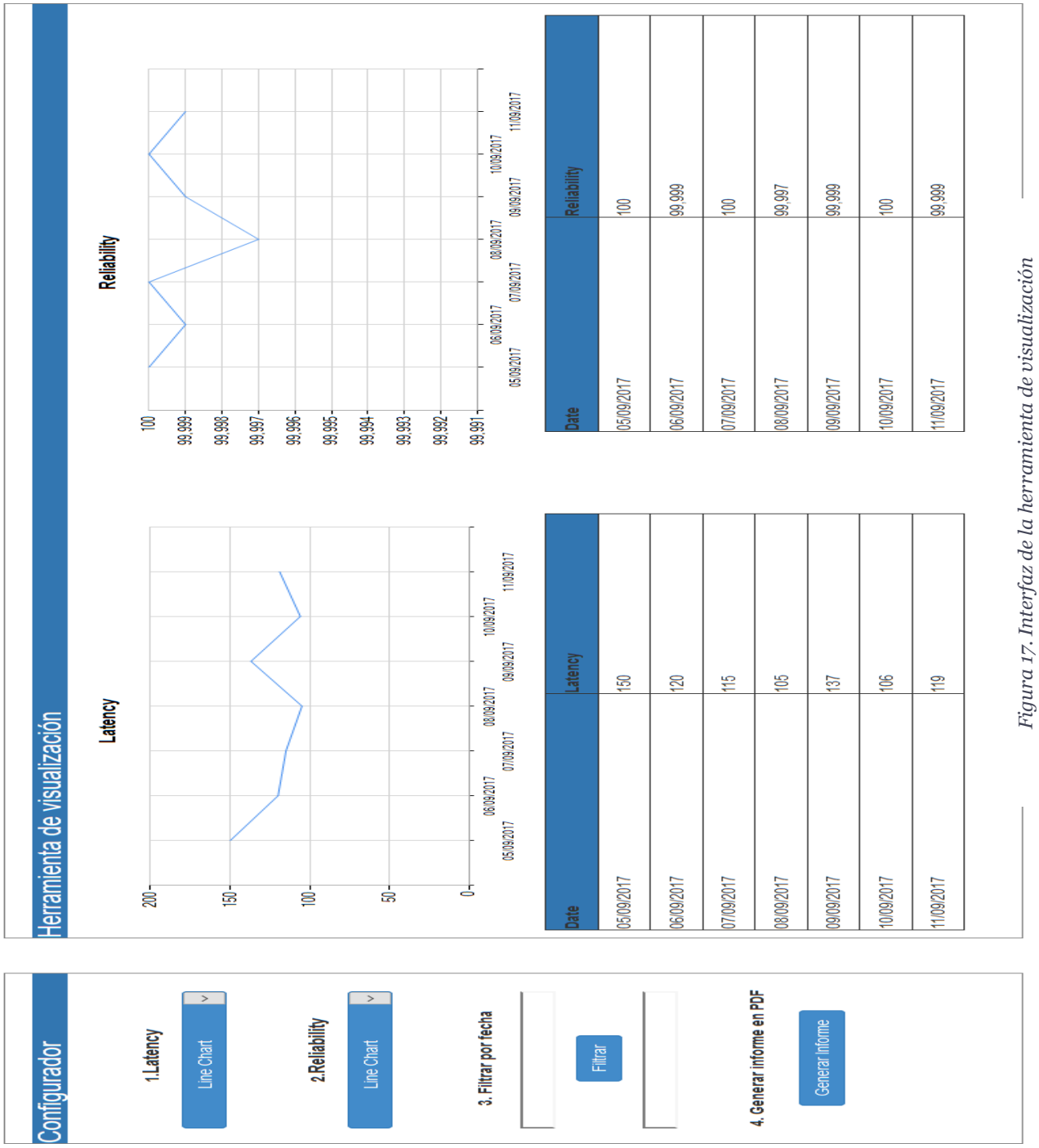

Figura 17. Interfaz de la herramienta de visualización

El la Figura 18 se muestra un ejemplo de visualización de datos de medición con otro tipo de gráficas elegidas por el usuario y el filtrado de datos por fechas.

