Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Modelo Comparativo de Plataformas Cloud y Evaluación de Microsoft Azure, Google App Engine y Amazon EC2

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: José Miguel Álvarez Vañó

Tutor: Silvia Abrahao Gonzales

Cotutor: César Emilio Insfrán Pelozo

2017 - 2018

Agradecimientos

En primer lugar, quiero agradecer todo el apoyo que Silvia me ha ofrecido tanto en las asignaturas que he realizado con ella durante la carrera, como en este trabajo. He podido disfrutar de un trabajo interesante que me ha hecho aprender mucho y que seguro me servirá en mi futuro.

Quiero agradecer a mi familia todo el apoyo durante mi vida académica. Sin ellos no habría llegado tan lejos. Y, por último, agradecer a mis compañeros por estos años de carrera en los que hemos podido compartir algo que tanto nos gusta.

Resumen

Existe una gran cantidad de proveedores de servicios en la nube siendo los más importantes Microsoft, Google y Amazon. Otros proveedores también son Rackspace, IBM, Oracle, Salesforce, etc. Un aspecto relevante para los desarrolladores y clientes es conocer las características de estos proveedores para tener información objetiva de cómo elegir entre una plataforma u otra dependiendo de sus objetivos y necesidades. En este proyecto se ha realizado un estudio para determinar las características de calidad relevantes de las plataformas cloud y se ha propuesto un modelo de calidad basado en la ISO/IEC 25010 para guiar a los usuarios en la comparación y selección de dichas plataformas. El modelo está soportado por un sistema de recomendación que permite a los usuarios especificar sus objetivos y comparar plataformas cloud mediante un conjunto de atributos y métricas de calidad. Este modelo se ha aplicado a un estudio para comparar las plataformas Microsoft Azure, Google App Engine y Amazon Elastic Compute Cloud (EC2) permitiendo la evaluación de sus características de calidad más relevantes.

Palabras clave: Cloud Computing, Plataformas Cloud, Modelos de Calidad, ISO/IEC 25010, Evaluación

Abstract

There is a large number of cloud service providers being the most important Microsoft, Google and Amazon. Other providers are also Rackspace, IBM, Oracle, Salesforce, etc. A relevant aspect for developers and customers is to determine the characteristics of these providers to have objective information on how to choose between one platform or another depending on their objectives and needs. In this project, we have carried out a study to determine the relevant quality characteristics of cloud platforms and a quality model based on ISO/IEC 25010 has been proposed to guide users in the comparison and selection of these platforms. The model is supported by a recommendation system that allows users to specify their objectives and compare cloud platforms through a set of quality attributes and metrics. This model has been applied to a case study which compares the Microsoft Azure, Google App Engine and Amazon Elastic Compute Cloud (EC2) platforms allowing the evaluation of its most relevant quality characteristics.

Keywords: Cloud Computing, Cloud Platforms, Quality Models, ISO/IEC 25010, Evaluation

Resum

Hi ha una gran quantitat de proveïdors de serveis en el núvol sent els més importants Microsoft, Google i Amazon. Altres proveïdors també són Rackspace, IBM, Oracle, Salesforce, etc. Un aspecte rellevant per als desenvolupadors i clients és conèixer les característiques d'aquests proveïdors per tenir informació objectiva de com triar entre una plataforma o una altra depenent dels seus objectius i necessitats. En aquest projecte s'ha realitzat un estudi per determinar les característiques de qualitat rellevants de les plataformes cloud i s'ha proposat un model de qualitat basat en la ISO / IEC 25010 per guiar els usuaris en la comparació i selecció d'aquestes plataformes. El model està suportat per un sistema de recomanació que permet als usuaris especificar els seus objectius i comparar plataformes cloud mitjançant un conjunt d'atributs i mètriques de qualitat. Aquest model s'ha aplicat a un estudi per comparar les plataformes Microsoft Azure, Google App Engine i Amazon Elastic Compute Cloud (EC2) permetent l'avaluació de les seves característiques de qualitat més rellevants.

Paraules claus: Cloud Computing, Plataformes Cloud, Models de qualitat, ISO/IEC 25010, Avaluació

Índice

Índice de Figuras

Índice de tablas

1. Introducción

La computación en la nube es un modelo de prestación y consumo de servicios que ofrece muchas ventajas a las empresas (alta disponibilidad, elasticidad, máximo aprovechamiento de recursos, etc.) que se traducen en requisitos de calidad que deben ser cumplidos por el servicio [2].

Sin embargo, la elección de una plataforma de cloud computing puede ser complicada desde el punto de vista del usuario, ya que no está claro las ventajas y limitaciones de cada una de ellas y por lo tanto hay cierta incertidumbre a la hora de realizar esta selección.

Dentro del cloud computing existen distintos roles, 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.

Existen varios proveedores de servicios en la nube, siendo Microsoft, Google y Amazon los más importantes. Otros proveedores relevantes son Rackspace, IBM, Oracle, Salesforce, etc. Por lo tanto, existe una alta competencia de proveedores de servicios en la nube, basada en los precios a los que ofertan los recursos. Sin embargo, los proveedores deben considerar otro factor diferenciador más allá del precio de oferta: la calidad de sus servicios.

Un aspecto importante para los desarrolladores y clientes es conocer las características de estos proveedores para tener información objetiva sobre cómo elegir una plataforma en la nube en función de sus necesidades [13]. Por ejemplo, si los niveles de rendimiento no cumplen las expectativas o se vuelven impredecibles, los clientes rechazarán el servicio o evitarán su adopción. Por otro lado, si se reúnen o se exceden las expectativas, la reputación de un proveedor en la nube incrementará favorablemente y por tanto sus servicios tendrán una mayor acogida y por ende utilización [22]. De ahí que, los proveedores de servicios deberán realizar fuertes inversiones económicas para mantenerse en el negocio, ya que cualquier mejora en la calidad de servicio será percibida y valorada por el cliente final.

Además, según el informe “¿Cielos despejados? El nivel de adopción de la nube” de *Intel Security*, actualmente hay una tendencia cada vez mayor a incorporar los servicios cloud en los productos software, lo que conlleva una clara necesidad a proporcionar una seguridad que disminuya ciertos riesgos tales como las fugas de datos [10].

1.1. Motivación

La motivación de este trabajo es debida a la incertidumbre que tienen los usuarios a la hora de elegir una plataforma cloud, ya que no tienen fuentes de información que no sean las propias plataformas.

Se necesitan mecanismos que permitan a los usuarios describir las características de los proveedores y plataformas cloud de modo que puedan comparar entre varias alternativas y decidir qué plataforma es la más adecuada dependiendo de sus necesidades.

Las necesidades de los clientes, como más adelante se explica con más detalle, pueden ser reducir costes pagando únicamente por los recursos utilizados, acceder al servicio desde cualquier lugar, una escalabilidad flexible y ágil o evitar tener que mantener la infraestructura pudiendo emplear estos recursos a otros objetivos.

El beneficio que puede proporcionar este proyecto es que el usuario tenga un mayor control de los servicios que contrata, las distintas opciones que existen en el mercado y de esta manera aumentar la competitividad entre proveedores de servicios cloud.

1.2. Objetivos

El propósito de este trabajo es definir y desarrollar un sistema de recomendación basado en un modelo de calidad que permita a los usuarios comparar de forma objetiva las distintas plataformas cloud que hay actualmente en el mercado, seleccionando las características/atributos más relevantes para él, de manera que, mediante el uso de este modelo, reciba una serie de recomendaciones sobre qué plataforma es la que más se adecua a sus necesidades.

Los modelos de calidad son referencias que pueden ser utilizada por las organizaciones para obtener directrices que les guíen hacia un aumento de calidad [11]. Un modelo de calidad proporciona un mecanismo en torno al cual se establece el sistema para la evaluación de la calidad de un producto software. En este modelo, se determinan las características de calidad que se van a tener en cuenta a la hora de evaluar las propiedades de un producto software determinado. En los modelos de calidad, la calidad de define de forma jerárquica. El estándar ISO/IEC 25010 [12] define un modelo de calidad genérico para productos software. Es el estándar más reciente sobre calidad de productos software.

La ventaja de los modelos de calidad es que la calidad se convierte en algo concreto, que se puede definir, que se puede medir, y, sobre todo, que se puede planificar. Los modelos de calidad ayudan también a comprender las relaciones que existen entre diferentes características de un producto software. En este proyecto, utilizaremos el ISO/IEC 25010 [12] para definir un modelo de calidad que describa y descomponga las características de las plataformas cloud.

El objetivo principal podríamos dividirlo en 4 objetivos específicos que se detallan a continuación:

1. **Estudio de las plataformas cloud y determinación de características/atributos de calidad relevantes**

Este objetivo consiste en estudiar las distintas plataformas cloud para entender cuáles son las características y atributos de calidad más relevantes y la relación entre ellas. Para ello, se llevará a cabo un estudio del estado del arte.

2. **Definición de un modelo de calidad para la selección de plataformas cloud**

El objetivo es crear un modelo de calidad, alineado con la ISO/IEC 25010 [1], que descomponga la calidad de las plataformas cloud en características, subcaracterísticas, atributos y métricas que permitan medir estas características.

También se pretende relacionar cada característica, atributo o métrica con los roles para los que son relevantes (proveedores cloud, consumidores cloud, bróker, etc.), permitiendo así, que el modelo pueda ser usado desde distintos puntos de vista.

3. **Definición de un sistema de recomendación**

El objetivo es definir la arquitectura de un sistema de recomendación que utilice el modelo de calidad para la selección de plataformas cloud. Este sistema será construido a partir de un análisis acerca de los distintos sistemas de recomendación existentes.

4. **Implementación de la herramienta**

El objetivo es definir un prototipo que permita al usuario emplear el sistema de recomendación propuesto de una manera sencilla, permitiéndole elegir su rol y las características más relevantes para él.

5. **Evaluación del sistema de recomendación**

El objetivo es realizar un caso de estudio para evaluar algunas plataformas cloud seleccionadas. Estas plataformas serán posteriormente evaluadas utilizando el modelo de calidad y sistema de recomendación definidos.

1.3. Estructura del documento

En este capítulo se ha realizado una introducción en la que se ha presentado lo que ha motivado a la realización de este trabajo, los objetivos que se esperan alcanzar.

En el Capítulo 2 se introduce la computación en la nube y las plataformas cloud, sus características más esenciales, los estándares actuales, las plataformas existentes, etc.

En el Capítulo 3 se presenta el modelo de calidad propuesto, destacando la extensión realizada sobre el estándar ISO/IEC 25010, realizando una selección de métricas para la selección de plataformas cloud.

En el Capítulo 4 se encuentra el sistema de recomendación, que se apoya en el modelo de calidad presentado en el capítulo 3.

En el Capítulo 5 se presenta un caso de estudio a modo de ejemplo en el que usamos el sistema de recomendación y el modelo de calidad para la selección de plataformas cloud.

Finalmente, en el Capítulo 6 se presentan las conclusiones de este trabajo.

2. Cloud Computing

La computación en la nube es un modelo para habilitar el acceso ubicuo, bajo demanda, por red a un conjunto compartido de recursos de computación configurables (por ejemplo, redes, servicios, almacenamiento, aplicaciones y servicios) que pueden ser rápidamente provistos y publicados con un esfuerzo mínimo de manejo o interacción del proveedor de servicios [21].

La computación en la nube presenta cinco características esenciales, tres modelos de servicio y cuatro modelos de despliegue.

2.1. Características esenciales

El modelo de cloud computing tiene las siguientes características esenciales:

- **Pago por uso:** Una de las principales características del cloud computing es la posibilidad de pagar únicamente por los recursos que utilizas, manteniendo en todo momento un control de los gastos y el consumo realizado [6] [7] [8].

- **Acceso sin restricciones:** Debido a que todos los servicios se encuentran alojados en la red, los usuarios tienen la posibilidad de acceder a estos servicios de cloud computing desde cualquier lugar mediante una conexión a internet, ya sea a través de un teléfono móvil, un ordenador, tablet, etc. Esto facilita mucho la accesibilidad a los servicios dando una gran libertad al usuario para poder acceder a los servicios como mejor le convenga [6] [7] [8].

- **Agilidad en la escalabilidad:** Los servicios cloud ofrecen una gran escalabilidad y elasticidad con una adaptación a las necesidades del usuario con gran agilidad. El usuario puede en cualquier momento aumentar o disminuir la cantidad de recursos contratados y pagar únicamente por los recursos utilizados en cada momento gracias al pago por uso [6] [7] [8].

- **Reparto de recursos:** El proveedor de los recursos del servicio cloud tiene una única aplicación que es compartida por muchos usuarios, de manera que cada usuario tiene asignados una cantidad concreta de recursos de acuerdo con sus necesidades, permitiendo la optimización de los recursos del servicio cloud [6] [7] [8].

- **Abstracción:** Gracias a la capacidad de abstraer los recursos proporcionados del servicio cloud del equipo del cliente, los usuarios no necesitan preocuparse del mantenimiento de la infraestructura [6].

2.2 Modelos de Servicio

Dentro de los servicios cloud generalmente se definen 3 tipos de modelos:

- **Infraestructura como servicio (IaaS):** En este modelo de servicio el proveedor proporciona a los clientes toda la infraestructura necesaria. De manera que los clientes no necesitan adquirir hardware directamente, si no que pagan según el uso que hagan de la infraestructura que les ofrece el proveedor [9].

- **Plataforma como Servicio (PaaS):** En este modelo de servicio el cliente hace uso de una plataforma que le ofrece el proveedor y le sirve para realizar el desarrollo, gestión y distribución de sus aplicaciones. En este caso el cliente no debe preocuparse por la gestión de la infraestructura, ya que de esta se encarga el proveedor, gestionando tanto la seguridad como el software de los servidores [9]. Con el PaaS, lo que tenemos en la nube no es un servicio, sino la capacidad de utilizar una plataforma de cloud computing para lo que queramos, normalmente crear servicios y hostearlos.

- **Software como Servicio (SaaS):** En este modelo de servicio los usuarios pueden hacer uso de software sin la necesidad de tenerlo instalado en sus propios dispositivos. Los usuarios acceden a través de una API a este software y no necesitan gestionar el software, de manera que los proveedores realizan el mantenimiento del software sin necesidad de actualizar los dispositivos de los clientes [9]. Este uso puede ser gratuito, de pago o una solución intermedia donde el uso del software es gratuito hasta cierto punto, pasando a ser de pago si se quiere usar durante más tiempo o se quieren más funcionalidades. Un ejemplo del software como servicio o SaaS es el del correo electrónico (p. ej., Gmail de Google).

2.3. Modelos de Despliegue

El despliegue consiste en todo el conjunto de actividades que hacen posible que un producto software esté disponible para su uso. Los modelos de despliegue se diferencian principalmente por la propiedad, el tamaño y el acceso. De acuerdo con estas características encontramos 4 tipos de modelo de despliegue:

- ***Public Cloud* (Nube Pública)**: La infraestructura está disponible y compartida por el público general. Este modelo proporciona escalabilidad y elasticidad según las necesidades del usuario en cada momento y pagando únicamente por los recursos utilizados.

- ***Private Cloud* (Nube Privada)**: En este caso una única organización hace uso de la plataforma cloud. En este modelo se pierde la ventaja del pago por recursos utilizados ya que la plataforma está reservada en exclusiva a una organización. Sin embargo, se gana un mayor control y seguridad al poder gestionar toda la plataforma.

- ***Community Cloud* (Nube comunitaria)**: La infraestructura está disponible exclusivamente para un conjunto de organizaciones que comparten unos objetivos comunes. La infraestructura puede ser propiedad de la comunidad o de una tercera parte.

- ***Hybrid Cloud* (Nube Híbrida)**: En este modelo la infraestructura es una combinación de dos o más de los modelos anteriores que varía según las necesidades del negocio.

2.4. Estándares para la Computación en la Nube

Existen estándares que rigen la computación en la nube y las plataformas distribuidas, en lo referente a la especificación de un formato de virtualización abierto, vocabulario, arquitectura de referencia, interoperabilidad, acuerdos de nivel de servicio, entre otros [66].

Teniendo como base el estándar ISO/IEC 17788:2014, en el cual se provee una visión sobre las definiciones y términos de la nube, siendo esta recomendación aplicable a todos los tipos de organizaciones. En este estándar se definen términos similares a los ya mencionados e introducidos por el NIST, más ciertos términos que son propios de las tecnologías cloud.

Como ejemplo podríamos citar conceptos relacionados a la disponibilidad, confidencialidad, seguridad de la información, integridad, entre otros. Términos que serán abordados a lo largo de este trabajo y que, dentro de este estándar, han sido recopilados incluso desde diversas fuentes. Más información incluido un glosario se puede encontrar en el estándar ISO/IEC 17788 [20].

2.5. Estándar ISO/IEC 25000 (SQuARE)

Aspectos como el hecho de tener dos normas complementarias: ISO/IEC 9126 (Calidad del producto software) e ISO/IEC 14598 (Evaluación del producto software), han propiciado el impulso necesario para la elaboración de la norma ISO/IEC 25000 (2005) conocida como SQuaRE (Software Quality Requirement Evaluation). El objetivo perseguido en la creación de esta norma es dar un paso hacia un conjunto de estándares organizados de manera más lógica, enriquecidos con nuevas aportaciones y unificados con respecto a las normas anteriores para ser capaces de cubrir los dos principales procesos: *especificación de requisitos de calidad del software* y *evaluación de la calidad del software*, soportados por un proceso de medición.

SQuaRE se centra exclusivamente en el producto software estableciendo criterios para su especificación, su medición y su evaluación. Es decir, básicamente se trata de una unificación y revisión de los estándares ISO/IEC 9126 e ISO/IEC 14598 [75].

La norma está compuesta de cinco divisiones bien diferenciadas:

- División de gestión de calidad (ISO/IEC 2500n): Los estándares que forman esta división definen todos los modelos comunes, términos y referencias a los que se alude en las demás divisiones de SQuaRE.

- División del modelo de calidad[1] (ISO/IEC 2501n): El estándar que conforma esta división presenta un modelo de calidad detallado, incluyendo características para la calidad interna, externa y en uso.

- División de mediciones de calidad (ISO/IEC 2502n): Los estándares pertenecientes a esta división incluyen un modelo de referencia de calidad del producto software (ver Fig.2.8), definiciones matemáticas de las métricas de calidad y una guía práctica para su aplicación. Presenta aplicaciones de métricas para la calidad de software interna, externa y en uso.

- División de requisitos de calidad (ISO/IEC 2503n): Los estándares que forman parte de esta división ayudan a especificar los requisitos de calidad. Estos requisitos pueden ser usados en el proceso de especificación de requisitos de calidad para un producto software que va a ser desarrollado o como entrada para un proceso de evaluación.

- División de evaluación de la calidad (ISO/IEC 2504n): Estos estándares proporcionan requisitos, recomendaciones y guías para la evaluación de un producto software, tanto si la llevan a cabo evaluadores, como clientes o desarrolladores. Basándose en la ISO/IEC 14598.

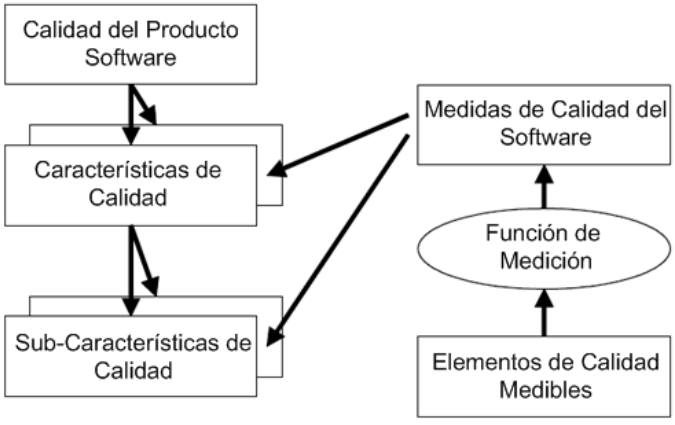

Figura 1. Modelo de medición de la calidad del producto software según SQuaRE.

[1] La versión de la ISO 25010 consultada en este trabajo es la de Julio 2008, por lo que en un futuro pueden existir variaciones en cuanto al modelo de calidad se refiere, al estar esta parte en fase de revisión a fecha de octubre 2009.

Las principales ventajas con respecto a sus predecesores, ISO/IEC 9126 y ISO/IEC 14598 son:

- Mejor coordinación de la guía en la medición y evaluación de la calidad de los productos de software.
- Mejor orientación para la especificación de requisitos de calidad de los productos de software.
- Mejor distinción entre las partes beneficiarias del producto software (usuario final, organización y equipo de mantenimiento) del sistema y sus necesidades.
- Mejor integración de las definiciones de usabilidad gracias a las vistas de los modelos.

Las principales diferencias con respecto a los mismos son:

- La introducción de un modelo de referencia general.
- La introducción de guías detalladas y dedicadas a cada división.
- La introducción de elementos de medidas de calidad dentro de la división de medición de la calidad.
- Incorporación y revisión de un modelo de calidad para los datos.
- Incorporación y revisión del proceso de evaluación.
- Coordinación y armonización del contenido de ISO/IEC 15939 (2000).
- Introducción de guías para uso práctico en forma de ejemplos.
- Renombramientos de algunas sub-características para evitar ambigüedades.
- Incorporación de nuevas sub-características.

La principal desventaja de emplear este estándar reside en ser un estándar de reciente creación y es genérico para cualquier producto software.

Con respecto al modelo de calidad planteado, existen tres vistas del modelo según el contexto donde se aplique (ver Figura 2): *Modelo de Calidad de Software* que será utilizado para evaluar un producto software concreto; *Modelo de Calidad de Datos* que será utilizado para evaluar la calidad de la información que maneja el software; y el *Modelo de Calidad en Uso* que pretende evaluar cómo las partes beneficiarias del producto alcanzan sus objetivos usando el producto en un contexto específico de uso.

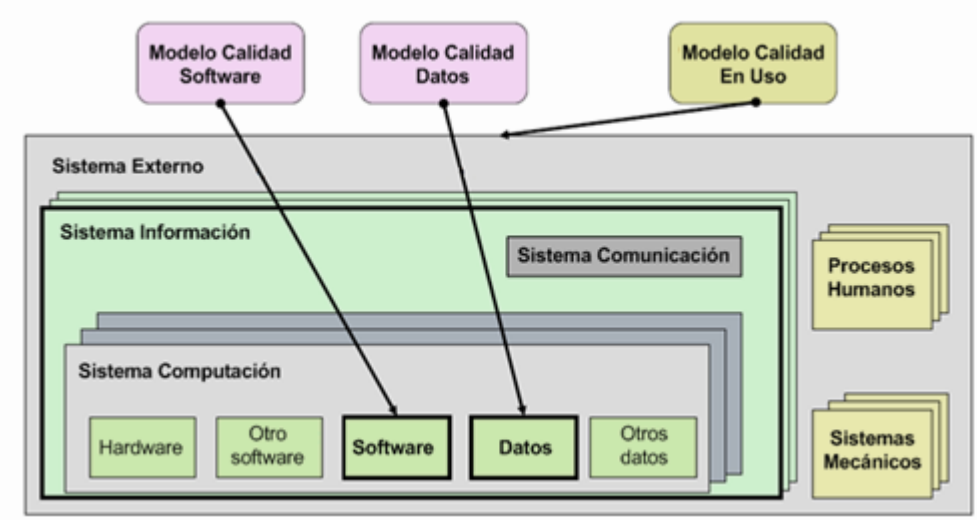

Figura 2. Vistas de los modelos de calidad según SQuaRE.

En este trabajo, nos centraremos en utilizar el modelo de calidad de software para customizarlo para la selección de plataformas cloud. El modelo de calidad definido por el estándar ISO/IEC 25010 (2011) se encuentra compuesto por ocho características que se muestran en la Figura 3.

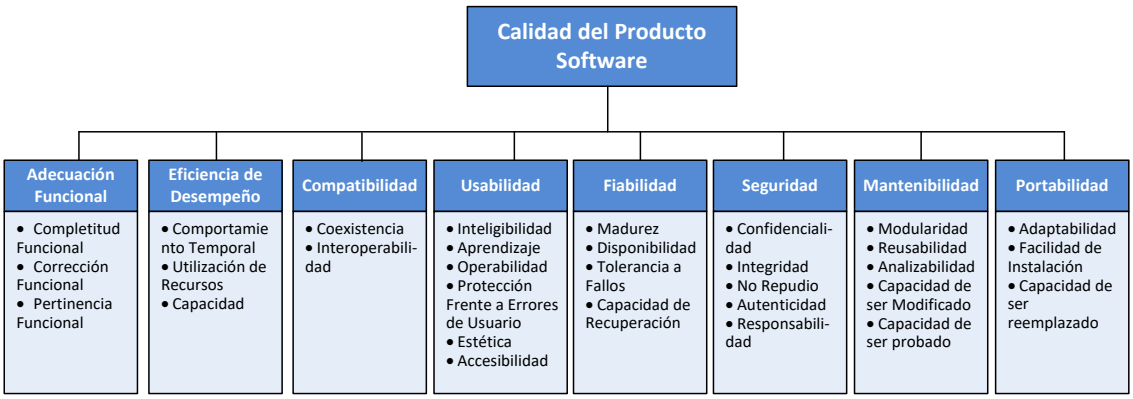

Figura 3. Modelo de calidad del producto definido por la ISO/IEC 25010 (2011).

2.6. Plataformas cloud

En esta sección vamos a introducir las principales plataformas cloud que se encuentran en la actualidad en el mercado, las cuales son Microsoft Azure, Amazon Web Services y Google Cloud Platform.

2.6.1. Microsoft Azure

Dado que la implementación tiene una estrecha relación con la plataforma en la nube en la que está siendo construida, así como con el tipo de servicios que permite monitorizar en este prototipo, es conveniente entrar en detalle en este aspecto en algunos de los aspectos de la plataforma, en especial las librerías de monitorización utilizadas y su funcionamiento.

La plataforma Microsoft Azure fue anunciada por primera vez en el evento PDC (Professional Developers Conference) en 2008 como Windows Azure Platform y lanzada en 2010 como Windows Azure. Es un servicio IaaS y PaaS ofrecido por Microsoft para construir, desplegar y gestionar aplicaciones y servicios alojados en los centros de datos de Microsoft.

Este servicio es capaz de soportar distintos lenguajes de programación (.NET, PHP, C++, Ruby, Java), herramientas y servicios, incluyendo los pertenecientes a terceros, así como un ecosistema creado por Microsoft basado en .Net.

Además del servicio de ejecución, dispone de diferentes mecanismos de almacenamiento de datos: tablas NoSQL, blobs, blobs para streaming, colas de mensajes o 'drives' NTFS para operaciones de lectura/escritura a disco.

Azure admite cualquier sistema operativo, lenguaje o herramienta, ya sea Windows, Linux, SQL Server, Oracle, C# o Java. Pone a nuestro alcance los ecosistemas de Windows y Linux, por lo que nos da la posibilidad de crear aplicaciones y servicios que funcionan con cualquier dispositivo.

Dentro de este entorno, las librerías más relevantes empleadas por este proyecto son las de Azure Diagnostics, librerías creadas para la monitorización de servicios en la nube y en particular los datos que estas emplean, los Performance Counters.

Windows ofrece una manera de proteger la información importante con una copia de seguridad automática dentro de un servicio de almacenamiento. Las copias de seguridad quedan cifradas antes de la transmisión y se almacenan cifradas en Windows Azure. Como ventajas podríamos citar las siguientes: pagas por lo que usas, facilidad y rapidez del escalado, el usuario no se preocupa de las tareas de configuración de los servicios, alta disponibilidad, gestión de recuperación de datos.

Como desventajas podríamos citar las siguientes: coste inicial elevado, plataforma con retraso de desarrollo, lentitud de la respuesta de las instancias, no dispone de estructura MapReduce.

2.6.2. Amazon Web Services

A finales de 2003 se propuso por primera vez ofrecer servicios cloud basados en la infraestructura de Amazon.com. En noviembre de 2004 se publicó Simple Queue Service, el primer servicio cloud de Amazon [14].

En 2006 fue lanzado oficialmente Amazon Web Services, ofreciendo a los usuarios una plataforma de servicios cloud con la finalidad de ayudar a las empresas a escalar y crecer.

Esta plataforma soporta los siguientes lenguajes de programación: Android, Browser, iOS, Java, .NET, Node.js, PHP, Python, Ruby, Go, C++, AWS Mobile SDK y AWS IoT Device SDK.

También ofrece herramientas que facilitan el desarrollo en los siguientes IDE: Eclipse, Visual Studio y VSTS.

Amazon ofrece el servicio Amazon CloudWatch para monitorizar las aplicaciones que están siendo ejecutadas en la plataforma. Este servicio permite realizar el seguimiento de distintas métricas, visualizar archivos de registro, configurar alarmas y reaccionar automáticamente a cambios en los recursos del cliente dentro de la plataforma [15].

La sencilla interfaz de servicios web de Amazon EC2 permite obtener y configurar su capacidad fácilmente. Proporciona un control completo sobre sus recursos informáticos y permite ejecutarse en el entorno informático acreditado de Amazon. Amazon EC2 reduce el tiempo necesario para obtener y arrancar nuevas instancias de servidor en minutos, lo que permite escalar rápidamente la capacidad.

Como ventajas podríamos citar las siguientes: el código puede ser escrito en lenguajes de programación sencillos como C#, .Net , MVC, no hay dependencia del proveedor en cuanto a temas del código y alta disponibilidad.

Como desventajas podríamos citar las siguientes: cobro del precio mínimo aunque no se utilice todos los recursos ofrecidos, el escalado es una tarea difícil, si una instancia falla puede fallar todo el sistema.

2.6.3. Google Cloud Platform

En abril de 2008 Google publicó su primer servicio cloud, App Engine, el cual era una plataforma (PaaS). El servicio inicialmente estaba limitado a 10.000 desarrolladores, pero tuvo que ser expandido hasta 75.000 a finales de mayo, quedando muchos desarrolladores en cola de espera.

En 2010, Google lanzó su segundo servicio, Cloud Storage, una infraestructura cloud (IaaS). Desde entonces, Google ha ido aumentando sus servicios y bajando los precios para sus usuarios, siendo un punto fuerte de su plataforma sus costes bajos [16].

Google Cloud Platform soporta los siguientes lenguajes de programación: Go, Java, .NET, Node.js, PHP, Python, Ruby. Google App Engine ofrece varias opciones

de almacenamiento: una base de datos tradicional MySQL usando la nube SQL, una base de datos NoSQL o almacenamiento de objetos haciendo uso de Cloud Storage.

Esta plataforma también ofrece herramientas para facilitar el desarrollo en IDEs como Android Studio, IntelliJ, Visual Studio o Eclipse. La App Engine SDK permite probar aplicaciones de forma local en un entorno simulado y luego implementar una aplicación con las herramientas de línea de comandos simples o el lanzador en el escritorio.

En cuanto a la monitorización del servicio Google ofrece Stackdriver Monitoring. Este servicio recoge métricas, eventos y metadatos de Google Cloud Platform, permitiendo también configurar alarmas, gráficas y la visualización de registros [17].

Como ventajas podríamos citar las siguientes: la plataforma se encarga del manejo de la carga del tráfico irregular, pagas los servicios que usas, todos los servicios están listo para usarse y el usuario no necesita configurar nada, servicios gratuitos hasta una determinada cuota, posee una red de alta velocidad, posibilidad de usar gran variedad de lenguajes y alta disponibilidad.

Como desventajas podríamos citar las siguientes: no soporta aplicaciones multihilo, el coste es elevado para carga constante, dificultad de cambiar de Google App Engine a otra plataforma. No es posible acceder al sistema operativo, ni ejecutar software de terceros, ni es posible realizar tareas de administración desde línea de comandos, ni servicio SSH.

2.6.4. Otras Plataformas

Existen en el mercado otras plataformas cloud menos populares. Entre ellas encontramos Red Hat Openshift, la plataforma ofrecida por Red Hat. La principal característica de esta plataforma es que ofrece un ecosistema de código abierto para proporcionar servicios de la plataforma como pueden ser Appcelerator, MongoDB, etc.

IBM nos ofrece la plataforma IBM SmartCloud. Esta plataforma ha sido diseñada para garantizar un alto rendimiento y una alta disponibilidad. Existe la posibilidad de definir acuerdos de nivel de servicio (SLA) personalizados, para sincronizar en la mayor medida posible las necesidades de negocio y los requisitos de uso.

Por último, hablaremos de Openstack, un conjunto de proyectos software de código abierto desarrollados principalmente por RackSpace y la NASA. OpenStack proporciona herramientas que pueden ser usadas para configurar y ejecutar los servicios cloud computing. También, OpenStack tiene sus servicios disponibles mediante una API compatible con los servicios cloud de Amazon.

2.7. Cloud Service Measure Index (SMI)

Existe una gran variedad de servicios cloud de los cuales los clientes pueden hacer uso, siendo muy importante elegir el servicio "adecuado". Los proveedores pueden satisfacer esos requerimientos. A menudo se deberán poner en una balanza los requisitos funcionales y no funcionales para evaluar a los distintos servicios como proveedores. De ahí no solamente es importante descubrir los servicios que existen sino además evaluarlos.

En este contexto, el *Cloud Service Measure Index Consortium* (2015) liderado por la Universidad Carnegie Mellon, ha propuesto indicadores y métricas que son combinados en forma de un Índice de Medición de Servicios (*Service Measure Index*, SMI), ofreciendo evaluación comparativas de proveedores de servicios cloud. Este índice pretende convertirse en un estándar para cuantificar y evaluar los servicios de cloud computing y aborda la necesidad de medidas aceptadas por toda la industria para calcular los beneficios y riesgos de los servicios de cloud computing.

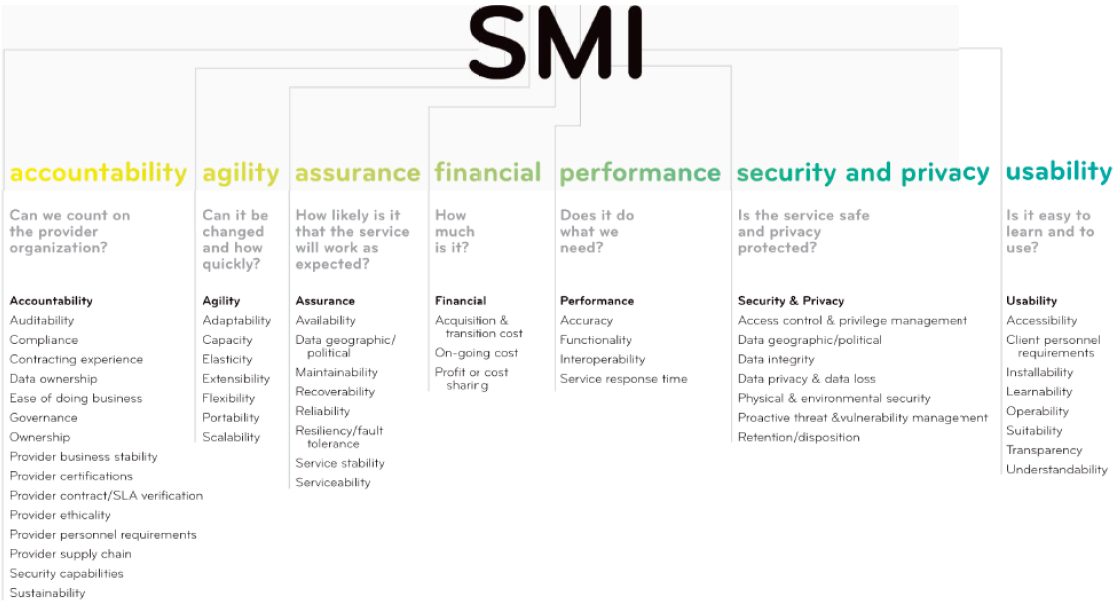

Figura 4. Índice de Medición de Servicios (SMI).

Esos índices de medición pueden ser usados por clientes para comparar los servicios ofrecidos. Por tanto, en su trabajo estos autores proponen un marco de trabajo (SMICloud) que puede comparar diferentes proveedores cloud basándose en sus requisitos. Este marco de trabajo permite comparar los servicios desde sus prioridades y algunas dimensiones.

Los atributos SMI son diseñados en base los estándares ISO/IEC 9126, ISO/IEC25000 e ISO/IEC 20000 y teniendo en consideración conceptos introducidos por ISACA (Csmic, 2014), por el consorcio CSMIC (2015). Estos consisten en un conjunto de indicadores claves de desempeño (KPIs) que puede proveer un método estandarizado de medición y comparación de servicios de negocios. Provee además

una vista de calidad de servicio necesaria para que los clientes puedan seleccionar un servicio cloud basándose en su manera de facturación, agilidad, aseguramiento del servicio, coste, rendimiento, seguridad, privacidad y usabilidad. La figura 4 describe los indicadores de más alto nivel y su descomposición en atributos.

En definitiva, el objetivo principal fue el proporcionar un modelo de calidad y un conjunto de métricas para servicios cloud a nivel de IaaS. Sin embargo, el modelo está orientado a la selección de proveedores – no de plataformas. Además, es un modelo teórico ya que las métricas propuestas son únicamente para medir la usabilidad, y se necesita definir nuevas métricas para medir los otros indicadores del SMI.

2.8. Conclusiones

En este capítulo se ha introducido cloud computing y discutido sus características más importantes como el pago por uso, acceso sin restricciones, agilidad en la escalabilidad, reparto de recursos y la abstracción. Estas características hacen que la computación en la nube sea una opción económica para los clientes, ya que ofrece una gran flexibilidad a la hora de contratar recursos, pudiendo prescindir de ellos cuando no sean necesario, pagando de esta manera únicamente por los recursos utilizados.

Hemos presentado también los tres modelos de servicios existentes en la actualidad: Infraestructura como servicio (IaaS), Plataforma como Servicio (PaaS) y Software como Servicio (SaaS), así como los distintos modelos de despliegue: *Public Cloud* (Nube Pública), *Private Cloud* (Nube Privada), *Community Cloud* (Nube comunitaria) y *Hybrid Cloud* (Nube Híbrida).

Hemos discutido las plataformas de cloud computing más populares actualmente. Siendo las plataformas de Google, Microsoft Azure y Amazon las que tienen más cuota de mercado actualmente.

Y, por último, hemos introducido el Índice de Medición de Servicios (SMI) para medir la calidad de servicios y proveedores de cloud computing. En la próxima sección plantearemos un modelo de calidad específico para la selección y complaración de plataformas cloud.

3. Modelo de Calidad para la Selección de Plataformas Cloud

En este capítulo se va a hablar de la calidad de las plataformas de cloud computing. Para ello empezaremos introduciendo los modelos de calidad, presentado seguidamente uno de los principales estándares, el estándar ISO/IEC 25000 (SQuARE). Por último, realizaremos una extensión de la ISO/IEC 25010 la cual utilizaremos para el desarrollo de nuestro propio modelo de calidad.

3.1. Modelos de Calidad

La selección de características y su descomposición en subcaracterísticas y atributos de calidad se ha realizado considerando algunos trabajos previos. En particular, se ha utilizado los resultados de la revisión sistemática sobre atributos y métricas para servicios cloud realizada por Navas (2016) seleccionando los atributos y métricas a nivel PaaS y el estudio sobre análisis y evaluación de marcos de trabajo para el desarrollo de aplicaciones y servicios en la nube PAAS realizado por Domínguez et al. (2014).

Los modelos de calidad del software vienen a ayudar en la puesta en práctica del concepto general de calidad que vimos en el apartado anterior, ofreciendo una definición más operacional. Unos de los modelos de calidad más antiguos y extendidos es el de McCall (McCall, 1977), y de él han derivado otros modelos, como el de Boehm (Boehm, 78) o el SQM (Murine, 1988).

En los modelos de calidad, la calidad se define de forma jerárquica. Es un concepto que se deriva de un conjunto de sub-conceptos, cada uno los cuales se va a evaluar a través de un conjunto de *indicadores* o *métricas*. Tienen una estructura, por lo general, en tres niveles (Figura 5):

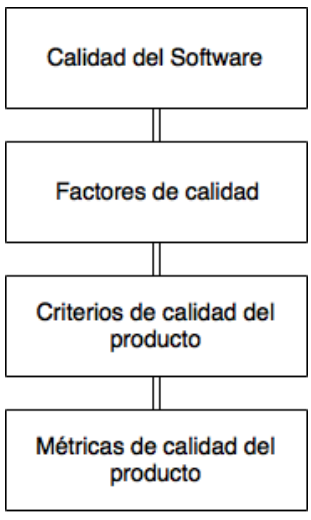

Figura 5. Estructura de un Modelo de Calidad.

- En el nivel más alto de la jerarquía se encuentran los **factores** de calidad, que representan la calidad desde el punto de vista del usuario. Son las características que componen la calidad. También se les llama **características o atributos de calidad externos**.

- Cada uno de los factores se descompone en un conjunto de **criterios** de calidad. Son atributos que, cuando están presentes, contribuyen al aspecto de la calidad que el factor asociado representa. Se trata de una visión de la calidad desde el punto de vista del producto software. También se les llama **características o atributos de calidad internos**.

- Para cada uno de los criterios de calidad se definen entonces un conjunto de **métricas**, que son medidas cuantitativas de ciertas características del producto que, cuando están presentes, dan una indicación del grado en que dicho producto satisface un determinado atributo de calidad.

La ventaja de los modelos de calidad es que la calidad se convierte en algo concreto, que se puede *definir*, que se puede *medir* y, sobre todo, que se puede *planificar*. Los modelos de calidad ayudan también a comprender las relaciones que existen entre diferentes características de un producto software.