Para el atributo *Latency* se ha elegido la gráfica de puntos (*Point Chart*) y para el segundo, *Reliability*, la gráfica de área (*Area Chart*). También se ha utilizado el filtro por fecha. La fecha se puede introducir manualmente o se puede seleccionar del calendario desplegable mostrado en la Figura 18.

La Figura 19 muestra un ejemplo de detección de violación del SLA en el caso de que el usuario habilite esta opción en el paso 4 de la Figura 16. Si se detecta una violación del SLA, se muestra una alerta en la pantalla indicando el atributo en el cual se ha detectado.

Además, en la tabla asociada al atributo, se muestra una línea en rojo para facilitar al usuario la búsqueda del dato que no cumple con los requisitos SLA establecidos en el fichero xmi de entrada.

La Figura 20 muestra un ejemplo del informe en formato PDF generado por la herramienta. El usuario puede generar en cualquier momento un informe con los datos visualizados. El informe contiene ambas gráficas y sus correspondientes tablas. El hecho de generar un informe que incluya gráficas y tablas facilita al usuario localizar la información deseada e interpretar los datos de forma más sencilla.

6.3 Conclusiones

Este caso práctico ha permitido poner en práctica la herramienta de visualización y probarla en un ejemplo de monitorización de un servicio cloud que ha sido desplegado en la plataforma Azure y cuyos datos de medición han sido recogidos por la herramienta de Jiménez [20], que carecía de un interfaz de usuario que permita la visualización de los datos de monitorización.

Una vez obtenidos los datos es necesario que una aplicación web sea capaz de utilizar y tratar dichos datos para la visualización en pantalla (mediante un dashboard configurable) y la obtención de informes sintéticos que permitan al usuario la confirmación del cumplimiento del SLA o la explicación de su violación e interpretación del mismo. Por lo tanto, el trabajo presentado en este proyecto ha consistido en definir una herramienta que de soporte al proceso de análisis y toma de decisiones. Como trabajo futuro, se pretende aplicar la herramienta de visualización a otros ejemplos más complejos, con más atributos de calidad y métricas.

Figura 18. Interfaz de la herramienta de visualización con gráfica de área y de puntos

Figura 19. Interfaz de la herramienta de visualización con alerta por violación del SLA