3.2. Extensión de la ISO/IEC 25010 para la Selección de Plataformas Cloud

Un modelo de calidad representa la piedra angular en torno a la cual se pueden clasificar las características de calidad que se van a tener en cuenta a la hora de seleccionar las plataformas cloud. Para la definición de este modelo utilizaremos el estándar ISO/IEC 25010 (2011) que plantea la descomposición de la calidad en características y sub-características. Sin embargo, dependiendo de la plataforma que se quiera evaluar, hará falta definir atributos y métricas específicos, que permitan realizar las mediciones. Para determinar estos atributos y métricas específicos utilizaremos algunos estudios previos [3] y un modelo de calidad para servicios cloud [2].

A continuación, describimos los roles involucrados en el uso del modelo de calidad para la selección de plataformas cloud planteado en este trabajo, así como la descomposición jerárquica del modelo en características de calidad de las plataformas cloud (PaaS), subcaracterísticas, atributos de calidad y métricas.

3.2.1. Roles

Consideramos que los siguientes roles planteados en la ISO/IEC 17788:2014 y el NIST [ref], son relevantes a nivel de plataforma cloud:

- **Proveedor (*Cloud Service Provider*):** Es la entidad que provee la plataforma cloud. El proveedor posee y controla la plataforma. Para el proveedor es muy importante que la plataforma esté disponible y se cumpla el SLA [4].

- **Cliente (*Cloud Service Customer*):** Es el usuario que compra y consume los servicios ofrecidos por el proveedor [4] [5]. Se trata de una relación comercial con el propósito de utilizar el servicio cloud.

En esta categoría se incluyen los desarrolladores y los CEOs. Los CEOs normalmente deciden dónde los desarrolladores desarrollan y albergan aplicaciones. A diferencia del desarrollador, preocupado por el desarrollo de aplicaciones, el CEO está preocupado principalmente por la seguridad y la conformidad, además de por la productividad del desarrollador.

En un estudio reciente realizado por Forrester[2] se menciona que las plataformas cloud toman formas diversas, incluyendo las que ofrecen servicio básico de infraestructura como servicio, hasta las que ofrecen herramientas y servicios completos o parciales de plataforma. Cada uno de estos tipos de plataforma se adapta mejor a un tipo diferente de desarrollador de aplicaciones". No todos los desarrolladores crean aplicaciones de la misma manera. Algunos quieren escribir código rápidamente, lanzarlo rápidamente y no tener que lidiar con la provisión de la infraestructura requerida para correr sus aplicaciones. Otros desarrolladores quieren involucrarse con la infraestructura necesaria en el lanzamiento de nuevos programas, para que se ajuste a las necesidades de las aplicaciones. Los autores del informe de Forrester identifican tres tipos de desarrolladores:

- **Desarrolladores Rápidos**: Estos desarrolladores valoran la facilidad de uso, prefieren las interfaces gráficas de desarrollo, desarrollan aplicaciones en días o semanas y no quieren perder tiempo configurando la infraestructura.

- **Programadores:** Estos desarrolladores aman la programación, pero no quieren gestionar la infraestructura necesaria para escribir aplicaciones. Desean una plataforma de desarrollo de aplicaciones cloud que configure los recursos que necesitan, pero que les dé la oportunidad de jugar con algunos de los componentes de la infraestructura para obtener las mejores prestaciones de la aplicación.

- **Desarrolladores:** A estos desarrolladores les gusta meterse con la infraestructura que soporta las aplicaciones que escriben. Quieren gestionar las bases de datos, el almacenamiento y las máquinas virtuales en las que descansan sus aplicaciones.

- **Bróker:** Es la entidad que ofrece servicios de intermediación entre el consumidor y el proveedor. Los tres principales servicios que ofrecen son servicios de agregación (integración en diferentes proveedores), servicios de intermediación (ayudan a seleccionar el proveedor más apropiado), servicios de "arbitraje" (obtener buenas ofertas en lo que se paga por los servicios). [4]

El objetivo de establecer los roles es poder guiar el proceso de selección de plataformas cloud en función de un punto de vista de interés.

[2] goo.gl/GoJRf6

3.2.2. Características, Subcaracterísticas y Atributos

De las 8 características de más alto nivel hemos seleccionado adecuación funcional, eficiencia de desempeño, usabilidad, fiabilidad, seguridad y portabilidad. Dentro de estos no hemos seleccionado todas las subcaracterísticas y dentro de cada característica se encuentra explicado el razonamiento seguido para la selección de subcaracterísticas.

La característica de compatibilidad no la hemos seleccionado debido a la escasez de métricas para evaluar la característica y que las métricas que existen no son relevantes para evaluar la calidad de las plataformas cloud.

La característica de mantenibilidad tampoco la hemos seleccionado porque el mantenimiento de la plataforma no es algo de importancia para el usuario de plataformas cloud y por lo tanto no influye en la decisión a la hora de elegir plataforma cloud.

Las razones por la que hemos elegido el resto de características se encuentran más abajo.

A continuación, se describen las características seleccionadas y su descomposición en subcaracterísticas y atributos de calidad específicos para las plataformas cloud.

Adecuación funcional:
Representa la capacidad de la plataforma cloud para proporcionar funciones que satisfacen las necesidades declaradas e implícitas, cuando la plataforma se usa en unas condiciones especificadas [1].

Es una característica muy importante, ya que como en la propia descripción se indica, es el grado en el que se ofrecen funciones que satisfagan las necesidades del cliente, algo fundamental en este tipo de servicios. Esta característica de calidad se descompone en las siguientes subcaracterísticas/atributos:

- **Completitud funcional:** Grado en el cual el conjunto de funcionalidades de la plataforma cubre todas las tareas y los objetivos del usuario especificados [1].

- **Corrección funcional:** Capacidad de la plataforma cloud para proveer resultados correctos con el nivel de precisión requerido [1].

- **Pertinencia funcional:** Capacidad de la plataforma cloud para proporcionar un conjunto apropiado de funciones para tareas y objetivos de usuario especificados [1].

La tabla 1 describe la descomposición de la característica adecuación funcional en atributos y métricas.

Tabla 1. Características, atributos y métricas de adecuación funcional

Característica	Atributo	Significado	Métrica	Ref.	Rol	ID
Completitud funcional	Idoneidad de la plataforma cloud	Es el grado en que se cumplen los requisitos de un cliente por un proveedor de la plataforma cloud.	Idoneidad de características no esenciales	[23]	Proveedor, cliente	**M1**
	Completitud	Representa el grado en que un servicio es compatible con todas las tareas y los objetivos especificados por el cliente.	Completitud de conjunto variante	[24]	Proveedor, Cliente	**M2**
Corrección funcional	Capacidad computacional de la plataforma cloud	Se define como el grado en el que la plataforma se ejecuta en el tiempo esperado.	Capacidad computacional de la plataforma	[25]	Proveedor, cliente	**M3**
Pertinencia funcional	Consistencia de la plataforma cloud	Se puede definir como cualquier desviación de los valores prometidos.	Consistencia durante un período de tiempo	[26]	Proveedor, cliente	**M4**

Eficiencia de desempeño: Esta característica representa el desempeño relativo a la cantidad de recursos utilizados por la plataforma cloud bajo determinadas condiciones [1]. El cliente paga por unos recursos, por lo que a mayor cantidad de recursos utilizados mayor es el precio del servicio. Al cliente le interesa que la plataforma ofrezca un buen desempeño para obtener un mayor provecho del servicio.

Esta característica de calidad se descompone en las siguientes subcaracterísticas:

- **Tiempo de respuesta del servicio:** Un indicador del tiempo entre cuando se solicita un servicio, y cuando la respuesta está disponible [3].

- **Utilización de recursos:** Las cantidades y tipos de recursos utilizados cuando la plataforma lleva a cabo sus funciones bajo unas condiciones determinadas [1].

- **Capacidad:** Grado en que los límites máximos de un parámetro de una plataforma cloud cumplen con los requisitos [1].

En entornos cloud la latencia es un atributo relevante ya que distintos ordenadores pueden estar muy lejos geográficamente uno de los otros.

La tabla 2 describe la descomposición de la característica eficiencia de desempeño en atributos y métricas.

Tabla 2. Características, atributos y métricas de eficiencia de desempeño

Característica	Atributo	Significado	Métrica	Ref.	Rol	ID
Tiempo de respuesta del servicio	Capacidad latente del servicio	Representa la cantidad de capacidad computacional disponible sin necesidad de encender nuevos *hosts*.	Promedio de capacidad de la CPU libre de todos los *hosts* que se ejecutan	[27]	Proveedor, Bróker	**M5**
	Eficiencia de la respuesta	Representa la eficiencia de respuesta sobre todos los trabajos presentados durante un período de tiempo.	Eficiencia de respuesta	[28]	Cliente, Proveedor	**M6**
	Latencia del servicio	Representa el tiempo necesario para que la solicitud de un usuario pueda ser manejada por el servicio competente.	Latencia	[25], [27]	Cliente	**M7**
			Retraso de las reacciones de ida y vuelta	[29]	Cliente	**M8**
	Tiempo de ejecución del artefacto cloud	Representa el tiempo que tarda en ejecutarse una tarea.	Tiempo de iteración del bucle principal	[30]	Cliente, Proveedor	**M9**
			Tiempo dedicado al motor de físicas	[30]	Cliente	**M10**
			Tiempo dedicado al motor de renderizado	[30]	Cliente	**M11**
	Tiempo de espera del servicio	Representa el tiempo que se toma en procesar una solicitud.	Tiempo de espera	[31], [32]	Cliente, Proveedor	**M12**
	Tiempo de respuesta del servicio	Representa la diferencia entre el tiempo de la solicitud de servicio y el tiempo cuando el servicio está disponible.	Tiempo promedio del tiempo de respuesta	[23], [27], [33]	Proveedor, Cliente	**M13**
			Tiempo de respuesta máximo	[23], [27]	Proveedor, Cliente	**M14**
			Tiempo de respuesta	[3], [27], [34],	Proveedor, Cliente	**M15**

				[35], [36], [37], [38], [39]		
	Tiempo de retraso del servicio	Representa el retraso esperado entre el momento en que se envía una solicitud al servicio y al momento en que se reciben los resultados del servicio.	Tiempo de retraso	[40], [44], [49], [51], [52]	Proveedor, Cliente	**M16**
	Violación del Acuerdo de Nivel de Servicio (SLA) del servicio	Representa la evaluación de las violaciones del Acuerdo de Nivel de Servicio para garantizar la calidad del servicio para los clientes.	Violación del SLA	[45]	Proveedor, Cliente	**M17**
Utilización de recursos	Ancho de banda del servicio	Representa los datos y recursos de comunicación disponibles o consumidos.	Ancho de banda de la red	[25]	Cliente	**M18**
			Disponibilidad del ancho de banda	[41]	Proveedor	**M19**
	Coste de la energía del servicio	Representa el consumo de energía por cada *host*.	Coste de energía	[27]	Bróker	**M20**
	Coste en curso del servicio	Representa el coste a nivel computacional del servicio.	Almacenamiento de datos	[23]	Bróker	**M21**
			Uso computacional	[23]	Bróker	**M22**
			Coste de almacenamiento	[42]	Bróker, Cliente	**M23**
	Eficiencia energética del servicio	Se refiere a la utilización de menos energía para producir la misma cantidad de servicios.	Potencia consumida por la CPU	[53]	Bróker	**M24**
			Consumo total de energía	[53]	Bróker	**M25**
	Eficiencia del servicio	Es una medida de lo bien que un servicio utiliza los recursos.	Eficiencia de un proveedor del servicio	[34]	Bróker	**M26**
			Utilización de recursos	[46]	Bróker	**M27**
	Rendimiento (*Throughput*) del servicio	Representa la cantidad del trabajo completado contra el tiempo consumido.	Rendimiento de un proveedor de servicio	[34]	Bróker	**M28**

	Tasa de utilización del servicio	Representa el grado de uso del servicio.	Tasa de utilización del servicio	[37]	Proveedor	**M29**
	Utilización de recursos del servicio	Se refiere a la fracción de tamaño de recurso que está asignado para servir las peticiones de los usuarios.	Utilización del recurso	[36]	Proveedor, Bróker	**M30**
Capacidad	Capacidad del proveedor del servicio	Significa la máxima cantidad de recursos que un proveedor de servicios puede proporcionar en las horas punta.	Capacidad del CPU	[34]	Cliente	**M31**
			Capacidad de la memoria	[34]	Cliente	**M32**
			Capacidad de almacenamiento	[34]	Cliente	**M33**
			Capacidad de la red	[34]	Cliente	**M34**
	Desempeño de la CPU del servicio	Representa la calidad en tiempo real de aprovisionamiento de la CPU en entornos virtualizados.	Tiempo de no disponibilidad real de la CPU	[54]	Proveedor, Cliente	**M35**
			Número de núcleos de la CPU	[54]	Cliente	**M36**
			Coeficiente de rendimiento de la CPU	[54]	Cliente	**M37**
	Desempeño de la transmisión de datos del servicio	Se refiere a la calidad de servicio para la transferencia de datos de un servicio a otro.	Velocidad de transmisión de datos alcanzada	[49]	Cliente	**M38**
	Desempeño de la máquina virtual	Proporciona una visión específica del rendimiento de la máquina virtual.	Rendimiento de la CPU	[55]	Cliente	**M39**
			Rendimiento del disco (I/O)	[55]	Cliente	**M40**
			Rendimiento de la memoria	[55]	Cliente	**M41**
			Media del tiempo de respuesta	[55]	Cliente	**M42**
	Rechazo de solicitud del servicio	Representa la probabilidad de rechazo de una solicitud al servicio.	Probabilidad de rechazo	[58]	Proveedor, Cliente	**M43**

Usabilidad: Capacidad de la plataforma cloud para ser entendida, aprendida, usada y resultar atractiva para el usuario, cuando se usa bajo determinadas condiciones [1]. Es una característica que depende del usuario el grado de importancia que le dé. Si es un usuario con conocimientos muy avanzados quizás no le de mucha importancia a esta característica, pero si el usuario que va a utilizar la plataforma por primera vez y no tiene tantos conocimientos o no puede emplear mucho tiempo en adquirir los conocimientos necesarios para ello, le dará mucha más importancia.

La usabilidad es esencial en entornos cloud. El acceso a los servicios en la nube, a datos, facilita la vida de personas con algún tipo de limitación para los desplazamientos. También permite la integración laboral, al acceder a través de ella a una diversidad de plataformas adaptadas a distintas capacidades.

Esta característica de calidad se descompone en las siguientes subcaracterísticas:

- **Capacidad de aprendizaje:** Capacidad de la plataforma que permite al usuario aprender su aplicación [1].

- **Capacidad para ser usada:** Capacidad de la plataforma que permite al usuario operarla y controlarla con facilidad [1].

- **Protección contra errores de usuario:** Capacidad del sistema para proteger a los usuarios de hacer errores [1].

No hemos seleccionado las subcaracterísticas restantes de usabilidad de la ISO 25010 debido a la falta de métricas existentes para medir esta característica.

La tabla 3 describe la descomposición de la característica usabilidad en atributos y métricas.

Tabla 3. Características, atributos y métricas de usabilidad

Característica	Atributo	Significado	Métrica	Ref.	Rol	ID
Capacidad de aprendizaje	Facilidad de aprendizaje	Representa el esfuerzo del usuario para aprender el servicio.	Tiempo medio experimentado por los usuarios anteriores para aprender	[23], [25]	Cliente	**M44**
Capacidad para ser usada	Facilidad de uso	Representa la facilidad con que un servicio puede ser usado.	Facilidad de uso	[56], [57]	Cliente	**M45**
Protección contra errores de usuario	Tasa de éxito	Representa el grado de éxito del usuario al realizar las operaciones disponibles por el servicio.	Tasa de éxito	[25]	Cliente	**M46**

Fiabilidad: Capacidad de la plataforma cloud para desempeñar las funciones especificadas, cuando se usa bajo unas condiciones y periodo de tiempo determinados [1]. El cliente querrá tener una certeza de que la plataforma funcionará correctamente cuando se necesite, por lo tanto, es una característica relevante.

Esta característica de calidad se descompone en las siguientes subcaracterísticas:

- **Madurez:** capacidad de la plataforma para satisfacer las necesidades de fiabilidad en condiciones normales [1].

- **Disponibilidad:** la adecuación de la ventana de la disponibilidad del servicio, así como la probabilidad de que en realidad será siempre la ventana de disponibilidad a los clientes [3].

- **Tolerancia a fallos:** capacidad de la plataforma cloud para operar según lo previsto en presencia de fallos hardware o software [1].

- **Capacidad de recuperación:** capacidad de la plataforma cloud para recuperar los datos directamente afectados y restablecer el estado deseado de la plataforma en caso de interrupción o fallo [1].

La tabla 4 describe la descomposición de la característica fiabilidad en atributos y métricas.

Tabla 4. Características, atributos y métricas de fiabilidad

Característica	Atributo	Significado	Métrica	Ref.	Rol	ID
Madurez	Confianza (*Dependability*)	Representa el grado en el que el servicio cumple bien sus objetivos bajo condiciones externas no controladas.	Confianza	[59]	Proveedor, Cliente	**M47**
Disponibilidad	Disponibilidad	Representa el grado en que un servicio está disponible y accesible cuando se requiere para su uso.	Tiempo fuera de línea	[43]	Proveedor, Cliente	**M48**
			Tiempo medio entre fallos	[25]	Proveedor, Cliente	**M49**
			Tiempo medio hasta el fallo	[25]	Proveedor, Cliente	**M50**
			Tiempo medio para recuperar	[37]	Proveedor, Cliente	**M51**
			Porcentaje del tiempo en línea	[50]	Proveedor, Cliente	**M52**
			Puntuación sobre la disponibilidad dada por el usuario	[63] [64]	Proveedor, Cliente	**M53**
			Disponibilidad	[3]	Proveedor, Cliente	**M54**
	Tasa de disponibilidad	Representa la tasa de disponibilidad.	Tasa de disponibilidad	[42]	Proveedor, Cliente	**M55**
	Número de máquinas físicas	Representa el número de máquinas físicas que tiene el proveedor.	Número medio de máquinas físicas en modo activo	[48]	Proveedor	**M56**
	Utilidad	Representa el grado en que el servicio ha estado en línea.	Utilidad	[25]	Proveedor, Cliente	**M57**
Tolerancia a fallos	Resistencia	Representa la capacidad del servicio para reaccionar ante una situación de sobrecarga.	Tiempo de degradación	[31]	Proveedor, Cliente	**M58**
			Tiempo de recuperación	[31]	Proveedor, Cliente	**M59**

			Pérdida de rendimiento máximo	[31]	Cliente	**M60**
			Tiempo de asentamiento	[62]	Proveedor, Cliente	**M61**
			Tiempo máximo	[62]	Proveedor, Cliente	**M62**
			Tiempo de percentil	[62]	Proveedor, Cliente	**M63**
	Constancia	Representa el grado en el que el servicio es tolerante a fallos, y que a la vez se recupera de ellos.	Tolerancia a fallos	[24]	Proveedor, Cliente	**M64**
			Recuperación de fallas	[24]	Proveedor, Cliente	**M65**
Capacidad de recuperación	Continuidad	Representa el grado en que un servicio se proporciona en todas las circunstancias, incluyendo la mitigación de los riesgos a un nivel razonable y la recuperación del negocio después de la suspensión.	Tasa de reparación de accidentes	[50]	Proveedor, Cliente	**M66**
			Prontitud de la preparación de emergencias para las empresas claves	[50]	Proveedor, Cliente	**M67**
			Cobertura de recuperación de fallas	[46]	Proveedor, Cliente	**M68**

Seguridad: Capacidad de protección de la información y los datos de manera que personas o sistemas no autorizados no puedan leerlos o modificarlos [1]. Esta característica se encarga de determinar si la plataforma cumple con los requisitos de seguridad y conformidad requeridos.

La creación de una infraestructura de cloud computing moverá los datos confidenciales fuera del entorno empresarial y hacia la nube, lo que significa que los usuarios necesitan mejorar la seguridad y actualizar sus arquitecturas para abordar posibles problemas de gestión de riesgos y entrega de aplicaciones.

La importancia de esta característica dependerá de las necesidades del cliente. Si el cliente tiene información confidencial seguramente le será de mucha importancia la confidencialidad y la integridad, sin embargo, si la información que se guarda no es muy importante, es bastante probable que le dé poca importancia a la confidencialidad y mucha a la integridad.

Además, para acceder con seguridad a los servicios cloud es fundamental la autentificación por diferentes vías para tener en cuenta a todos los clientes. También es importante evaluar si la plataforma proporciona mecanismos que permiten a los usuarios gestionar sus permisos de acceso a la red.

Esta característica de calidad se descompone en las siguientes subcaracterísticas:

- **Confidencialidad:** Capacidad de protección contra el acceso de datos e información no autorizados, ya sea accidental o deliberadamente [1].

- **Integridad:** Capacidad del sistema o componente para prevenir accesos o modificaciones no autorizados a datos o programas de ordenador [1].

No hemos seleccionado las subcaracterísticas restantes de seguridad de la ISO 25010 debido a la falta de métricas existentes relacionadas con estas.

La tabla 5 describe la descomposición de la característica seguridad en atributos y métricas.

Tabla 5. Características, atributos y métricas de seguridad

Característica	Atributo	Significado	Métrica	Ref.	Rol	ID
Confidencialidad	Índice de seguridad	Describe el nivel de seguridad logrado por un proveedor de servicio evaluado.	Valor del índice de seguridad	[47]	Cliente	**M69**
	Protección	Representa el grado en que un servicio protege tanto a los activos del cliente y el acceso a su información y datos.	Número de alarmas falsas monitoreadas por la seguridad corporativa	[25]	Cliente	**M70**
			Número de respuestas de servicio ineficaces a los problemas identificados por la Seguridad como debilidades de control	[25]	Cliente	**M71**
			Número de riesgos de seguridad proactiva identificados	[25]	Cliente	**M72**
			Porcentaje de recursos de datos peligrosos que residen en los sistemas	[25]	Cliente	**M73**
			Costo de la seguridad	[25]	Bróker, Cliente	**M74**
			Puntuación sobre la confidencialidad dada por el usuario	[64]	Cliente	**M75**
Integridad	Integridad de los datos	Representa la habilidad de prevención no autorizado para acceder, o modificar la información o datos del cliente.	Integridad de los datos	[28]	Cliente	**M76**
			Integridad de los datos del recurso	[61]	Cliente	**M77**
	Gestión de datos	Representa la capacidad de garantizar la integridad de los datos y gestionar las cuentas y archivos de los usuarios.	Puntuación sobre la gestión de datos dada por el usuario	[64]	Cliente	**M78**

Portabilidad:

La portabilidad representa la capacidad del servicio de ser transferido de forma efectiva y eficiente de un entorno hardware, software, operacional o de utilización a otro [2].

Esto es algo muy importante, ya que, si las necesidades del cliente aumentan, será necesario emplear más recursos hardware o software. Y para ello es necesario que el grado en el que lo que se está ejecutando sobre la plataforma se pueda portar a otros entornos debe ser lo más alto posible para poder cubrir esa demanda del cliente lo mejor posible.

Los clientes deben tener en cuenta que pueden tener que cambiar los proveedores de servicios debido a varios motivos (en aras de un aumento inaceptable de los costes en el tiempo de renovación del contrato, el cese de operaciones de negocios por parte de los proveedores de servicios, el cierre parcial del servicio en la nube sin planes de migración, la disminución inaceptable de la calidad del servicio y la disputa comercial entre el cliente y el proveedor de la nube etc.). Por lo tanto, la portabilidad y la interoperabilidad deben considerarse como parte de la gestión de riesgos a la hora de seleccionar una plataforma en la nube.

Esta característica de calidad se descompone en las siguientes subcaracterísticas:

- **Adaptabilidad:** la capacidad del proveedor de servicios para adaptarse a los cambios en las necesidades del cliente [3].

- **Elasticidad:** la capacidad de un proveedor de servicio en la nube, para ajustar el consumo de recursos para un servicio a una velocidad suficientemente rápida para satisfacer la demanda del cliente [3].

- **Escalabilidad:** la capacidad de un proveedor de servicios cloud para aumentar o disminuir la cantidad de servicios disponibles [3].

La tabla 6 describe la descomposición de la característica portabilidad en atributos y métricas.

Tabla 6. Características, atributos y métricas de portabilidad

Característica	Atributo	Significado	Métrica	Ref.	Rol	ID
Adaptabilidad	Adaptabilidad	Representa la capacidad del servicio de adaptarse a los cambios que requiere el usuario.	Adaptabilidad	[3], [23], [24]	Cliente	M79
	Completitud de variabilidad	Representa el grado en que un servicio es compatible con todas las tareas y los objetivos especificados por el cliente.	Completitud de conjunto variante	[24]	Proveedor, Cliente	M80
	Cobertura de variabilidad	Se mide teniendo en cuenta el número de puntos de variación que pueden ser personalizados por los usuarios del servicio.	Cobertura de la variabilidad	[24]	Cliente	M81
	Portabilidad	Representa la cantidad de marcos de trabajo (m) que soportan aplicaciones desarrolladas en otros marcos	Portabilidad	[3]	Cliente	M82
Elasticidad		Representa la habilidad de que un servicio suministre recursos de acuerdo a las necesidades del cliente.	Bajo-aprovisionamiento	[60]	Cliente	M83
			Sobre-aprovisionamiento	[60]	Cliente	M84
			Reactividad de Asignación	[60]	Cliente	M85
			Reactividad de liberación	[60]	Cliente	M86
			Elasticidad	[3],	Cliente	M87

			[34], [56], [57]		
		Tiempo medio necesario para expandir o contraer la capacidad	[23]	Cliente	M88
		Capacidad máxima	[23]	Cliente	M89
		Reactividad	[27]	Cliente	M90
Escalabilidad	Representa la habilidad del servicio de incrementar o reducir la cantidad de recursos disponible.	Escalabilidad horizontal de cloud	[23]	Cliente	M91
		Escalabilidad vertical de cloud	[23]	Cliente	M92
		Escalabilidad	[3], [27], [37]	Cliente	M93
		Puntuación sobre la escalabilidad dada por el usuario	[63] [64]	Cliente	M94

3.2.3. Métricas

Una métrica es *"una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado* "(IEEE, 1993.). Una métrica mide las características o atributos de una determinada entidad, que en nuestro caso son las plataformas cloud.

En este trabajo, cada métrica se describe de acuerdo a los siguientes criterios:

- **ID:** identificador de la métrica que debe ser único. Se corresponde con el código de la métrica que se muestra en la última columna de las tablas 1 a 6.
- **Nombre:** nombre de la métrica
- **Finalidad:** finalidad de la métrica en correspondencia con el atributo de calidad al que mide.
- **Fórmula:** función de cálculo de la métrica (si es indirecta) o descripción de la métrica (si es directa).
- **Interpretación:** Interpretación del valor obtenido. Por ejemplo, las métricas de la familia de estándares ISO/IEC 250000 asumen un valor entre zero y uno y hay que indicar si valores próximos a uno son mejores o peores. También se pueden establecer distintos rangos de valores para indicar mayor o menor grado de aceptación.
- **Tipo de escala:** una escala es un conjunto ordenado de valores, continuos o discretos, o conjunto de categorías. Hay cinco tipos de escala:
    - **Nominal:** clasifica las entidades medidas en categorías que no implican un orden.
    - **Ordinal:** clasifica las entidades medidas en categorías que implican un orden.
    - **De intervalo:** la diferencia entre dos valores consecutivos de la escala es igual.
    - **Ratio:** comienza con el valor 0 correspondiente a la ausencia del atributo medido, crece a intervalos iguales (unidades) y distancias iguales corresponden a cantidades iguales del atributo.
    - **Absoluta:** la medición se realiza mediante el recuento del número de elementos en una unidad.
- **Audiencia destino:** tipo de usuario(s) a los que va destinado la métrica. En este trabajo, pueden ser de tres tipos: proveedor del servicio, consumidor del servicio o bróker.

La medición es el proceso por el cual se asignan números o símbolos a atributos de entidades del mundo real. La figura 6 muestra la correspondencia entre el mundo real y el mundo formal donde se representan los valores de las métricas, sus unidades y tipo de escala.

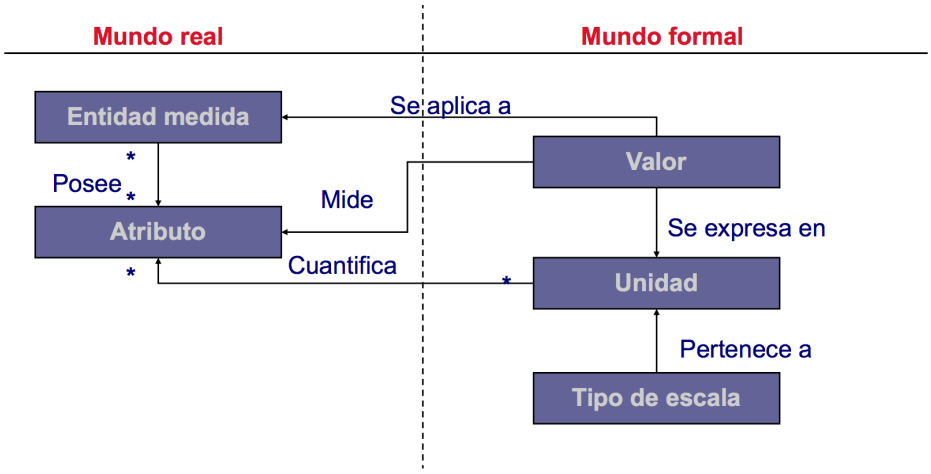

Figura 6. Medición del software ([Kitchenham et al., 1995).

A continuación, se presenta el catálogo de métricas que soporta el modelo de calidad planteado.

Métricas de Adecuación Funcional

ID	M1
Nombre	Idoneidad de características no esenciales
Finalidad	Idoneidad de la plataforma cloud
Fórmula	Nº de características no esenciales previstas por la plataforma / Nº de características no esenciales requeridas por el cliente
Interpretación	1 si todas las características son satisfechas 0 en cualquier otro caso
Tipo de escala	Nominal
Audiencia destino	Proveedor, cliente

ID	M2
Nombre	Completitud de conjunto variante
Finalidad	Completitud
Fórmula	(desde i=2 hasta n) Σ (nº de variantes soportadas / nº total de variantes potenciales soportadas) /n
Interpretación	Cuántas variantes son soportadas por la plataforma de manera que el cliente puede elegir alguna de ellas en caso de que lo necesite.
Tipo de escala	Razón
Audiencia destino	Proveedor, Cliente

ID	M3
Nombre	Capacidad computacional de la plataforma
Finalidad	Capacidad computacional de la plataforma cloud
Fórmula	Tiempo real de uso de la plataforma / tiempo esperado de uso de la plataforma
Interpretación	1 sería el tiempo esperado, por debajo sería mejor de lo esperado y por encima peor de lo esperado.
Tipo de escala	Intervalo
Audiencia destino	Proveedor, Cliente

ID	M4
Nombre	Consistencia durante un período de tiempo
Finalidad	Consistencia de la plataforma cloud
Fórmula	Cualquier desviación de los valores prometidos de velocidad de la CPU, memoria, etc. durante un período de tiempo.
Interpretación	Cualquier desviación se considera fallo.
Tipo de escala	Razón
Audiencia destino	Proveedor, Cliente

Métricas de Eficiencia de Desempeño

ID	M5
Nombre	Promedio de capacidad de la CPU libre de todos los hosts que se ejecutan
Finalidad	Capacidad latente del servicio
Fórmula	$Dyn = \frac{\sum_{i=0}^{n_H} \xi_{f,cpu}^{\vec{n}_i \cdot \vec{k}} - \omega_{f,cpu}^{\vec{n}_i \cdot \vec{k}}}{n_H}$
Interpretación	Cantidad de capacidad computacional disponible sin necesidad de encender nuevos hosts.
Tipo de escala	Razón
Audiencia destino	Proveedor, Bróker

ID	M6
Nombre	Eficiencia de respuesta
Finalidad	Eficiencia de la respuesta
Fórmula	Nº de trabajos completados / Nº de trabajos aceptados
Interpretación	Valor entre 0 y 1. Cuanto más cercano a 1 mejor.
Tipo de escala	Razón
Audiencia destino	Proveedor, Cliente

ID	M7
Nombre	Latencia
Finalidad	Latencia del servicio
Fórmula	Tiempo en el que se hace la solicitud – Tiempo en el que se produce el resultado de esa solicitud
Interpretación	Latencia desde que se inicia la solicitud hasta que se completa
Tipo de escala	Razón
Audiencia destino	Cliente

ID	M8
Nombre	Retraso de las reacciones de ida y vuelta
Finalidad	Latencia del servicio
Fórmula	Tiempo en el que se manda el evento – Tiempo en el que se obtiene el resultado
Interpretación	Latencia desde que se manda el evento hasta que se completa
Tipo de escala	Razón
Audiencia destino	Cliente

ID	M9
Nombre	Tiempo de iteración del bucle principal
Finalidad	Tiempo de ejecución del artefacto cloud
Fórmula	Tiempo en el que los datos de la escena son actualizados – Tiempo en el que se simulan las físicas
Interpretación	Cuanto más bajo mejor
Tipo de escala	Razón
Audiencia destino	Cliente, Proveedor

ID	M10
Nombre	Tiempo dedicado al motor de físicas
Finalidad	Tiempo de ejecución del artefacto cloud
Fórmula	Tiempo del resultado de simulación de las físicas – Tiempo en el que se simulan las físicas
Interpretación	Cuanto más bajo mejor
Tipo de escala	Razón
Audiencia destino	Cliente

ID	M11
Nombre	Tiempo dedicado al motor de renderizado
Finalidad	Tiempo de ejecución del artefacto cloud
Fórmula	Tiempo en el que los datos de la escena son actualizados – Tiempo en el que la escena se actualiza
Interpretación	Cuanto más bajo mejor
Tipo de escala	Razón
Audiencia destino	Cliente

ID	M12
Nombre	Tiempo de espera
Finalidad	Tiempo de espera del servicio
Fórmula	$\mathcal{P}_i = \frac{1}{\omega_t \max_{t_{ij} \in t_i}\{t_{ij}\} + \omega_c \sum_{j=1}^{m} c_{ij} + \omega_e \sum_{j=1}^{m} e_{ij}}$
Interpretación	Cantidad de tiempo en el que se tarda en preparar una solicitud.
Tipo de escala	Razón
Audiencia destino	Cliente, Proveedor

ID	M13
Nombre	Tiempo promedio del tiempo de respuesta
Finalidad	Tiempo de respuesta del servicio
Fórmula	$\sum_i T_i/n$ donde Ti es el tiempo entre que el usuario i solicita un servicio y el momento en el que el servicio está disponible
Interpretación	Tiempo promedio entre la solicitud de un servicio y su disponibilidad
Tipo de escala	Razón
Audiencia destino	Cliente, Proveedor

ID	M14
Nombre	Tiempo de respuesta máximo
Finalidad	Tiempo de respuesta del servicio
Fórmula	Tiempo de respuesta máximo asegurado por la plataforma cloud
Interpretación	Tiempo de respuesta máximo
Tipo de escala	Razón
Audiencia destino	Cliente, Proveedor

ID	M15
Nombre	Tiempo de respuesta
Finalidad	Tiempo de respuesta del servicio
Fórmula	$\sum_i T_i/n$ donde Ti es el tiempo entre que el usuario i realiza una solicitud y la respuesta está disponible
Interpretación	Tiempo promedio entre la realización de una solicitud y disponibilidad de la respuesta
Tipo de escala	Razón
Audiencia destino	Cliente, Proveedor

ID	M16
Nombre	Tiempo de retraso
Finalidad	Tiempo de retraso del servicio
Fórmula	Tws = (Tl, Tm, Tu) donde Tl representa el retraso mínimo en un cierto período de tiempo, Tm el retraso medio y Tu el retraso máximo
Interpretación	-
Tipo de escala	Razón
Audiencia destino	Cliente, Proveedor

ID	M18
Nombre	Ancho de banda la red
Finalidad	Ancho de banda del servicio
Fórmula	Bits transferidos en un segundo / Bits recibidos en un segundo
Interpretación	Bits / segundos
Tipo de escala	Razón
Audiencia destino	Bróker

ID	M19
Nombre	Disponibilidad del ancho de banda
Finalidad	Ancho de banda del servicio
Fórmula	$$bandwidth\ blocking\ rate = \frac{\sum Bandwidth\ of\ rejected\ requests}{\sum Bandwidth\ of\ arrived\ requests}$$
Interpretación	Porcentaje de ancho de banda no disponible. Se obtiene el disponible restándole a uno el resultado obtenido.
Tipo de escala	Razón
Audiencia destino	Proveedor

ID	M20
Nombre	Coste de energía
Finalidad	Coste de la energía del servicio
Fórmula	$P(F_i)=\alpha(P_{full}(F_i)-P_{idle}(F_i))+P_{idle}(F_i)$ donde α es la carga de la CPU, $P_{full}(F_i)$ y $P_{idle}(F_i)$ son la potencia dada por el host usando la frecuencia Fi al 100% y 0% del uso de la CPU, respectivamente.
Interpretación	
Tipo de escala	Razón
Audiencia destino	Bróker

ID	M21
Nombre	Almacenamiento de datos
Finalidad	Coste en curso del servicio
Fórmula	$\frac{p}{cpu^a * net^b * data^c * RAM^d}$ donde cpu son las unidades de red cpu, net son las unidades de red, data la cantidad de datos y RAM la cantidad de memoria. Las letras a, b, c y d son los pesos dados a cada atributo. La suma de ellos debe ser 1. En nuestro deberíamos darle todo el peso a data.
Interpretación	Coste del almacenamiento de datos.
Tipo de escala	Razón
Audiencia destino	Bróker

ID	M22
Nombre	Uso computacional
Finalidad	Coste en curso del servicio
Fórmula	$\frac{p}{cpu^a * net^b * data^c * RAM^d}$ donde cpu son las unidades de red cpu, net son las unidades de red, data la cantidad de datos y RAM la cantidad de memoria. Las letras a, b, c y d son los pesos dados a cada atributo. La suma de ellos debe ser 1. En este caso deberíamos darle todo el peso a CPU.
Interpretación	Coste de la CPU.
Tipo de escala	Razón
Audiencia destino	Bróker

ID	M23
Nombre	Coste de almacenamiento
Finalidad	Coste en curso del servicio
Fórmula	$\sum_{j=1}^{k} c_j * D_j$ donde Dj el total de datos que hay que almacenar en la tarea j y Cj es el tiempo de ejecución de la tarea j.
Interpretación	Coste total de almacenamiento
Tipo de escala	Razón
Audiencia destino	Bróker

ID	M24
Nombre	Potencia consumida por la CPU
Finalidad	Eficiencia energética del servicio
Fórmula	$P_{idle} + (P_{busy} - P_{idle}) * u$ donde Pidle denota la potencia consumida por el servidor inactivo, Pbusy denota la potencia consumida por el servidor cuando se utiliza al 100% y u representa la potencia consumida en el momento actual.
Interpretación	Potencia consumida por la CPU
Tipo de escala	Razón
Audiencia destino	Bróker

ID	M25
Nombre	Consumo total de energía
Finalidad	Eficiencia energética del servicio
Fórmula	Total Energy consumption = (Fixed energy consumption + Storage energy consumption + Computation energy consumption + Communication Energy consumption)
Interpretación	Consumo total de energía
Tipo de escala	Razón
Audiencia destino	Bróker

ID	M26
Nombre	Eficiencia de un proveedor del servicio
Finalidad	Eficiencia del servicio
Fórmula	$e_{app} = \frac{T}{T - T_o}$ donde T representa la ejecución de las tareas y To la sobrecarga de la plataforma.
Interpretación	
Tipo de escala	Razón
Audiencia destino	Bróker

ID	M27
Nombre	Utilización de recursos
Finalidad	Eficiencia del servicio
Fórmula	Cantidad de recursos asignados / Cantidad de recursos predefinidos
Interpretación	Un valor entre 0 y 1. Cuanto más alto más recursos compartidos tiene la plataforma.
Tipo de escala	Razón
Audiencia destino	Bróker

ID	M28
Nombre	Rendimiento de un proveedor de servicio
Finalidad	Rendimiento (*Throughput*) del servicio
Fórmula	$Thr_{app} = \frac{n}{T+T_o}$ donde n es una colección de tareas, T representa la ejecución de las tareas y To la sobrecarga de la plataforma.
Interpretación	
Tipo de escala	Razón
Audiencia destino	Bróker

ID	M31
Nombre	Capacidad del CPU
Finalidad	Capacidad del proveedor del servicio
Fórmula	$cpu = \{cpu_u, cpu_{sp}\}$ donde cpuu es la cpu demandada y cpusp es el servicio ofrecido.
Interpretación	
Tipo de escala	Razón
Audiencia destino	Cliente

ID	M32
Nombre	Capacidad de la memoria
Finalidad	Capacidad del proveedor del servicio
Fórmula	$mem = \{mem_u, mem_{sp}\}$ donde memu es la memoria demandada y memsp es el servicio ofrecido.
Interpretación	
Tipo de escala	Razón
Audiencia destino	Cliente

ID	M33
Nombre	Capacidad de almacenamiento
Finalidad	Capacidad del proveedor del servicio
Fórmula	$sto = \{sto_u, sto_{sp}\}$ donde stou es la memoria demandada y stosp es el servicio ofrecido.
Interpretación	
Tipo de escala	Razón
Audiencia destino	Cliente