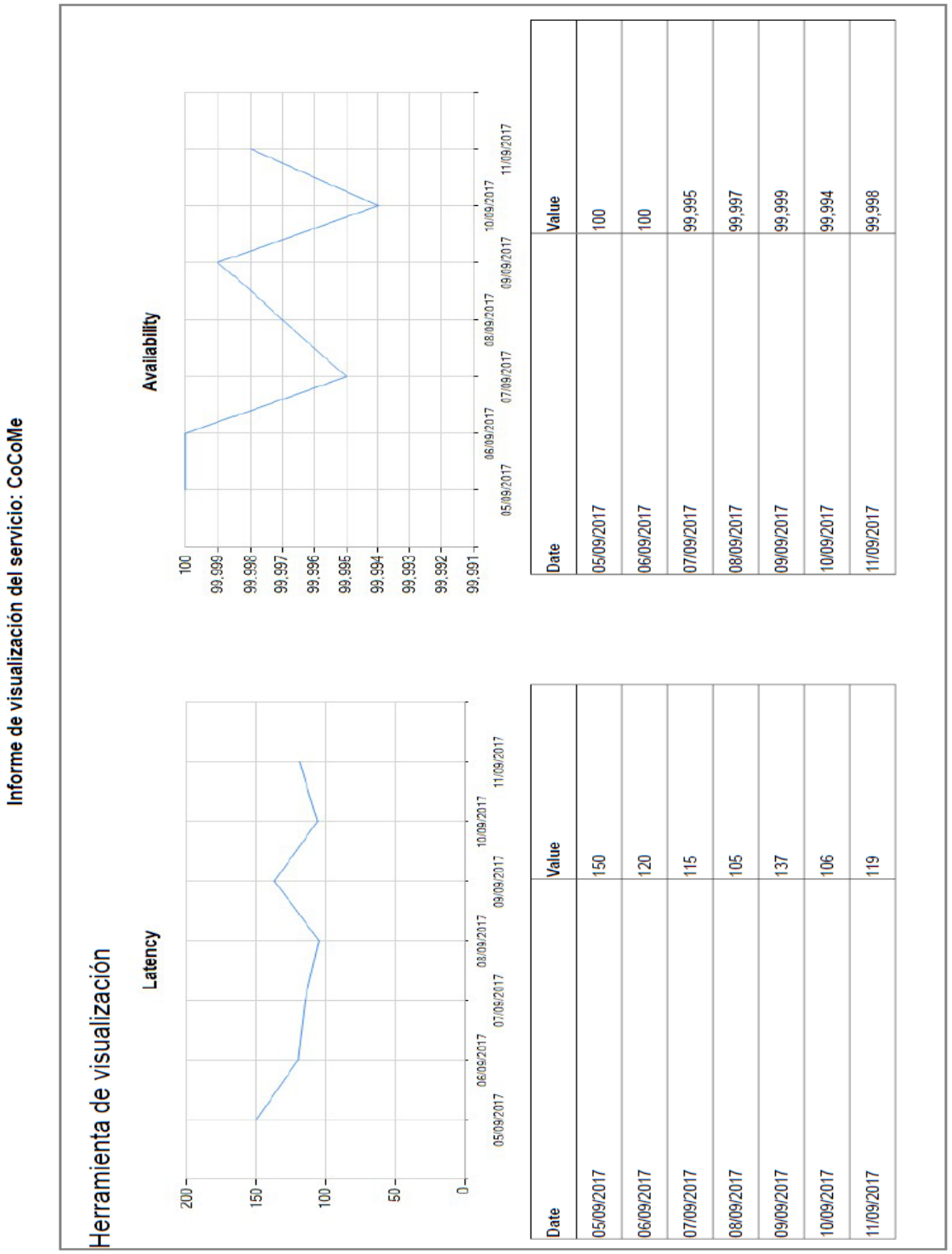

Figura 20. Informe visualización

6.3 Diseño elegido para la visualización de datos

En esta sección se va explicar porque se ha optado por el uso de determinadas gráficas y los *DataGrid* para representar la información recopilada por el monitor de Jiménez [20].

6.3.1 Gráficas utilizadas

A continuación, se van explicar el uso de las gráficas utilizadas en el desarrollo de la herramienta:

- **Gráfica de líneas (*Line Chart*):** este tipo de gráficas se utiliza para mostrar un mismo tipo de dato y su evolución, por lo que son útiles para mostrar tendencias. Utilizar este tipo de gráfica en la herramienta le permite al usuario identificar fácilmente cualquier variación en los datos visualizados.
- **Gráfica de columnas (*Column Chart*):** es el tipo de gráfico más común. La escala de valores se muestra normalmente en el eje vertical y la altura de cada columna se corresponde con cada valor. El uso de este tipo de gráfica le permite al usuario tener otra perspectiva sobre la forma en la que se muestran los datos. Igual que en la gráfica anterior, los datos son fáciles de interpretar y se puede observar cualquier cambio sin ninguna dificultad.
- **Gráfica de puntos (*Point Chart*):** esta gráfica se emplea para reflejar la relación entre dos variables. En nuestro caso sería la relación entre la fecha y los valores del atributo de calidad a mostrar.
- **Gráfica de área (*Area Chart*):** se ha utilizado este tipo de gráfica porque es similar a la gráfica de líneas, pero con colores por debajo de las líneas para ayudar a su identificación.

6.3.2 Uso del DataGrid

El *DataGrid* proporciona una tabla personalizable para mostrar datos de solo lectura. Se ha utilizado en la herramienta de visualización como apoyo a los distintos tipos de gráficos que el usuario puede visualizar. *DataGrid* permite al usuario localizar e interpretar los datos de una forma más simple. Por otra parte, cuando el usuario habilita la detección de violación del SLA, la columna que no cumpla el umbral establecido en el fichero xmi de entrada, se resaltará en color rojo para llamar la atención del usuario. De esta forma el usuario podrá detectar de forma inmediata si hay algún dato que no cumpla el SLA.

7. Conclusiones y trabajos futuros

En este capítulo se recogen las conclusiones del presente trabajo y a continuación, se van a presentar los futuros trabajos que se pueden realizar para mejorar el prototipo desarrollado.