ID	M34
Nombre	Capacidad de la red
Finalidad	Capacidad del proveedor del servicio
Fórmula	$BW = \{BW_u, BW_{sp}\}$ donde BWu es la memoria demandada y BWsp es el servicio ofrecido.
Interpretación	
Tipo de escala	Razón
Audiencia destino	Cliente

ID	M36
Nombre	Número de núcleos de la CPU
Finalidad	Desempeño de la CPU del servicio
Fórmula	Nº de núcleos de la CPU
Interpretación	
Tipo de escala	Razón
Audiencia destino	Cliente

ID	M38
Nombre	Velocidad de transmisión de datos alcanzada
Finalidad	Desempeño de la transmisión de datos del servicio
Fórmula	Valor medio de la ratio de transmisión
Interpretación	
Tipo de escala	Razón
Audiencia destino	Cliente

Métricas de Fiabilidad

ID	M52
Nombre	Porcentaje del tiempo en línea
Finalidad	Disponibilidad
Fórmula	t / ts
Interpretación	t es el tiempo total y ts es el tiempo del período que ha estado operativo
Tipo de escala	Razón
Audiencia destino	Proveedor, Cliente

ID	M56
Nombre	Número medio de máquinas físicas en modo activo
Finalidad	Número de máquinas físicas
Fórmula	$r_m = \sum_{i=1}^{M} \#P^m_{active_i}$
Interpretación	Número medio de máquinas físicas en modo activo durante un período concreto de tiempo
Tipo de escala	Razón
Audiencia destino	Proveedor

Métricas de Seguridad

ID	M70
Nombre	Número de alarmas falsas monitoreadas por la seguridad corporativa
Finalidad	Protección
Fórmula	Número de alarmas falsas monitoreadas por la seguridad corporativa en un período concreto de tiempo
Interpretación	
Tipo de escala	Razón
Audiencia destino	Cliente

ID	M71
Nombre	Número de respuestas de servicio ineficaces a los problemas identificados por la Seguridad como debilidades de control
Finalidad	Protección
Fórmula	Número de respuestas de servicio ineficaces a los problemas identificados por la Seguridad como debilidades de control en un período concreto de tiempo
Interpretación	
Tipo de escala	Razón
Audiencia destino	Cliente

ID	M72
Nombre	Número de riesgos de seguridad proactiva identificados
Finalidad	Protección
Fórmula	Número de riesgos de seguridad proactiva identificados en un período concreto de tiempo
Interpretación	
Tipo de escala	Razón
Audiencia destino	Cliente

ID	M73
Nombre	Porcentaje de recursos de datos peligrosos que residen en los sistemas
Finalidad	Protección
Fórmula	Porcentaje de recursos de datos peligrosos que residen en los sistemas
Interpretación	
Tipo de escala	Razón
Audiencia destino	Cliente

ID	M76
Nombre	Integridad de los datos
Finalidad	Integridad de los datos
Fórmula	$Data\ Integrity\ (DI)\ \ R_k = \frac{D_k}{C_k}$ Dónde Ck es el nº de trabajos completados por la fuente Rk y Dk es el nº de trabajos de integridad proporcionados por Rk en un período de tiempo concreto.
Interpretación	Se obtiene un valor entre 0 y 1 siendo 1 el mejor valor.
Tipo de escala	Razón
Audiencia destino	Cliente

Métricas de Portabilidad

ID	M82
Nombre	Portabilidad
Finalidad	Portabilidad
Fórmula	Cantidad de marcos de trabajo que soportan aplicaciones desarrolladas en otros marcos
Interpretación	
Tipo de escala	Razón
Audiencia destino	Cliente

ID	M89
Nombre	Capacidad máxima
Finalidad	Elasticidad
Fórmula	La capacidad es el número máximo de unidades de computación que pueden ser proporcionadas en horas punta.
Interpretación	
Tipo de escala	Razón
Audiencia destino	Cliente

ID	M92
Nombre	Escalabilidad vertical de cloud
Finalidad	Escalabilidad
Fórmula	$\sum_i^m \sum_j^{n_i} (proportion\ of\ increase\ inr_{ij})$. Dónde Rij representa al recurso j que necesita ser mejorado en el servicio cloud i. Donde n y m son el número de recursos asignados a un servicio cloud particular y el número de servicios cloud usados por el usuario, respectivamente.
Interpretación	
Tipo de escala	Razón
Audiencia destino	Cliente

Las métricas descritas en esta sección proporcionan mecanismos cuantitativos para seleccionar plataformas cloud de acuerdo a su capacidad en satisfacer un conjunto de características y atributos de calidad propuestos en la literatura. Aunque el método de selección de plataformas cloud propuesto en este trabajo considera ese conjunto de métricas como punto de partida, éste puede ser fácilmente extendido para soportar nuevas métricas.

3.3. Conclusiones

Actualmente existe una diversidad de criterios adoptados por la comunidad científica para referirse a las características de calidad de los servicios cloud. Aunque todos coinciden en la importancia que significan estas características, no todos hacen la misma interpretación de las mismas. La propuesta más aceptada es el CSMIC SMI ya que recoge un buen número de indicadores y características, aunque la propuesta carece de métricas para medir estas características (solo se ha propuesto métricas para medir la usabilidad). La mayoría de modelos están orientados a la evaluación de la calidad de servicios en la nube.

Como consecuencia, podemos concluir que no están muy extendidos los modelos de calidad en la actualidad sobre las plataformas de cloud computing, y, en consecuencia, podemos observar la falta de mecanismos cuantitativos (métricas) para comparar las plataformas cloud de manera objetiva.

Debido a esto, hemos propuesto un modelo calidad para la comparación y selección de plataformas cloud. El modelo está soportado por un catálogo de métricas de calidad que permitirán a los usuarios elegir con criterio la plataforma que más se adecue a sus necesidades.

4. Sistema de Recomendación de Plataformas Cloud

En esta sección discutiremos los sistemas de recomendación de plataformas cloud existentes e implementaremos nuestro propio sistema de recomendación basado en el modelo de calidad que hemos extendido en la sección anterior.

4.1. Sistemas de Recomendación

Un sistema de recomendación es un sistema que *proporciona a los usuarios una serie de sugerencias personalizadas (recomendaciones) sobre un determinado tipo de elementos (ítems).* Los sistemas de recomendación estudian los objetivos y características de los usuarios y mediante un procesamiento de los datos, encuentra un subconjunto de ítems que pueden resultar de interés para el usuario.

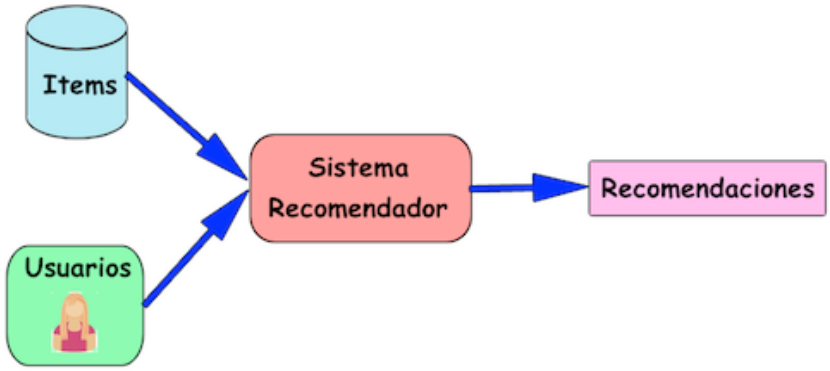

Figura 7. Elementos de un Sistema de Recomendación[3].

En los últimos años y debido principalmente a la sobre carga de información que tenemos en internet, han proliferado los sistemas de recomendación para ayudar a los usuarios a encontrar la información (contenidos) de mayor interés y más valiosa para los usuarios de una forma más rápida y eficiente.

Existen diferentes formas de realizar recomendaciones, en función de la navegación o compras de los usuarios o en función de su perfil (edad, preferencias, sexo, profesión) o que el usuario vote.

En este trabajo, el objetivo es diseñar un sistema de recomendación de plataformas cloud basada en el modelo de calidad propuesto en la sección 3. Por lo tanto, el sistema de recomendación se basará en un conjunto de características de calidad externas. En un futuro, el sistema de recomendación podrá ser extendido para considerar además las características del comportamiento de los usuarios (calidad en uso y experiencia de usuario).

4.2. Análisis de los Sistemas de Recomendación Existentes

Hemos encontrado un sistema de recomendación para el mercado de Cloud Computing [67] que para recomendar un proveedor u otro se basa en la calidad de servicio en relación con las *conexiones de red*, el *rendimiento* de las máquinas virtuales y las *opiniones* de los usuarios.

La principal razón por la que recomiendan en base a la calidad de servicio en relación con las conexiones de red es porque el Cloud Computing es un servicio web, por lo tanto, esta es una de las principales facetas a evaluar cuando trabajamos con servicios cloud. Otro punto importante evaluado es el rendimiento de las máquinas virtuales. Las máquinas virtuales ofrecen una gran flexibilidad y eficiencia a la hora de ajustar dinámicamente los recursos demandados por el usuario.

Por último, el sistema de recomendación tiene en cuenta las opiniones de los usuarios para clasificar los servicios Cloud Computing y también el coste de estos servicios, ya que los usuarios tienden a elegir los servicios de menor coste. La clasificación final se realiza por un valor que llaman S-Rank, el cual es calculado sumando la puntuación en los apartados mencionados. El usuario puede dar más peso a las características que considere más importantes para él.

El artículo nos presenta un caso de estudio en el que se establecen unos requisitos de ejemplo de un posible usuario y las características de posibles proveedores cloud. Primero se seleccionan los 4 proveedores de menor servicio y, por último, se ordenan por el S-Rank.

En [68] se propone un sistema que evalúa la nube en base a requisitos difusos de calidad de servicios de los usuarios y al desempeño dinámico de los servicios con el fin de contribuir a la selección de servicios. También hacen una propuesta de atributos para evaluar los servicios cloud. Dicha propuesta se basa en un modelo jerárquico que considera las siguientes características de calidad: desempeño, aseguramiento y elasticidad.

La principal desventaja de estos sistemas es que se basan en un número reducido de características. La propuesta tampoco considera distintos perfiles de usuario.

4.3. Diseño del Sistema de Recomendación

El sistema de recomendación debe tener buenas técnicas de *representación de las preferencias o recomendaciones de los usuarios* para poder captar verdaderamente el concepto del contenido recomendado. Además, la *interfaz del sistema de recomendación tiene que ser atractiva* y debe mejorar la interacción sistema de recomendación-usuario.

Basándonos en un estudio que analiza los sistemas de recomendación existentes hemos sacado dos conclusiones que nos han ayudado a diseñar nuestro sistema de recomendación. La primera conclusión es que podríamos usar listas para mostrar los resultados y quizás algunas gráficas cuando se comparen ciertos requisitos. Esto ayuda al usuario a entender mejor las recomendaciones que se realizan, de acuerdo con el estudio [67].

La segunda conclusión que el estudio recomienda es explicar las recomendaciones que se dan al final y añadir información acerca del contexto, como podría ser descripciones de las plataformas o las características que se van a medir.

El sistema va a seguir la ISO/IEC 25040 [19], el cual define el proceso para llevar a cabo la evaluación de un producto software. A continuación, detallaremos como nuestro sistema va a cumplir con las distintas actividades a seguir del proceso.

Hemos desarrollado un prototipo sencillo con las distintas pantallas que tendrá la aplicación final. La versión final de la aplicación tiene, lógicamente, variaciones con el prototipo y nos sirve para ver el proceso real que se ha seguido desde que se diseña la aplicación hasta que se implementa. Introduciremos capturas del prototipo a lo largo de la sección.

La primera actividad consiste en ***establecer los requisitos del cliente*.** Los requisitos se definen en función de las necesidades de los usuarios. Para ello, hay que establecer el propósito de la evaluación *("medir la calidad de la plataforma Google App Engine, comparar el rendimiento de una plataforma con otra(s), decidir cuál plataforma utilizar, etc.")* para que el usuario obtenga una recomendación acorde a sus preferencias y necesidades.

Como parte de los requisitos, el usuario debería indicar:

- *Plataforma(s) a evaluar*: nombre de las plataformas cloud que serán evaluadas
- *Perfil del usuario:* proveedor, consumidor o bróker
- *Restricciones de la evaluación:* cualquier restricción relacionada al proceso de evaluación o a las plataformas cloud. Por ejemplo, se deben identificar y documentar las partes del producto (plataformas cloud) incluidas en la evaluación. Por último, se debe definir el rigor de la evaluación en función del propósito y el uso previsto de la plataforma cloud, basándose, por ejemplo, en aspectos como el riesgo para la seguridad, el riesgo económico o el riesgo ambiental.

Las figuras 8 y 9 indican la selección de plataformas a ser comparadas y el punto de vista de la evaluación.

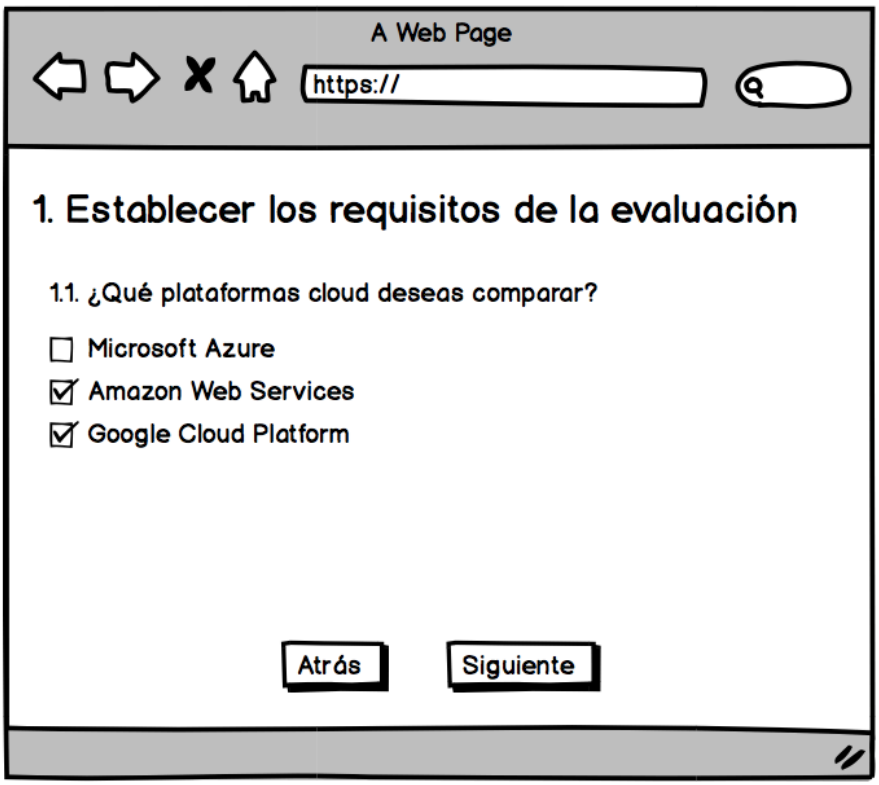

Figura 8. Pantalla de selección de plataformas en el prototipo.

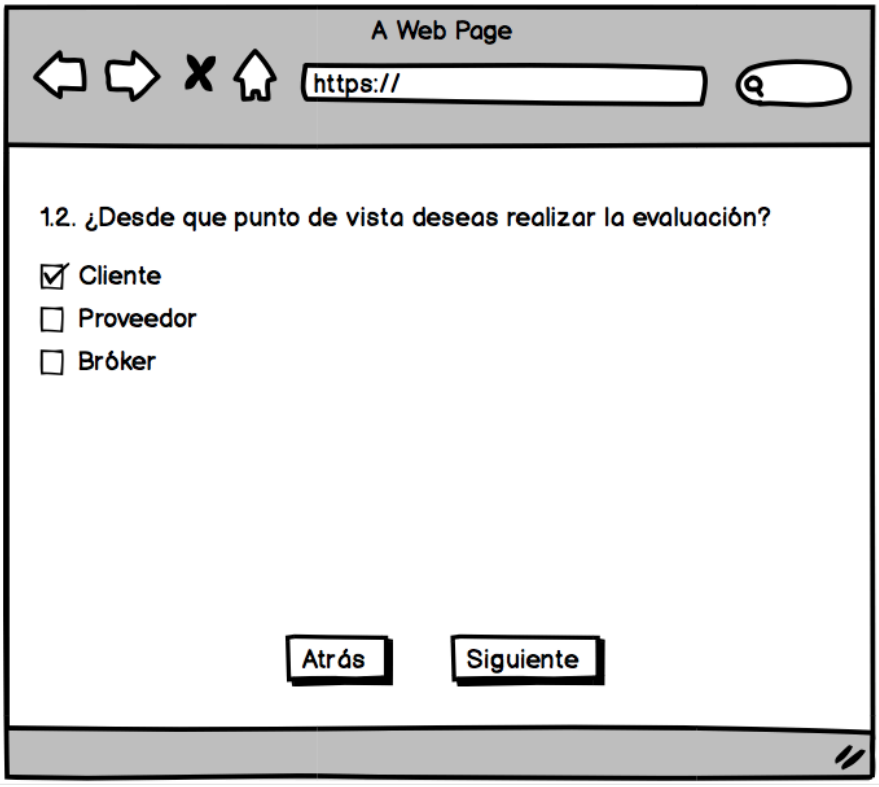

Figura 9. Pantalla de selección de rol en el prototipo.

A continuación, el usuario debe indicar el *contexto de la evaluación* (donde y como se tomarán las medidas). Las figuras 10 y 11 muestra las pantallas de introducción del contexto de evaluación y de las restricciones, siendo estas últimas opcionales.

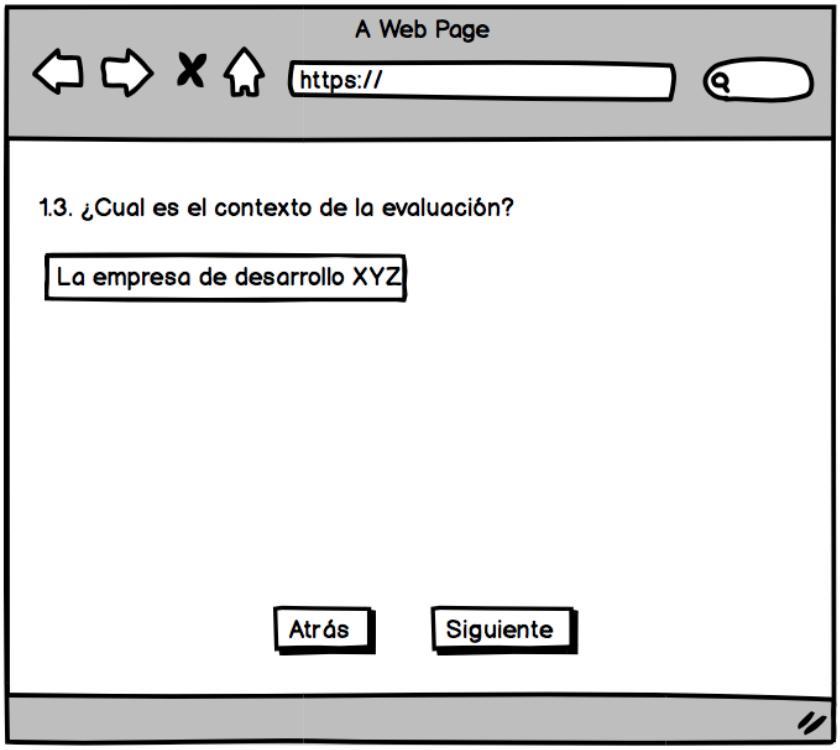

Figura 10. Pantalla de introducción del contexto en el prototipo.

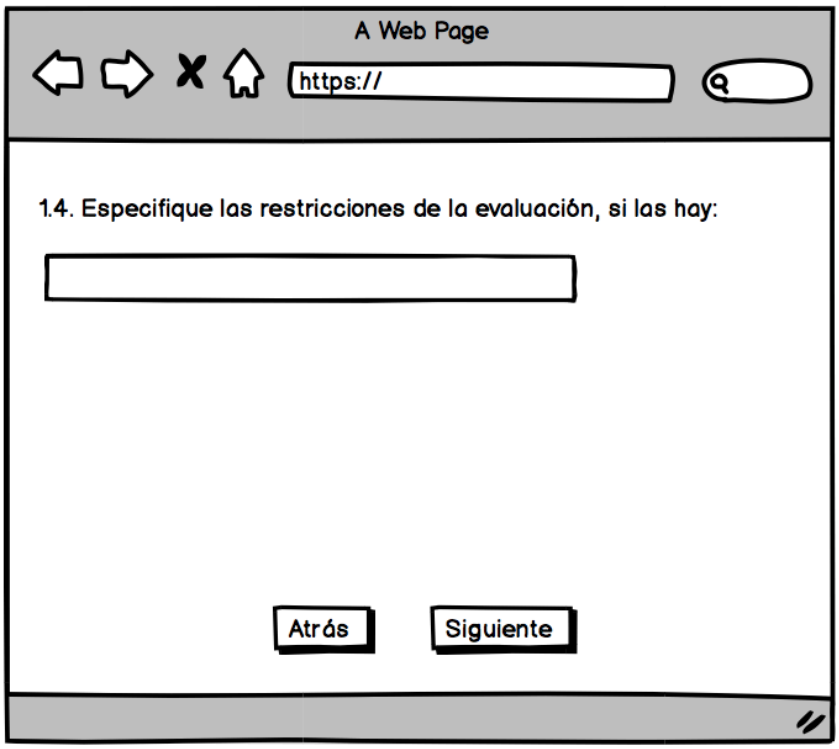

Figura 11. Pantalla de introducción de restricciones.

Como parte de la evaluación también hay que especificar cómo las plataformas serán comparadas, o sea, qué *características de calidad se desea evaluar* de las plataformas. En esta fase, el usuario podría seleccionar todas las características planteadas en el modelo de calidad o seleccionar solo algunas.

La segunda actividad consistirá en ***especificar la evaluación*** en la que el usuario indicará las características de calidad y para cada característica seleccionará las subcaracterísticas a ser comparadas. Las figuras 11 y 12 muestran la selección de características y subcaracterísticas.

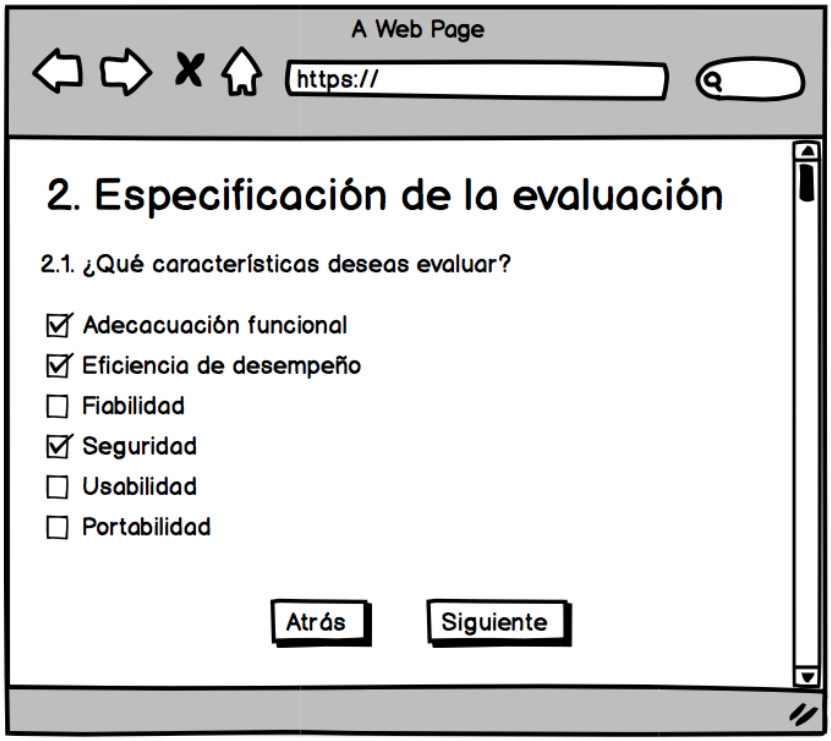

Figura 12. Pantalla de selección de características en el prototipo.

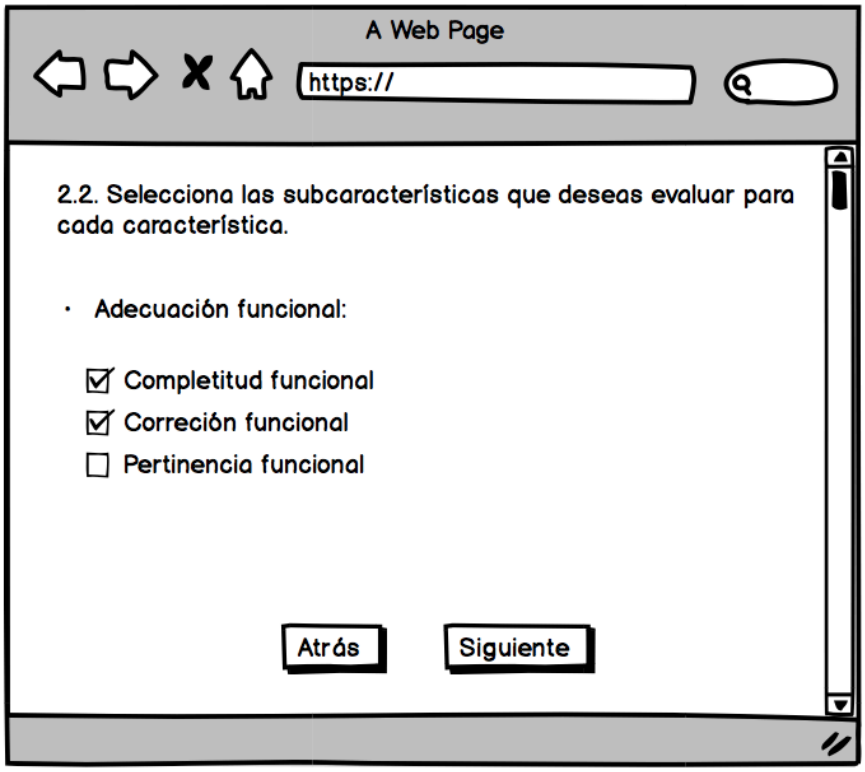

Figura 13. Pantalla de selección de subcaracterísticas en el prototipo.

A continuación, el usuario seleccionará los atributos de calidad para medir las subcaracterísticas deseadas y seleccionará las métricas que desea utilizar, así como establecerá los umbrales numéricos a la hora de realizar la evaluación de la plataforma cloud de acuerdo con su criterio y sus necesidades. Las figuras 13 y 14 ilustran el proceso de selección de atributos y métricas.

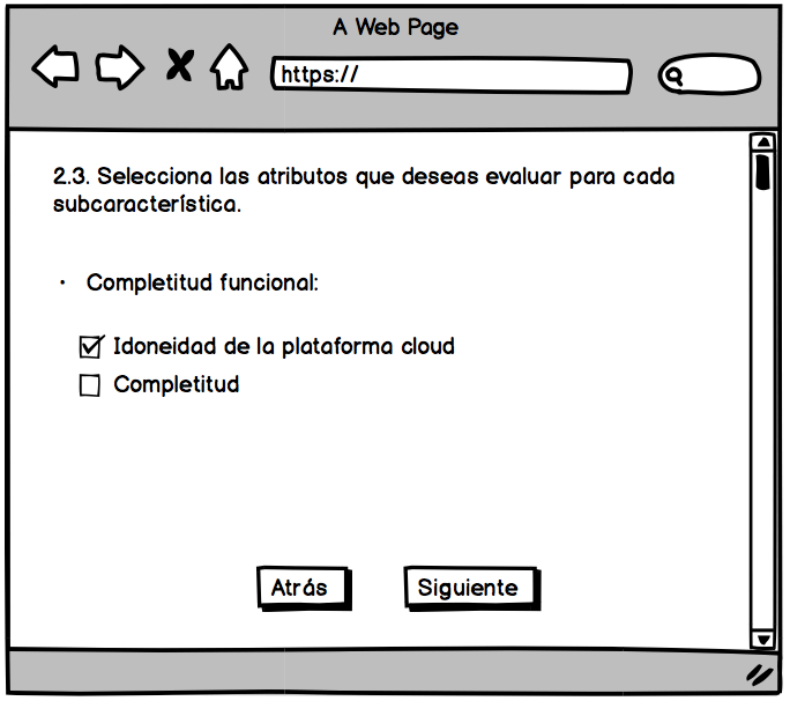

Figura 14. Pantalla de selección de atributos en el prototipo.

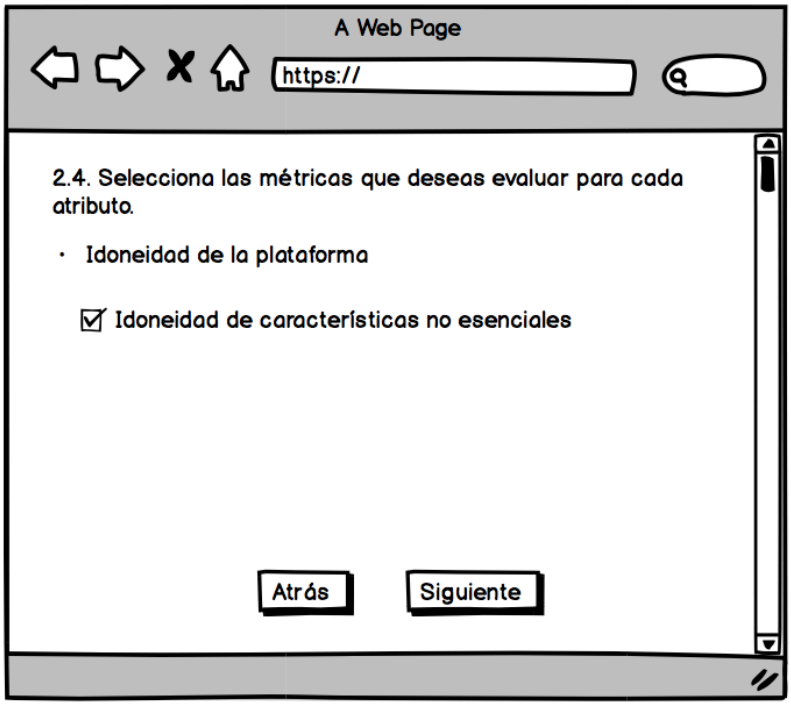

Figura 15. Pantalla de selección de métricas en el prototipo.

Los *criterios de decisión para las métricas* son umbrales numéricos que se pueden relacionar con los requisitos de calidad y posteriormente con los criterios de evaluación para decidir la calidad de la plataforma. Estos umbrales se pueden establecer a partir de *benchmarks*, límites de control estadísticos, datos históricos, requisitos del cliente, etc.

Para cada métrica, el experto del dominio puede definir los umbrales esperados que son utilizados por defecto. Por ejemplo, para la métrica "Coste de la Seguridad", asumiendo que una métrica puede obtener un valor entre 0 y 100 (y que valores más próximos a 0 mejor), el experto del dominio ha establecido los siguientes umbrales:

(0 < X ≤ 20): Excede los requisitos (es más de lo esperado)
(21 < X ≤ 40): Rango Objetivo
(41 < X ≤ 60): Mínimamente Aceptable
(60 < X ≤ 100): Inaceptable

La Figura 15 muestra cómo el usuario puede cambiar, para una determinada métrica, los umbrales establecidos por el experto del dominio para una evaluación de plataformas cloud específica. Ese paso se debe realizar para cada métrica seleccionada por el usuario.

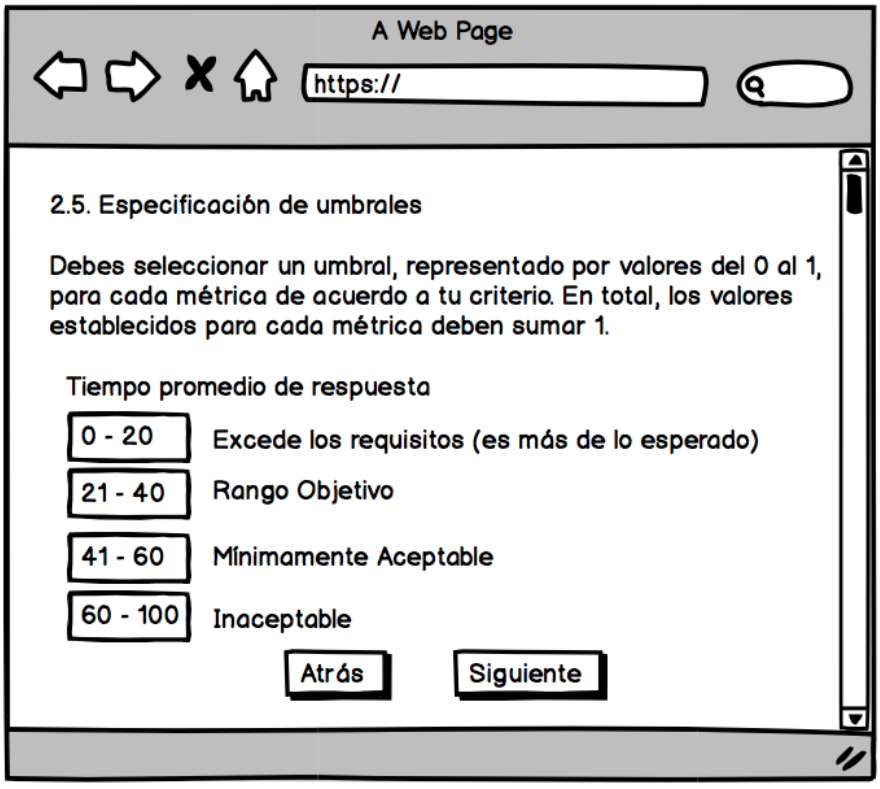

Figura 16. Pantalla de introducción de umbrales en el prototipo.

Finalmente, habrá que definir *criterios de decisión de la evaluación*. Se deben definir criterios para las diferentes características evaluadas a partir de las subcaracterísticas y métricas de calidad. Estos resultados a mayor nivel de

abstracción permiten realizar la valoración de la calidad de la plataforma cloud de forma general.

En este trabajo, adoptamos los criterios de decisión de la evaluación propuestos por Rodríguez y Piattini (2014) [65]. El valor de la calidad de una plataforma cloud se obtiene a partir de los valores de sus características que a su vez se obtienen a partir del valor de la calidad de sus atributos (o propiedad) de calidad.

La evaluación se realiza siguiendo un proceso *bottom-up*. A partir de los valores obtenidos en las métricas, se escala en la jerarquía al siguiente nivel para poder obtener un valor de calidad que asignarle al atributo y luego a la subcaracterística. Este valor se ha normalizado entre 0 y 100 utilizando una función por perfiles de manera que todos los atributos y subcaracterísticas tengan la misma escala, siendo 0 el valor más bajo de calidad que puede tener una subcaracterística o atributo y 100 el valor más alto.

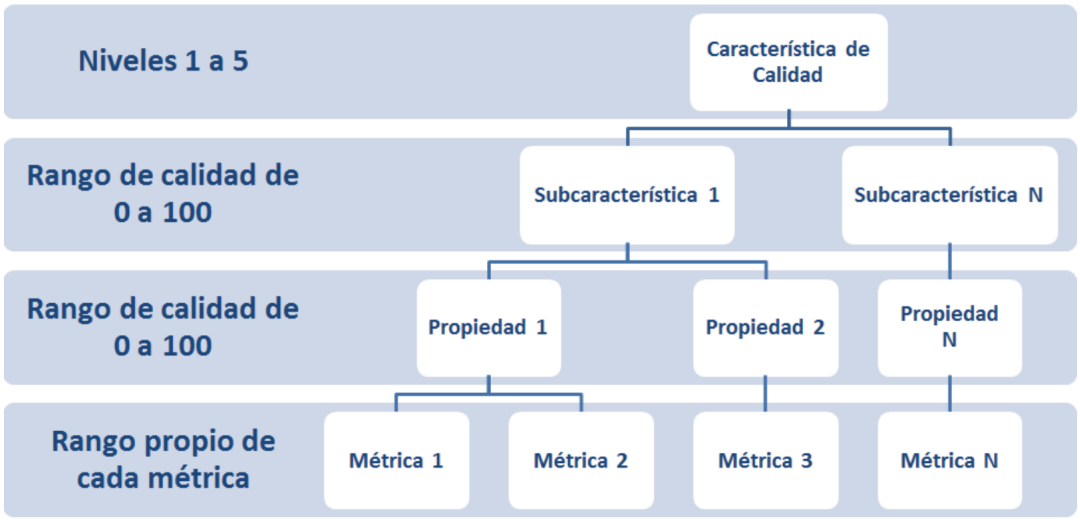

Figura 17. Jerarquía y elementos que influyen en la evaluación del producto software (fuente: Rodriguez y Piattini [61]).

Una vez se ha calculado el valor de todas la subcaracterísticas, estamos en disposición de calcular el nivel de calidad de la característica (p.ej., Adecuación Funcional). Para ello, se aplica una función de evaluación sobre el conjunto de valores de 0 a 100 que presentan el grupo de subcaracterísticas y se obtiene un valor definitivo para la característica de calidad. En este caso, el valor de calidad de las características se ha normalizado entre 1 y 5, inspirado en los modelos de madurez de procesos software como el CMMI o el SPICE.

La correspondencia de estos niveles de calidad con los valores normalizados de 0 a 100 se corresponden con los *criterios de decisión de la evaluación* propuestos por Rodriguez y Piattini que se muestran en la figura 17. Como se puede observar, un producto con una característica de calidad (adecuación funcional) en Nivel 1 habrá obtenido un valor de calidad entre 0 y 25, por lo que se podrá considerar como *Deficiente*. Una adecuación funcional en Nivel 3, habrá obtenido un valor de calidad entre 50 y 75, por lo que se podrá considerar *Buena*. Y una adecuación funcional en el

Nivel 5 habrá obtenido un valor de calidad entre 95 y 100, por lo que se podrá considerar como *Excelente*.

Tabla 7. Criterios de decisión para la calidad de plataformas cloud.

Nivel	Valor de Calidad	Descripción
1	0-25	Calidad Deficiente
2	25-50	Calidad Insuficiente
3	50-75	Calidad Buena
4	75-95	Calidad Muy Buena
5	95-100	Calidad Excelente

La tercera actividad consiste en ***diseñar la evaluación***, que consiste en la definición de un ***plan de actividades para realizar la evaluación.*** En nuestro caso consistirá en usar las métricas seleccionadas por el usuario y los umbrales para generar unas puntuaciones y poder realizar una recomendación. Por lo tanto, el *plan de evaluación* lista a todas las características y atributos cuantificables y que modelan a la calidad de la plataforma según las necesidades específicas del perfil de usuario seleccionado.

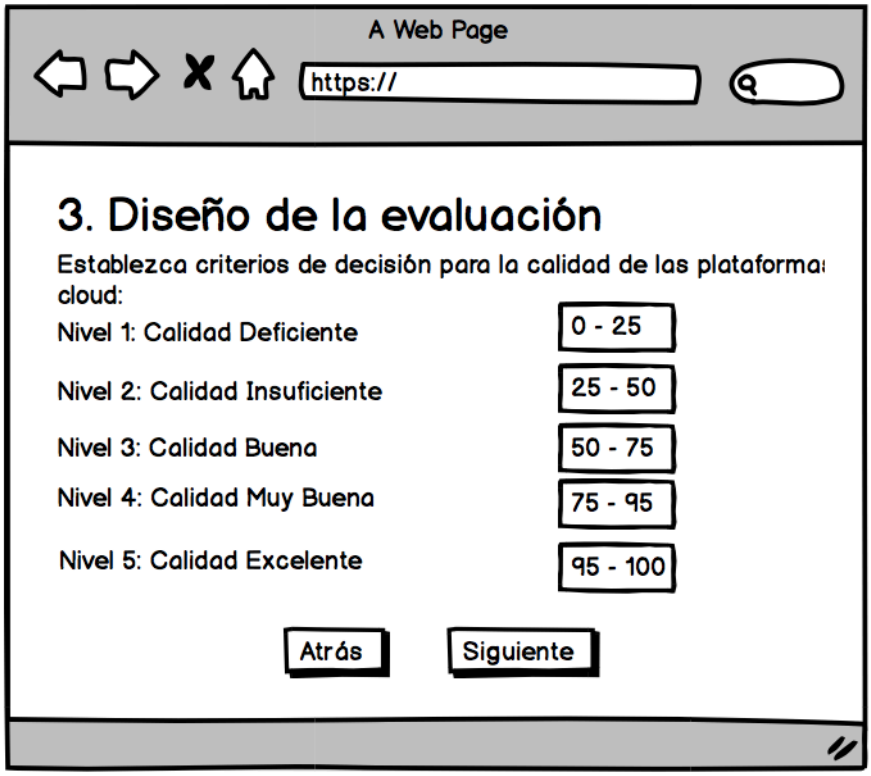

Figura 18. Pantalla de diseño de la evaluación en el prototipo.