7.1 Conclusiones

Hoy en día los servicios de computación en la nube son cada vez más presentes. Esto se debe a que esta tecnología ofrece comodidad y flexibilidad tanto al usuario como al proveedor. Ofrece la posibilidad de pagar solamente por los servicios que se utilizan, la posibilidad de acceder al servicio desde cualquier parte siempre y cuando se disponga de una conexión a Internet, ofrecer una alta disponibilidad y fiabilidad del servicio, etc. Sin embargo, todos estos aspectos deben de ser controlados para comprobar que realmente se ofrece el servicio que se ha solicitado.

En este aspecto, la herramienta de Jiménez [20] proporciona la capacidad de monitorizar la calidad de este tipo de servicios y comparar los resultados con los valores establecidos previamente en el SLA. De esta manera, en caso de que no se haya cumplido el acuerdo establecido, el usuario puede reclamar al proveedor el incumplimiento del SLA presentándole los datos recogidos por el monitor.

Con la herramienta presentada en este trabajo se pretende facilitar al usuario la visualización e interpretación de los datos recogidos por el monitor independientemente de la plataforma. De esta forma se facilita la comprensión del estado actual de los servicios y se facilita la búsqueda de atributos que en algún momento no han cumplido con los requisitos establecidos en el SLA y también se ofrece la posibilidad de generar un informe para poder presentarlo al proveedor del servicio.

7.2 Tecnologías

En este apartado se exponen algunos de los inconvenientes presentados durante la realización del trabajo relacionados con las tecnologías utilizadas.

7.2.1 Entorno de desarrollo .NET

Esta tecnología era completamente nueva para mí, tanto el entorno de desarrollo como los lenguajes de programación utilizados. A la hora de

aprender el lenguaje C# no se han presentado muchas dificultades debido a que es un lenguaje orientado a objetos igual que el lenguaje Java que se estudió a lo largo de la carrera. Por otra parte, el lenguaje ASP.NET si que era completamente nuevo. En este caso sí que se han presentado bastantes dificultades a la hora de utilizarlo. Se ha invertido una gran parte del tiempo en comprender el funcionamiento del lenguaje y en aprender cual es la mejor forma de combinarlo con el lenguaje C#. Además, se ha utilizado un entorno de desarrollo nuevo para mí, en el cual también se ha invertido un tiempo de aprendizaje.

A pesar de todas las dificultades que se han presentado, el hecho de trabajar con una tecnología y entorno nuevo ha resultado ser una experiencia enriquecedora. Se ha adquirido un poco de experiencia en desarrollo web con tecnología .NET lo cual puede ser de gran utilidad en futuros proyectos.

7.2.2 Lectura y extracción de datos del fichero XMI.

Una de las partes más complicadas fue entender la estructura del modelo en tiempo de ejecución para conseguir leer y extraer correctamente los datos del fichero xmi que lo representa. Los ficheros xmi son extensos y tienen una estructura compleja, esto ha supuesto una dificultad adicional a la hora de extraer los datos necesarios para el funcionamiento correcto de la herramienta de visualización.

7.3 Trabajos futuros

En este apartado se comentarán algunos de los posibles trabajos a realizar:

- **Desplegar la herramienta:** Como ya se ha comentado en apartados anteriores, la herramienta no se ha desplegado en ninguna plataforma cloud debido a que no se dispone de las licencias necesarias. En ese caso sería interesante desplegar la herramienta en un entorno cloud para comprobar que sigue funcionando correctamente.
- **Incrementar la precisión:** El prototipo de la herramienta visualiza los datos monitorizados en un día concreto. Sería interesante añadir más precisión a la herramienta permitiendo mostrar información de los atributos monitorizados por horas, minutos y segundos. Los ciclos de extracción y visualización de estos datos en ocasiones necesitan ser muy cortos (del orden de segundos) de manera que puedan detectarse correctamente ciertas situaciones sensibles al tiempo, por

ejemplo tiempos de respuesta excesivamente largos solo podrán controlarse con el contador *Request Time* si el valor de este se extrae en cortos espacios de tiempo pues solo obtiene el valor de la última petición respondida de manera que el tiempo de extracción pueda ocultar este tipo de resultados si resulta ser demasiado largo.

- **Añadir más tipos de gráficas:** El prototipo de la herramienta tiene una serie de gráficas las cuales el usuario puede visualizar en cualquier momento. Como mejora se podrían añadir más tipos de gráficas para que el usuario disponga de más formas de visualizar la información.
- **Mejorar la interfaz:** Uno de los objetivos de la herramienta es tener una interfaz simple e intuitiva para que el usuario no tenga ninguna dificultad a la hora de utilizarla. Como una posible mejora se propone modificar la interfaz para que sea más atractiva y usable.
- **Evaluación de la herramienta.** Se pretende realizar un estudio con un grupo de usuarios finales para evaluar la usabilidad de la herramienta propuesta.

8.Referencias

1. **Cedillo P., Jimenez-Gomez J., Abrahão S., Insfrán E.** Towards a Monitoring Middleware for Cloud Services. 12th IEEE International Conference on Services Computing (IEEE SCC 2015). 27 June - 2 July, 2015. Vols. New York, USA. pp. 451-458, IEEE Computer Society.

2. **New York, USA. pp. 451-458, IEEE Computer Society.**

3. **Amazon CloudWatch. [En línea] [Citado el: 6 de Septiembre de 2017.] https://aws.amazon.com/es/cloudwatch/.**

4. Amazon Web Services. [En línea] [Citado el: 10 de Septiembre de 2017.] https://aws.amazon.com/es/.

5. AppDynamics. [En línea] [Citado el: 15 de Septiembre de 2017.] https://www.appdynamics.com/.

6. ASP.NET. [En línea] [Citado el: 18 de Septiembre de 2017.] https://msdn.microsoft.com/es-es/library/4w3ex9c2(v=vs.100).aspx.

7. Azure Diagnostics. [En línea] [Citado el: 18 de Septiembre de 2017.] https://docs.microsoft.com/en-us/azure/monitoring-and-diagnostics/azure-diagnostics.

8. CooperEgg. [En línea] [Citado el: 18 de Septiembre de 2017.] https://www.idera.com/infrastructure-monitoring-as-a-service.

9. Dynatrace. [En línea] [Citado el: 25 de Septiembre de 2017.] https://www.dynatrace.es/.

10. Google App Engine. [En línea] [Citado el: 25 de Septiembre de 2017.] https://cloud.google.com/appengine/docs/.

11. HTML. [En línea] [Citado el: 3 de Octubre de 2017.] https://es.wikipedia.org/wiki/HTML.

12. Librería Bootstrap. [En línea] [Citado el: 4 de Octubre de 2017.] https://getbootstrap.com/.

13. Librerías Jquery. [En línea] [Citado el: 4 de Octubre de 2017.] https://jquery.com/.

14. QoS. *Wikipedia*. [En línea] [Citado el: 5 de Octubre de 2017.] https://es.wikipedia.org/wiki/Calidad_de_servicio.

15. Stackdriver Monitoring. [En línea] [Citado el: 17 de Octubre de 2017.] https://cloud.google.com/monitoring/?hl=es.

16. SysDig. [En línea] [Citado el: 14 de Octubre de 2017.] https://sysdig.com/.

17. VRealize Hyperic. [En línea] [Citado el: 19 de Octubre de 2017.] https://www.vmware.com/products/vrealize-hyperic.html.

18. 3ZQU3RR4, J053M4. La red infinita. [En línea] 2014 de Mayo de 2014. [Citado el: 27 de Noviembre de 2017.] https://laredinfinita.wordpress.com/2014/05/10/cluster-y-grid-colecciones-de-sistemas/.

19. Feinberg, Leonid. CloudEndure. [En línea] [Citado el: 15 de Noviembre de 2017.] https://www.cloudendure.com/blog/5-cloud-experts-predict-cloud-computing-trends-2017/..

20. Gómez, Javier Jiménez. Middleware para la monitorización de la calidad de servicios cloud. [En línea] [Citado el: 23 de Noviembre de 2017.] https://goo.gl/HPVWdh.

21. Rosales, Andrés Páez. Monitor de Calidad de Servicios en Google. [En línea] 2016. [Citado el: 28 de Noviembre de 2017.] https://goo.gl/GbAZJA.

22. Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, Matei Zaharia. Communications Acm. [En línea] Abril de 2010. [Citado el: 22 de Noviembre de 2017.] https://goo.gl/mcLLUE.

Anexo

1. Modelo en Tiempo de Ejecución (Fichero xmi)

```
<?xml version="1.0" encoding="ASCII"?>
<MonitoringRequirementsNS:MonitoringRequirementsModel
    xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:MonitoringRequirementsNS="MonitoringRequirementsNSURI"
    xsi:schemaLocation="MonitoringRequirementsNSURI
MonitoringRequirementsModel.ecore"
    name="MonitoringRequirementsModelAuctionSite">
  <hasParties
      description="Signature and Supporting Parties">
    <hasSP name="Supporting Auditing Part"
        address=""
        isSponsoredBy="//@hasParties.0/@HasSignP.0"/>
    <HasSignP
        name="Service Provider"
        address=""
        offers="//@hasObligations.0/@hasActionG.0
//@hasObligations.0/@hasSLO.0             //@hasObligations.0/@hasSLO.1
//@hasObligations.0/@hasActionG.1         //@hasObligations.0/@hasSLO.2
//@hasObligations.0/@hasActionG.2     //@hasObligations.0/@hasActionG.3
//@hasObligations.0/@hasSLO.4         //@hasObligations.0/@hasActionG.4
//@hasObligations.0/@hasSLO.5 //@hasObligations.0/@hasActionG.5"

isCommunicated="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOpe
ration.0/@hasSLAP.0/@hasPC.0
//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperation.0/@hasSLA
P.0/@hasPC.1
//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperation.0/@hasSLA
P.1/@hasPC.0
//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperation.0/@hasSLA
P.1/@hasPC.1
//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperation.0/@hasSLA
P.2/@hasPC.0
//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperation.0/@hasSLA
P.2/@hasPC.1
//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperation.0/@hasSLA
P.3/@hasPC.0
//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperation.0/@hasSLA
P.3/@hasPC.1
//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperation.0/@hasSLA
P.4/@hasPC.0
//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperation.0/@hasSLA
P.4/@hasPC.1"
        isSponsorOf="//@hasParties.0/@hasSP.0"/>
    <HasSignP
        name="Service Customer"
        offers="//@hasObligations.0/@hasSLO.3"

isCommunicated="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOpe
ration.0/@hasSLAP.0/@hasPC.2
//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperation.0/@hasSLA
P.1/@hasPC.2
//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperation.0/@hasSLA
```

```
P.2/@hasPC.2
//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperation.0/@hasSLA
P.3/@hasPC.2
//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperation.0/@hasSLA
P.4/@hasPC.2"/>
  </hasParties>
  <hasObligations
      nameObligation="ObligationsPartAuctionSite">
    <hasSLO
        nameGuarantee="Reliability 99.999%"
        isOfferedBy="//@hasParties.0/@HasSignP.0"

corresponds="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperat
ion.0/@hasSLAP.0"
        validityStart="01-05-2015"
        validityEnd="01-05-2016"/>
    <hasSLO
        nameGuarantee="Availability 99.995%"
        isOfferedBy="//@hasParties.0/@HasSignP.0"

corresponds="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperat
ion.0/@hasSLAP.1"
        validityStart="2016-05-01"
        validityEnd="2015-05-01"/>
    <hasSLO
        nameGuarantee="Latency &lt;130ms"
        isOfferedBy="//@hasParties.0/@HasSignP.0"

corresponds="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperat
ion.0/@hasSLAP.2"
        validityStart="2016-05-01"
        validityEnd="2015-05-01"/>
    <hasSLO
        nameGuarantee="Stability 99.999%"
        isOfferedBy="//@hasParties.0/@HasSignP.1"

corresponds="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperat
ion.0/@hasSLAP.3"
        validityStart="2016-05-01"
        validityEnd="2015-05-01"/>
    <hasSLO
        nameGuarantee="Service Accuracy"
        isOfferedBy="//@hasParties.0/@HasSignP.0"

corresponds="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperat
ion.0/@hasSLAP.0"
        validityStart="2016-05-01"
        validityEnd="2015-05-01"/>
    <hasSLO
        nameGuarantee="Transactions Per Hour"
        isOfferedBy="//@hasParties.0/@HasSignP.0"

corresponds="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperat
ion.0/@hasSLAP.4"
        validityStart="2015-05-01"
        validityEnd="2016-05-01"/>
    <hasActionG
        nameGuarantee=">99.999"
        isOfferedBy="//@hasParties.0/@HasSignP.0"
```

```
corresponds="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperat
ion.0/@hasSLAP.0"
        name="GuaranteeOfReliability"/>
    <hasActionG
        nameGuarantee=">99.995"
        isOfferedBy="//@hasParties.0/@HasSignP.0"

corresponds="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperat
ion.0/@hasSLAP.1"
        name="GuaranteeofAvailability"/>
    <hasActionG
        nameGuarantee="&lt;130"
        isOfferedBy="//@hasParties.0/@HasSignP.0"

corresponds="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperat
ion.0/@hasSLAP.2"
        name="Latency"/>
    <hasActionG
        nameGuarantee=">99.999"
        isOfferedBy="//@hasParties.0/@HasSignP.0"

corresponds="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperat
ion.0/@hasSLAP.3"
        name="Stablity"/>
    <hasActionG
        nameGuarantee="=100%"
        isOfferedBy="//@hasParties.0/@HasSignP.0"

corresponds="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperat
ion.0/@hasSLAP.0"
        name="Service Accuracy"/>
    <hasActionG
        nameGuarantee="=10000"
        isOfferedBy="//@hasParties.0/@HasSignP.0"

corresponds="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperat
ion.0/@hasSLAP.4"
        name="Transactions per Hour"/>
  </hasObligations>
  <hasServiceDefinition
      name="AuctionService">
    <hasServiceObjects
        name="AuctionServiceA">
      <hasSchedule
          name="businessDaySchedule"
          periodStart="2015-05-01"
          periodEnd="2016-05-01"
          intervalMinutes="1440"

isIn="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperation.0/@
hasSLAP.0/@hasMetric.1/@hasFunction"
          intervalSeconds="0"/>
      <hasSchedule
          name="hourlySchedule"
          periodStart="2015-05-01"
          periodEnd="2016-05-01"
          intervalMinutes="60"

isIn="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperation.0/@
hasSLAP.4/@hasMetric.0/@hasFunction"/>
```

```
      <hasSchedule
          name="5MinutesSchedule"
          periodStart="2015-05-01"
          periodEnd="2016-05-01"
          intervalMinutes="5"/>
      <hasSchedule
          name="availabilitySchedule"
          intervalMinutes="1"

isIn="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperation.0/@
hasSLAP.1/@hasMetric.0/@hasFunction"/>
      <hasOperation
          name="BidNow"

operSLAPar="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperati
on.0/@hasSLAP.4"

canHas="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperation.0
/@hasSLAP.1/@hasMetric.0
//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperation.0/@hasSLA
P.0/@hasMetric.0
//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperation.0/@hasSLA
P.2/@hasMetric.0
//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperation.0/@hasSLA
P.0/@hasMetric.1
//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperation.0/@hasSLA
P.4/@hasMetric.0">
        <hasSLAP
            name="Reliability"
            has="//@hasObligations.0/@hasActionG.0
//@hasObligations.0/@hasSLO.0             //@hasObligations.0/@hasSLO.4
//@hasObligations.0/@hasActionG.4">
          <hasPC
              communicates="//@hasParties.0/@HasSignP.0"/>
          <hasPC
              communicates="//@hasParties.0/@HasSignP.0"
              type="push"/>
          <hasPC
              communicates="//@hasParties.0/@HasSignP.1"
              type="pull"/>
          <hasMetric
              name="DPM"
              metricURI="URI DPM"
              metricMacroName="Defective Operations Per Million"
              has="//@hasUnits.0"

belongstoOp="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperat
ion.0">
            <hasFunction
                name="DPM"
                formula="((OperationsAttempted-
OperationsSuccessful)/OperationsAttempted)*10^6"/>
          </hasMetric>
          <hasMetric
              name="Service Accuracy"
              metricURI="ServiceAccuracyURI"
              metricMacroName="ServiceAccuracyMetricMacro"
              has="//@hasUnits.2"

belongstoOp="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperat
ion.0">
```

```
            <hasFunction
                name="SA"