En la cuarta actividad se ***ejecuta la evaluación***. Realizaremos las mediciones aplicando las métricas seleccionadas a las distintas plataformas cloud para ver el grado en el que son afines de acuerdo con las necesidades del usuario. En particular se realizan las siguientes tareas:

- *Realización de mediciones:* Se deben realizar las mediciones sobre la(s) plataforma(s) seleccionadas y sus componentes para obtener los valores de las métricas seleccionadas e indicadas en el plan de evaluación. Los resultados obtenidos deberán ser almacenados.
- *Aplicación de los criterios de decisión para las métricas:* se aplican los criterios de decisión para las métricas seleccionadas sobre los valores obtenidos en la medición de las plataformas.
- Aplicación de los criterios de decisión de la evaluación: se deben aplicar los criterios de decisión a nivel de características y subcaracterísticas de calidad, produciendo como resultado la valoración del grado en que la plataforma cloud cumple los requisitos de calidad establecidos.

En la última actividad se realiza un ***informe con los datos obtenidos en la evaluación*** en el que se muestra el grado en el que las plataformas cumplen con los requisitos establecidos por el usuario. Por último, se le permite al usuario mandar feedback a cerca del proceso de evaluación con el objetivo de poder mejorarlo en un futuro.

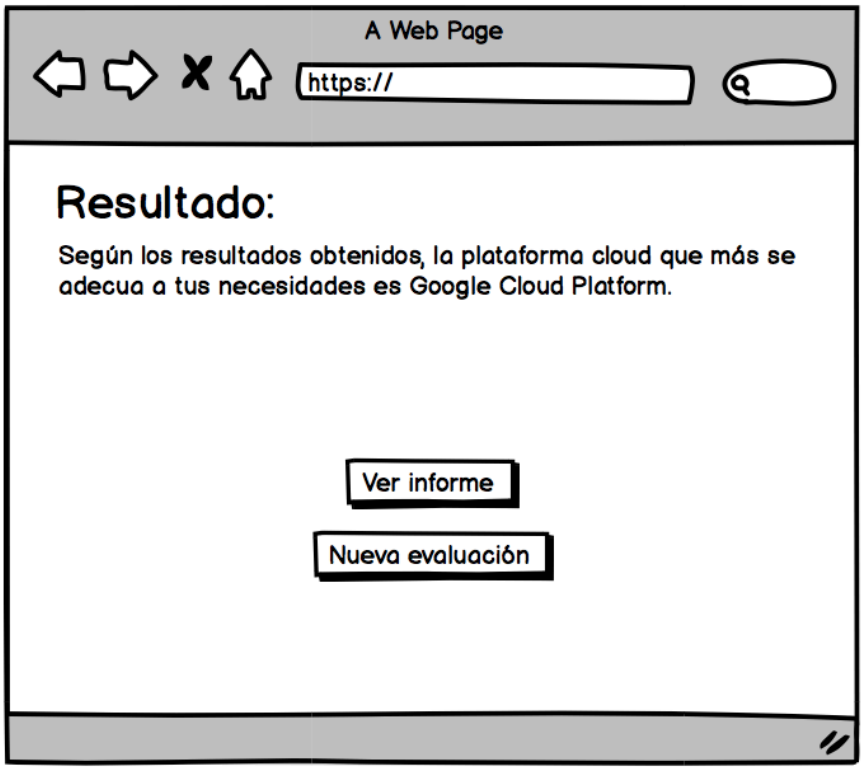

Figura 19. Pantalla de resultado de la evaluación en el prototipo.

Figura 20. Pantalla de selección de plataforma en el resultado de la evaluación en el prototipo.

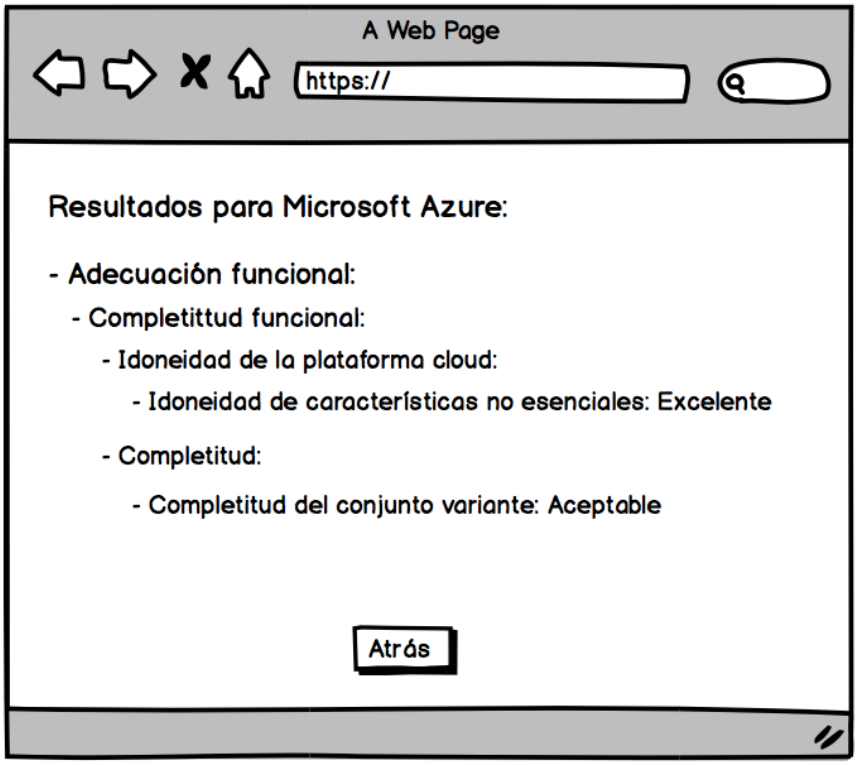

Figura 21. Pantalla de informe de la evaluación en el prototipo.

4.4. Implementación del Sistema de Recomendación

Para implementar el sistema hemos decidido desarrollar una aplicación web, de manera que sea fácilmente desplegable y accesible. Las principales tecnologías con las que hemos desarrollado la aplicación son las siguientes:

- HTML, CSS y JavaScript como lenguajes de programación principales.
- React.js, una librería de JavaScript para construir interfaces de usuario que se ha hecho popular hace relativamente poco tiempo [69].
- Redux.js, otra librería de JavaScript la cual se complementa muy bien con React.js, ya que React.js se encarga de la interfaz y Redux.js se encarga del estado de la aplicación [70].
- Bootstrap, para gestionar el estilo de la aplicación [71].

Estas son las principales tecnologías que hemos usado en el desarrollo de la aplicación. A parte, hemos usado otras librerías para ayudarnos en el desarrollo pero que tienen un menor protagonismo dentro de la aplicación.

En los enlaces que hemos referenciado se puede encontrar mucha documentación de React.js y Redux.js. En esta sección se dará una pequeña introducción a los distintos conceptos que se emplean en estas tecnologías, pero se recomienda leer la documentación para entender en detalle cómo funcionan.

La estructura de la aplicación es la siguiente:

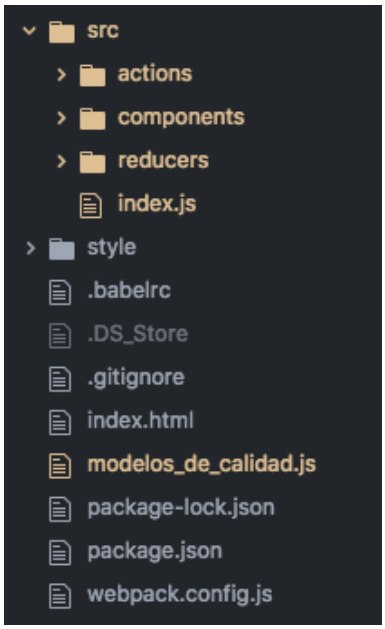

Figura 22. Pantalla de informe de la evaluación en el prototipo.

Dentro de la carpeta /src se encuentra todo el código de la aplicación. En la carpeta /style se encuentra el código CSS que se aplica a la aplicación, este código tiene mayor prioridad que los estilos aplicados por Bootstrap.

Después tenemos el archivo index.html, el cual es la única página html a la cual se accede en la aplicación, el código es el siguiente:

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link rel="stylesheet" href="/style/style.css">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
  </head>
  <body>
    <div class="container"></div>
  </body>
  <script src="/bundle.js"></script>
</html>
```

Figura 23. Página HTML de la aplicación.

La información que se encuentra en <meta> es utilizada por Bootstrap para la visualización de la aplicación. Para más detalle se puede leer la documentación en el enlace referenciado anteriormente.

Después incorporamos al html dos links del tipo “stylesheet”. El archivo css para el estilo de nuestra página y Bootstrap, para poder utilizarlo en toda la aplicación.

Después, en el cuerpo de la página, tenemos un con la clase “container”, el cual nos servirá para generar dinámicamente el contenido de la página mediante React como veremos más adelante.

Y por último tenemos el script /bundle.js, que es el archivo generado por la librería Webpack en el cual mete todo el código JavaScript de nuestra aplicación.

Después encontramos el archivo modelos_de_calidad.js. En este archivo podemos incluir los distintos modelos de calidad que queramos usar en la aplicación. En el archivo debemos exportar el modelo como una variable. El modelo se encuentra en formato JSON (JavaScript Object Notation) y podemos definir y exportar tantos modelos como queramos en este archivo. Más adelante veremos cómo seleccionar el modelo que queramos usar.

El formato sería el siguiente, en el primer nivel tenemos las características, en el siguiente las subcaracterísticas, en el siguiente los atributos y por último las métricas con su fórmula. Ejemplo:

export const modeloDeCalidadEjemplo: {

“Adecuación funcional”: {

“Corrección funcional”: {

“Capacidad computacional de la plataforma cloud”: {

“Capacidad computacional de la plataforma”:

“tiempo_real / tiempo_esperado”

}

}

}

}

Cómo se puede ver en este modelo tenemos la característica “Adecuación funcional”, con la subcaracterística “Corrección funcional”, con el atributo “Capacidad computacional de la plataforma cloud” y la métrica “Capacidad computacional de la plataforma” que tendría la fórmula “tiempo real / tiempo esperado.

Para leer las fórmulas utilizaremos JavaScript Expression Evaluator [72], mediante el cual podemos expresar las fórmulas texto y después podemos sustituir las variables para obtener un resultado. En este caso las variables serían “tiempo_real” y “tiempo_esperado”. Hay que darle un nombre auto explicativo a las variables, de manera que se entienda que representan.

Por último, tendríamos el archivo package.json [73], el cual, su principal función es gestionar las dependencias de nuestro proyecto, como son todas las tecnologías que hemos mencionado anteriormente.

Estos serían todos los archivos importantes que tenemos en la raíz. A continuación, vamos a explicar el contenido de la carpeta /src.

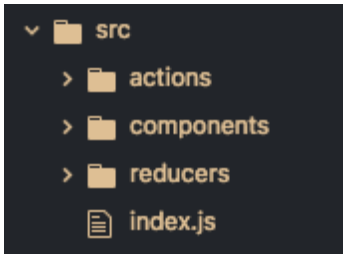

Figura 24. Estructura del código de la aplicación.

Primero es conveniente dar una introducción a los distintos elementos que tenemos en nuestro proyecto utilizando React.js y Redux.js. Lo más importante es que no podemos modificar el estado global de la aplicación directamente, para ello necesitamos las acciones. Encontramos las acciones dentro de la carpeta /actions.

Las acciones sirven para ser lanzadas desde cualquier punto de la aplicación. Cuando una acción es lanzada es capturada por un "reducer", que se encuentran en la carpeta /reducers. Un "reducer" captura todas las acciones que son lanzadas y dependiendo de la acción que sea realiza unos cambios en el estado de la aplicación u otros. Por último, tenemos los componentes en la carpeta /components. Los componentes renderizan el contenido de la aplicación, lo que el usuario puede ver.

Resumiendo, tenemos los componentes que es la parte visual. Desde los componentes podemos lanzar acciones que son capturadas por los "reducers" para cambiar el estado global de la aplicación.

Al principio es complicado de entender, por ello recomiendo ir a la documentación de estas tecnologías para leer con mayor profundidad y comprender como funciona. Una vez entendido como funcionan es muy sencillo. Las ventajas de utilizar estas tecnologías es que tenemos un estado global de la aplicación consistente que no puede ser modificado aleatoriamente, si no que se modifica únicamente en los "reducers". Y mediante los componentes tenemos la posibilidad de gestionar de manera muy sencilla y dinámica lo que el usuario puede ver. De manera que cada vez que cambiamos el estado global de la aplicación los componentes se vuelven a renderizar, reflejando los cambios de este.

Como digo, es complicado de entender por primera vez y recomiendo ir a la documentación oficial.

Ahora pasaré a explicar un ejemplo de acción, un ejemplo de un componente y un ejemplo de un reducer.

```
export const LEER_MODELO = 'LEER_MODELO';
export function leerModelo(modelo) {
    return {
        type: LEER_MODELO,
        payload: modelo
    }
}
```

Figura 25. Ejemplo de una acción de la aplicación.

Esta sería la acción encargada de leer el modelo de calidad seleccionado. La variable constante que exportamos es el identificador de la acción. La función que se exporta, cuando es ejecutada genera una nueva acción que básicamente es un objeto que contiene dos propiedades: "type", el identificador de la acción y "payload", lo que queramos pasar al "reducer". En este caso la función tiene como parámetro "modelo", el cual pasamos dentro de "payload".

En cuanto a los componentes, se explica el código más importante de un componente que sería el método render(), el cual se encarga de renderizar lo que el usuario puede ver y también pondré un ejemplo de cómo lanzamos una acción.

```
27      render() {
28        return (
29            <div>
30              <form onSubmit={this.handleOnSubmit}>
31                <div className="row mt-3">
32                    <h2>1. Establecer los requisitos de la evaluación</h2>
33                </div>
34                <div className="row mt-3 mb-1">
35                    1.1. ¿Qué plataformas cloud deseas comparar?
36                </div>
37
38                { this.createCheckboxes() }
39
40                <div className="row mt-5 text-center">
41                  <div className="col">
42                    <Link to="/" className="btn btn-secondary mr-4">Atrás</Link>
43                    <button type="submit" className="btn btn-primary"
44                      disabled={this.state.plataformasSeleccionadas.length === 0}>
45                      Siguiente
46                    </button>
47                  </div>
48                </div>
49              </form>
50            </div>
51        );
52      }
```

Figura 26. Ejemplo de un componente de la aplicación.

Como podemos observar, el método render() devuelve una pieza de código HTML. Todo el código que se encuentra entre estas llaves { } es código JavaScript que es ejecutado por React.js antes de renderizar el componente.

Este es el componente encargado de seleccionar las plataformas. Es un formulario con un título, unas checkboxes y dos botones. Las checkboxes son generadas mediante otro método cómo se puede ver en "{ *this.createCheckboxes()* }" el cual básicamente devuelve más código HTML.

Uno de los botones es un elemento <Link> el cual simplemente un botón que enlaza a la dirección indica en la propiedad "to", en nuestro enlaza a la raíz.

```
82      handleOnSubmit(event) {
83        event.preventDefault();
84        this.props.seleccionarPlataformas(this.state.plataformasSeleccionadas)
85        this.props.history.push('/seleccion-rol');
86      }
87  }
```

Figura 27. Ejemplo de un método que gestiona un evento de la aplicación.

Este es el método encargado de gestionar el evento "onSubmit" del formulario. Mediante "*this.props.seleccionarPlatataformas(this.state.plataformasSeleccionadas)*" lanzamos una acción encargada de modificar el estado de la aplicación añadiendo las plataformas que han sido seleccionadas por el usuario. La última línea de código simplemente nos lleva a la dirección especificada, en este caso nos llevaría a la selección del rol.

```
23  export default function(state = initialState, action) {
24      switch (action.type) {
25          case SELECCIONAR_PLATAFORMAS:
26              return {
27                  ...state,
28                  plataformas: action.payload
29              };
```

Figura 28. Ejemplo de un "reducer" de la aplicación.

Este sería uno de los "reducers" de nuestra aplicación. En este caso podemos ver lo que ocurre cuando se lanza la acción de seleccionar plataformas. Los "reducers" siempre devuelven un objeto nuevo el cual es el nuevo estado de la aplicación. En nuestro devolvemos el anterior estado (*…state*) sobrescribiendo la propiedad "plataformas" con las plataformas que han sido pasadas con la acción.