hasA="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasSchedule.0"
                formula="Number of Correct Responses / Total Number of
Requests"/>
          </hasMetric>
        </hasSLAP>
        <hasSLAP
            name="Availability"
            has="//@hasObligations.0/@hasSLO.1
//@hasObligations.0/@hasActionG.1">
          <hasPC
              communicates="//@hasParties.0/@HasSignP.0"/>
          <hasPC
              communicates="//@hasParties.0/@HasSignP.0"
              type="push"/>
          <hasPC
              communicates="//@hasParties.0/@HasSignP.1"
              type="pull"/>
          <hasMetric
              name="Service Robustness"
              metricURI="ServiceRoubstnessMetricURI"
              metricMacroName="ServiceRobustnessMacroName"
              has="//@hasUnits.2"

belongstoOp="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperat
ion.0">
            <hasFunction
                name="ROS"

hasA="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasSchedule.3"
                formula="(available time for invoking SaaS) / total
time for operationg SaaS"/>
          </hasMetric>
        </hasSLAP>
        <hasSLAP
            name="Latency"
            has="//@hasObligations.0/@hasSLO.2
//@hasObligations.0/@hasActionG.2">
          <hasPC
              communicates="//@hasParties.0/@HasSignP.0"/>
          <hasPC
              communicates="//@hasParties.0/@HasSignP.0"
              type="push"/>
          <hasPC
              communicates="//@hasParties.0/@HasSignP.1"
              type="pull"/>
          <hasMetric
              name="Latency"
              metricURI="URIMacroName"
              metricMacroName="LatencyMacroName"
              has="//@hasUnits.3"

belongstoOp="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperat
ion.0">
            <hasFunction
                name="Latency"
                formula="sum(Request Response Time) / Number of
Requests"/>
          </hasMetric>
```

```
        </hasSLAP>
        <hasSLAP
            name="Stability"
            unit="Percentage"
            has="//@hasObligations.0/@hasSLO.3
//@hasObligations.0/@hasActionG.3">
          <hasPC
              communicates="//@hasParties.0/@HasSignP.0"/>
          <hasPC
              communicates="//@hasParties.0/@HasSignP.0"
              type="pull"/>
          <hasPC
              communicates="//@hasParties.0/@HasSignP.1"
              type="push"/>
        </hasSLAP>
        <hasSLAP
            name="Efficiency"
            unit="TransactionsPerHour"

SLAPOper="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperation
.0"
            has="//@hasObligations.0/@hasSLO.5
//@hasObligations.0/@hasActionG.5">
          <hasPC
              communicates="//@hasParties.0/@HasSignP.0"/>
          <hasPC
              communicates="//@hasParties.0/@HasSignP.0"
              type="push"/>
          <hasPC
              communicates="//@hasParties.0/@HasSignP.1"
              type="pull"/>
          <hasMetric
              name="Transactions Per Hour"
              metricURI="TransactionsPerHourURI"
              metricMacroName="Transactions Per Hour"
              has="//@hasUnits.1"

belongstoOp="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperat
ion.0">
            <hasFunction
                name="Transactions per Hour"
                resultType="integer"

hasA="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasSchedule.1"
                formula="TransactionsPerHour"/>
          </hasMetric>
        </hasSLAP>
      </hasOperation>
    </hasServiceObjects>
  </hasServiceDefinition>
  <hasUnits
      name="Segundo"

isInMetric="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperati
on.0/@hasSLAP.0/@hasMetric.0"/>
  <hasUnits
      name="Minuto"

isInMetric="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperati
on.0/@hasSLAP.4/@hasMetric.0"/>
  <hasUnits
```

```
      name="Percentage"

isInMetric="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperati
on.0/@hasSLAP.1/@hasMetric.0
//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperation.0/@hasSLA
P.0/@hasMetric.1"/>
  <hasUnits
      name="Milliseconds"

isInMetric="//@hasServiceDefinition.0/@hasServiceObjects.0/@hasOperati
on.0/@hasSLAP.2/@hasMetric.0"/>
</MonitoringRequirementsNS:MonitoringRequirementsModel>
```