```
22  ReactDOM.render(
23    <Provider store={store}>
24        <BrowserRouter>
25            <div>
26                <Switch>
27                    <Route path="/seleccion-metricas" component={SeleccionMetricas} />
28                    <Route path="/seleccion-atributos" component={SeleccionAtributos} />
29                    <Route path="/seleccion-caracteristicas" component={SeleccionCaracteristicas} />
30                    <Route path="/seleccion-restricciones" component={SeleccionRestricciones} />
31                    <Route path="/seleccion-contexto" component={SeleccionContexto} />
32                    <Route path="/seleccion-plataforma" component={SeleccionPlataforma} />
33                    <Route path="/seleccion-rol" component={SeleccionRol} />
34                    <Route path="/" component={Index} />
35                </Switch>
36            </div>
37        </BrowserRouter>
38    </Provider>,
39      document.querySelector('.container')
40  );
```

Figura 29. Archivo index.js de la aplicación.

Por último, quiero explicar el archivo index.js, el cual se encarga de añadir los componentes a nuestra página HTML. Mediante los elementos <Route> podemos

decirle a React.js que componentes queramos que se vean en cada momento dependiendo de la ruta en la que se encuentra el usuario.

Estos serían los conceptos más importantes que habría que entender. Para comprender como funciona todo en conjunto es necesario ir a la documentación oficial de las tecnologías.

5. Caso de Estudio: Evaluación de Microsoft Azure, Google App Engine y Amazon EC2

Hemos desarrollado un caso de estudio para mostrar como el modelo de calidad y sistema de recomendación puede ser utilizado para hacer una selección específica de plataformas cloud. Los requisitos del caso de estudio son los siguientes:

Un usuario llamado Carlos está interesado en contratar una plataforma cloud para la web de su tienda de muebles. Ha mirado información en internet, pero no le ha quedado claro cuál le convendría entre Microsoft Azure, Google Cloud Platform y Amazon Web Services. Carlos entonces ha decidido utilizar nuestro sistema de recomendación para apoyar su decisión.

Primero Carlos selecciona las tres plataformas que desea comparar: *Microsoft Azure (MA), Google Cloud Platform (GCP), Amazon Web Services (AWS).* Después selecciona su rol como *cliente*. Carlos introduce el *contexto de la evaluación*, el cual es: “La empresa de venta de muebles XYZ”.

En cuanto a las características le interesan la *adecuación funcional*, la *eficiencia de desempeño*, la *fiabilidad* y la *seguridad*. De entre todas las subcaracterísticas, las que más le interesan son corrección funcional (*adecuación funcional*), tiempo de respuesta del servicio (*eficiencia de desempeño*), utilización de recursos (*eficiencia de desempeño*), disponibilidad *(fiabilidad)* y confidencialidad *(seguridad)*.

Los atributos y métricas seleccionadas por Carlos para cada subcaracterística son las siguiente:

- Corrección funcional:
    - Capacidad computacional de la plataforma cloud
        - Capacidad computacional de la plataforma

- Tiempo de respuesta del servicio:
    - Tiempo de respuesta del servicio
        - Tiempo promedio del tiempo de respuesta
        - Tiempo de respuesta máximo

- Utilización de recursos:
    - Coste en curso del servicio
        - Coste de almacenamiento

- Disponibilidad:
    - Disponibilidad
        - Porcentaje del tiempo en línea

- Confidencialidad:
    - Protección

- Costo de la seguridad

Después Carlos introduce los *umbrales para cada métrica*, de manera que después en la evaluación el sistema podrá comparar el resultado con los umbrales para ver si se cumplen sus necesidades.

Los umbrales introducidos son los siguientes:

- Capacidad computacional de la plataforma:
  - mayor a 3 GHz – mejor de lo esperado
  - entre 3 GHz y 2,5 GHz – rango objetivo
  - entre 2,4 GHz y 2,2 GHz– mínimamente aceptable
  - menor a 2,2 GHz – inaceptable

- Tiempo promedio del tiempo de respuesta:
  - menor a 60 ms – mejor de lo esperado
  - entre 60 ms y 90 ms – rango objetivo
  - entre 91 ms y 110 ms – mínimamente aceptable
  - mayor a 110 ms – inaceptable

- Tiempo de respuesta máximo:
  - menor a 110 ms – mejor de lo esperado
  - entre 110 ms y 120 ms – rango objetivo
  - entre 121 ms y 125 ms – mínimamente aceptable
  - mayor a 125 ms – inaceptable

- Coste de almacenamiento:
  - menor a 0,001€ / GB al mes – mejor de lo esperado
  - entre 0,001€ / GB al mes y 0,003€ / GB al mes – rango objetivo
  - entre 0,004€ / GB al mes y 0,005€ / GB al mes – mínimamente aceptable
  - mayor a 0,005€ € / GB al mes – inaceptable

- Porcentaje del tiempo en línea:
  - mayor a 99,99% - mejor de lo esperado
  - entre 99,99% y 99,95% - rango objetivo
  - entre 99,94% y 99,90% - mínimamente aceptable
  - menor a 99,94% - inaceptable

- Costo de la seguridad:
  - menor a 10€ / mes – mejor de lo esperado
  - entre 10€ / mes y 15€ / mes – rango objetivo
  - entre 16€ / mes y 20€ / mes – mínimamente aceptable
  - mayor a 20€ / mes – inaceptable

El siguiente paso sería realizar los cálculos de todas las métricas. En nuestro caso, los valores para los cálculos de las métricas los hemos sacado de las páginas oficiales de las plataformas [76] [77] [78] y de páginas para hacer pruebas de conexión con las plataformas [79] [80] [81].

En la tabla 8 podemos ver los resultados obtenidos para cada métrica en todas las plataformas (MA = Microsoft Azure, GCP = Google Cloud Platform, AWS = Amazon Web Services).

Tabla 8. Valores para las métricas.

	MA	AWS	GCP
Capacidad computacional de la plataforma	2,4 GHz	2,5 GHz	2,6 GHz
Tiempo promedio del tiempo de respuesta	102 ms	73 ms	115 ms
Tiempo de respuesta máximo	153 ms	82 ms	152 ms
Coste de almacenamiento	0,0025 USD/GB	0,004 USD/GB	0,04 USD/GB
Porcentaje del tiempo en línea	99,90%	99,99%	99,95%
Costo de la seguridad	15 USD por nodo	0,29 USD por GB	gratis

El cálculo se realiza de la siguiente manera. Si el resultado obtenido de una métrica entra en el rango esperado o mejor de lo esperado obtendría 100 puntos. En caso de que se no lo cumpla, calcularíamos la diferencia en forma de porcentaje, pongamos un ejemplo de una métrica de este caso de estudio:

El tiempo promedio de respuesta deseado es entre 60 ms y 90 ms o mejor. Microsoft Azure tiene un valor de 102 ms, que son 12 ms más de lo deseado.

La diferencia en un porcentaje sería: 13,34%

La puntuación que obtendría Microsoft Azure sería: 86,66 puntos (100 – 13,34), entraría en el umbral de *mínimamente aceptable*.

La puntuación para Amazon Web Services: 100 puntos, entra en el *rango deseado*.

La puntuación para Google Cloud Platform: 72,22 puntos, entra en el umbral de *inaceptable*.

Después, si un atributo tiene más de una métrica seleccionada, hacemos una media simple, es decir, le damos la misma importancia a todas las métricas. Después haríamos lo mismo con las subcaracterísticas. Si una subcaracterística tiene más de un atributo se realiza la media simple y así hasta llegar a las características.

En nuestro caso de estudio estos son los resultados obtenidos (para el coste de seguridad suponemos que Carlos tendrá un tráfico equivalente a máximo 100 GB/mes, usando un nodo):

Tabla 9. Resultados de las métricas.

	MA	AWS	GCP
Capacidad computacional de la plataforma	96	100	100
Tiempo promedio del tiempo de respuesta	86,66	100	72,22
Tiempo de respuesta máximo	72,5	100	73,33
Coste de almacenamiento	100	66,66	0
Porcentaje del tiempo en línea	99,94	100	100
Costo de la seguridad	100	6,66	100

Ahora tenemos que ir escalando hasta obtener la puntuación total, por lo tanto los cálculos son los siguientes:

Microsoft Azure: 96,43 puntos

Adecuación funcional: 96 puntos – Calidad excelente

- Corrección funcional: 96 puntos
    - Capacidad computacional de la plataforma cloud: 96 puntos

Eficiencia de desempeño: 89,79 puntos – Calidad muy buena

- Tiempo de respuesta del servicio: 79,58 puntos
    - Tiempo de respuesta del servicio: 79,58 puntos
- Utilización de recursos: 100 puntos
    - Coste en curso del servicio: 100 puntos

Fiabilidad: 99,94 puntos – Calidad excelente

- Disponibilidad: 99,94
    - Disponibilidad: 99,94 puntos

Seguridad: 100 puntos – Calidad excelente

- Confidencialidad: 100 puntos
    - Protección: 100 puntos

Amazon Web Services: 72,49 puntos

Adecuación funcional: 100 puntos – Calidad excelente

- Corrección funcional: 100 puntos
    - Capacidad computacional de la plataforma cloud: 100 puntos

Eficiencia de desempeño: 83,33 puntos – Calidad muy buena

- Tiempo de respuesta del servicio: 100 puntos
    - Tiempo de respuesta del servicio: 100 puntos
- Utilización de recursos: 66,66 puntos
    - Coste en curso del servicio: 66,66 puntos

Fiabilidad: 100 puntos – Calidad excelente

- Disponibilidad: 100
    - Disponibilidad: 100 puntos

Seguridad: 6,66 puntos – Calidad deficiente

- Confidencialidad: 6,66 puntos
    - Protección: 6,66 puntos

Google Cloud Platform: 84,17 puntos

Adecuación funcional: 100 puntos – Calidad excelente

- Corrección funcional: 100 puntos
    - Capacidad computacional de la plataforma cloud: 100 puntos

Eficiencia de desempeño: 36,38 puntos – Calidad insuficiente

- Tiempo de respuesta del servicio: 72,77 puntos
    - Tiempo de respuesta del servicio: 72,77 puntos
- Utilización de recursos: 0 puntos
    - Coste en curso del servicio: 0 puntos

Fiabilidad: 100 puntos – Calidad excelente

- Disponibilidad: 100
    - Disponibilidad: 100 puntos

Seguridad: 100 puntos – Calidad excelente

- Confidencialidad: 100 puntos
    - Protección: 100 puntos

Por último, con los resultados ya obtenidos, calculamos que plataforma ha obtenido mayor puntuación y generamos el informe final, que le muestra a Carlos que plataforma es la que mejor se ajusta a sus necesidades, junto a todos los resultados para cada característica, subcaracterística, atributo y métricas.

En el caso de estudio, la plataforma Microsoft Azure es la que más se ajusta a las necesidades de Carlos, a pesar de que solo una de las características entra el rango esperado, las demás obtienen unos valores muy cercanos a sus necesidades.

6. Conclusiones

Actualmente se observa un aumento en el uso de los servicios y plataformas cloud. Las plataformas cloud son entornos que están en constante evolución y se necesitan nuevos métodos que guíen a los usuarios en la selección correcta de proveedores y plataformas cloud que más se adecuen a sus necesidades. Para afrontar la toma de decisiones en escenarios donde intervienen gran número de criterios o alternativas de selección, en este trabajo, se ha propuesto un modelo de calidad que descompone la calidad de las plataformas cloud en subcaracterísticas, atributos y métricas, y un sistema de recomendación que utiliza el modelo de calidad propuesto para apoyar a usuarios en la selección de plataformas cloud.

El principal beneficio del sistema es el que proporciona mecanismos cuantitativos (métricas) que permite a los usuarios discernir entre estos proveedores cloud para así obtener una solución que satisfaga, en la mejor medida posible, los distintos requisitos del usuario. Las métricas se han seleccionado en base a una revisión sistemática de la literatura.

Se ha desarrollado un prototipo que ha sido probado en la selección de algunas plataformas cloud. Sin embargo, se necesitan más casos de estudio para evaluar y refinar el modelo de calidad y sistema de recomendación propuesto. Un aspecto importante, que, si bien hemos avanzado en parte, que es motivo de futura investigación, reside en establecer distintos pesos para las características. En estos momentos el sistema establece la misma importancia para todas las características y este no tendría por qué ser el caso real. Otra posible mejora sería el estudio de métodos de selección multicriterio como el Proceso de Análisis Jerárquico (AHP) para mejorar el algoritmo de selección de plataformas cloud empleado actualmente por el sistema de recomendación desarrollado.

Otro aspecto que podría mejorarse en la aplicación sería la introducción de gráficas al mostrar los resultados de la evaluación. Cómo indicamos anteriormente, esto mejoraría la usabilidad de la herramienta y ayudaría al usuario a entender los resultados.

7. Referencias

[1] http://iso25000.com/index.php/normas-iso-25000/iso-25010

[2] Navas Rosales, R. *Modelo de Calidad para Servicios Cloud,* Tesis de Máster, Máster Universitario en Ingeniería y Tecnología de Sistemas Software, Universitat Politècnica de València (2016).

[3] Alor Hernández, G.; Dávila Nicanor, L.; Fernández Domínguez, R. *Análisis y evaluación de marcos de trabajo para el desarrollo de aplicaciones y servicios en la nube PAAS*. División de Estudios de Posgrado e Investigación Instituto Tecnológico de Orizaba (2014).

[4] Viswanathan, B. *Understanding the different roles in a cloud computing setup.* CloudTweaks (2012). http://cloudtweaks.com/2012/04/understanding-the-different-roles-in-a-cloud-computing-setup/

[5] Böhm, M.; Krcmar, H.; Leimeister, S.; Riedl, C. "The Business Perspective of Cloud Computing: Actors, Roles and Value Networks" (2010). ECIS 2010 Proceedings. 56. http://aisel.aisnet.org/ecis2010/56

[6] Mendoza Ricci, O. *Cloud Computing*. *http://www.monografias.com/trabajos-pdf5/cloud-computing/cloud-computing.shtml*

[7] Alcocer, A. Cloud Computing, características de las aplicaciones en Cloud. http://www.societic.com/2010/03/cloud-computing-caracteristicas-de-las-aplicaciones-en-cloud/

[8] Mestas, J. Características Esenciales Cloud Computing. http://geekswithblogs.net/gotchas/archive/2011/10/03/caracteristicas-esenciales-cloud-computing.aspx

[9] IBM Cloud. https://www.ibm.com/cloud-computing/es-es/learn-more/iaas-paas-saas/

[10] Intel Security. ¿Cielos despejados? El nivel de adopción de la nube. https://mcafee.app.box.com/v/cielosdespejados

[11] Asociación Española para la Calidad (AEC). Modelos de calidad. https://www.aec.es/web/guest/centro-conocimiento/modelos-de-calidad

[12] ISO/IEC 25010. http://iso25000.com/index.php/normas-iso-25000/iso-25010

[13] División consultoría de EvaluandoCloud.com. Actores de la nube o Cloud Computing. http://evaluandocloud.com/actores-la-nube-cloud-computing/

[14] Amazon Web Services. Wikipedia. https://es.wikipedia.org/wiki/Amazon_Web_Services

[15] Amazon Web Services. https://aws.amazon.com

[16] Datamation. Google Cloud Platform. https://www.datamation.com/cloud-computing/google-cloud-platform.html

[17] Google Cloud Platform. https://cloud.google.com/

[18] EvaluandoCloud.com. Estudio comparativo de plataformas Cloud Computing. http://evaluandocloud.com/estudio-comparativos-de-plataformas-cloud-computing/

[19] iso25000.com. ISO/IEC 25040. http://iso25000.com/index.php/normas-iso-25000/iso-25040

[20] ISO/IEC 17788:2014. Information technology -- Cloud computing -- Overview and vocabulary," 2014)

[21] Grance, T.; Mell, P. The NIST Definition of Cloud Computing. *Nist Special Publication*, *145*, 7 (2011).

[22] Varela, M., Technical, V. T. T. Challenges of QoE Management for Cloud Applications. *IEEE Communications Magazine*, *50*(April), 28–36 (2012).

[23] Buyya, R.; Kumar Garg, S.; Versteeg, S. A framework for ranking of cloud computing services. http://www.sciencedirect.com/science/article/pii/S0167739X12001422

[24] Nadanam, P.; Rajmohan, R. QoS evaluation for web services in cloud computing (2012). http://ieeexplore.ieee.org/document/6395991/

[25] Chana, I.; Singh, S. Q-aware: Quality of service based cloud resource provisioning (2015). http://www.sciencedirect.com/science/article/pii/S0045790615000282

[26] Vedam, V.; Vemulapati, J. Demystifying Cloud Benchmarking Paradigm - An in Depth View (2012). http://ieeexplore.ieee.org/document/6340191/

[27] Da Costa, G.; Guérout, T.; Medjiah, S.; Monteil, T. Quality of service modeling for green scheduling in Clouds (2014). http://www.sciencedirect.com/science/article/pii/S221053791400047X

[28] Ghobaei Arani, M.; Gholami, A. A trust model for resource selection in cloud computing environment (2015). http://ieeexplore.ieee.org/document/7436036/

[29] Hsiao, H.; Wen, Z. QoE-driven performance analysis of cloud gaming services (2014). http://ieeexplore.ieee.org/document/6958835/

[30] Belqasmi, F.; Glitho, R.; Hassam, M.; Kara, N. Virtualized infrastructure for video game applications in cloud environments. http://dl.acm.org/citation.cfm?doid=2642668.2642679

[31] Bruneo, D. A Stochastic Model to Investigate Data Center Performance and QoS in IaaS Cloud Computing Systems (2014). http://doi.org/10.1109/TPDS.2013.67

[32] Liu, G.; Wu, X.; Xu, J. A QoS-constrained scheduling for access requests in cloud storage (2015). http://ieeexplore.ieee.org/xpls/icp.jsp?arnumber=7334102

[33] Dou, W.; Liu, M.; Yu, S.; Zhang, Z. A clusterized firewall framework for cloud computing (2014). http://doi.org/10.1109/ICC.2014.6883911

[34] Baranwal, G.; Prakash Vidyarthi, D. A framework for selection of best cloud service provider using ranked voting method (2014). http://ieeexplore.ieee.org/document/6779430/

[35] Ding, C.; Karim, R.; Miri, A. End-to-End Performance Prediction for Selecting Cloud Services Solutions (2015). http://doi.org/10.1109/SOSE.2015.11

[36] Deep Kaur, P.; Chana, I. A resource elasticity framework for QoS-aware execution of cloud applications (2014). http://www.sciencedirect.com/science/article/pii/S0167739X14000430

[37] Chua, F.; Chan, G.; Mahmood Khan, H. An adaptive monitoring framework for ensuring accountability and quality of services in cloud

computing (2016). http://ieeexplore.ieee.org/xpls/icp.jsp?arnumber=7427071

[38] Machado, J. C.; Sousa, F. R. C. Towards Elastic Multi-Tenant Database Replication with Quality of Service (2012). http://doi.org/10.1109/UCC.2012.36

[39] Chen, X.; Xiong, K. Ensuring Cloud Service Guarantees via Service Level Agreement (SLA)-Based Resource Allocation (2015). http://doi.org/10.1109/ICDCSW.2015.18

[40] Cervino, J.; Mozo, A.; Rodriguez, P.; Salvachua, J.; Trajkovska, I. Testing a Cloud Provider Network for Hybrid P2P and Cloud Streaming Architectures (2011). http://ieeexplore.ieee.org/document/6008730/

[41] Al-Jawad, A.; Gemikonakli, O.; Shah, P.; Trestian, R. BaProbSDN: A probabilistic-based QoS routing mechanism for Software Defined Networks. http://ieeexplore.ieee.org/document/7116128/

[42] Brahmi, Z.; Bousselmi, K.; Mohsen Gammoudi, M. QoS-Aware Scheduling of Workflows in Cloud Computing Environments (2016). http://ieeexplore.ieee.org/document/7474163/

[43] Abrahao, S.; Cedillo, P.; Insfran, E.; Jimenez-Gomez, J. Towards a Monitoring Middleware for Cloud Services (2015). http://ieeexplore.ieee.org/xpls/icp.jsp?arnumber=7207386

[44] Feng, J.; Kong, L. A Fuzzy Multi-objective Genetic Algorithm for QoS-based Cloud Service Composition (2015). http://ieeexplore.ieee.org/document/7429378/

[45] Fazeli, M.; Mohssen Ghafari, S.; Rikhtechi, L.; Patooghy, A. Bee-MMT: A load balancing method for power consumption management in cloud computing (2013). http://ieeexplore.ieee.org/xpls/icp.jsp?arnumber=6612165

[46] Dong Kim, S.; Wan Cheun, D.; Woo Lee, J.; Yoo Lee, J. A Quality Model for Evaluating Software-as-a-Service in Cloud Computing (2009). http://ieeexplore.ieee.org/document/5381749/

[47] Rizvi, S., Ryoo, J., Kissell, J., & Aiken, B. (2015). A Stakeholder-oriented Assessment Index for Cloud Security Auditing. In Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication (pp. 55:1–55:7). New York, NY, USA: ACM. doi:10.1145/2701126.2701226. [ACM]

[48] Entezari-Maleki, R.; Movaghar, A.; Roohitavaf, M. Availability Modeling and Evaluation of Cloud Virtual Data Centers (2013). http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6808257

[49] Cristobo, L.; Ibarrola, E.; Taboada, I.; Saiz, E. A cloud platform for QoE evaluation: QoXcloud (2014). http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6858471

[50] Jiang, N.; Li, W.; Wang, Z.; Zhou, P. Quality Model of Cloud Service (2015). http://ieeexplore.ieee.org/xpls/icp.jsp?arnumber=7336367

[51] Beshley, M.; Klymash, M.; Maksymyuk, T.; Strykhalyuk, B. Research and development the methods of quality of service provision in Mobile Cloud systems (2014). http://doi.org/10.1109/BlackSeaCom.2014.6849030

[52] Dubalski, B.; Junior, J. C.; Ledzinski, D.; Pedersen, J. M.; Patel, A.; Riaz, M. T. Assessing Measurements of QoS for Global Cloud Computing

Services. In Dependable, Autonomic and Secure Computing (2011). http://doi.org/10.1109/DASC.2011.120

[53] Binu, A.; Chandrasekaran, K.; Joy, N. A study on energy efficient cloud computing (2015). http://doi.org/10.1109/ICCIC.2015.7435661

[54] Brandic, I.; Jaarevic, J.; Mastelic, T. CPU Performance Coefficient (CPU-PC): A Novel Performance Metric Based on Real-Time CPU Resource Provisioning in Time-Shared Cloud Environments (2014). http://doi.org/10.1109/CloudCom.2014.13

[55] Gagnaire, M.; Zant, B. E. Towards a unified customer aware figure of merit for CSP selection (2015). http://doi.org/10.1186/s13677-015-0049-1

[56] Brohman, K.; Martin, P.; Zheng, X. Cloud Service Negotiation: A Research Roadmap (2013). http://doi.org/10.1109/SCC.2013.93

[57] Brohman, K.; Martin, P.; Xu, L. D.; Zheng, X. CLOUDQUAL: A Quality Model for Cloud Services (2014). http://doi.org/10.1109/TII.2014.2306329

[58] Huang, Y.; Luo, X.; Xia, Y.; Zhou, M.; Zhu, Q.; Li, J.; Stochastic Modeling and Quality Evaluation of Infrastructure-as-a-Service Clouds (2015). http://doi.org/10.1109/TASE.2013.2276477

[59] Ravindran, K. Self-Assessment and Reconfiguration Methods for Autonomous Cloud-based Network Systems (2013). http://doi.org/10.1109/DS-RT.2013.37

[60] Abdeladim, A.; Baina, S.; Baina, K. Elasticity and scalability centric quality model for the cloud (2014). http://doi.org/10.1109/CIST.2014.7016607

[61] Manuel, P. A trust model of cloud computing based on Quality of Service (2015). http://doi.org/10.1007/s10479-013-1380-x

[62] Ghosh, R.; Longo, F.; Naik, V. K.; Trivedi, K. S. Quantifying Resiliency of IaaS Cloud (2010). http://doi.org/10.1109/SRDS.2010.49

[63] Swamynathan, S.; Viji Rajendran, V. Hybrid model for dynamic evaluation of trust in cloud services (2015). http://doi.org/10.1007/s11276-015-1069-y

[64] Choi, C.-R.; Jeong, H. Y. Quality evaluation and best service choice for cloud computing based on user preference and weights of attributes using the analytic network process (2014). http://doi.org/10.1007/s10660-014-9156-1

[65] Rodríguez M., Piattini M. Entorno para la Evaluación y Certificación de la Calidad del Producto Software, XIX Jornadas de Ingeniería del Software y Bases de Datos, Cádiz, 2014, pp. 163-176.

[66] iso.org. ISO/IEC 19086-1. https://www.iso.org/standard/67545.html (2016)

[67] Han, S.; Huh, E.; Mehedi Hassan, M.; Yoon, C. Efficient Service Recommendation System for Cloud Computing Market (2009).

[68] Buyya, R.; Qu, C. A Cloud Trust Evaluation System Using Hierarchical Fuzzy Inference System for Service Selection. Adv Inf Netw Appl AINA 2014 IEEE 28th Int Conf On. 2014 May 13;850-7.

[69] React.js. https://reactjs.org

[70] Redux.js. https://redux.js.org

[71] Bootstrap. https://getbootstrap.com

[72] JavaScript Expression Evaluator. https://silentmatt.com/javascript-expression-evaluator/

[73] package.json. https://docs.npmjs.com/files/package.json
[74] ISO/IEC 9126-1:2001. https://www.iso.org/standard/22749.html
[75] ISO/IEC 14598-1:1999. https://www.iso.org/standard/24902.html