Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN

Nombre del Centro

Trabajo Fin de Grado

CLOUD COMPUTING: INICIACIÓN A LA CREACIÓN DE APLICACIONES EN LA NUBE MEDIANTE EL PARADIGMA PLATFORM AS A SERVICE (PAAS)

Alumno: Francisco Nieves Pastor

Tutor: Prof. D. Víctor Manuel Rivas Santos

Dpto: Ingeniería Informática

Septiembre, 2020

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don VÍCTOR MANUEL RIVAS SANTOS , tutor del Proyecto Fin de Carrera titulado: CLOUD COMPUTING: INICIACIÓN A LA CREACIÓN DE APLICACIONES EN LA NUBE MEDIANTE EL PARADIGMA PLATFORM AS A SERVICE (PAAS), que presenta FRANCISCO NIEVES PASTOR, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, SEPTIEMBRE de 2020

El alumno:

FRANCISCO NIEVES PASTOR

El tutor:

Firmado por RIVAS SANTOS VICTOR MANUEL - 29798183G, profesor Titular de la Universidad de Jaén, el día 25/09/2020 con un certificado emitido por AC FNMT Usuarios

VÍCTOR MANUEL RIVAS SANTOS

Índice

Índice de Ilustraciones

Índice de tablas

1. INTRODUCCIÓN

En este punto, vamos a introducir la motivación que ha llevado a realizar este proyecto, los objetivos que se perseguirán y los pasos que se van a seguir.

1.1. Aspectos generales y motivación

Al realizar este proyecto encontramos una gran motivación, ya que vamos a tratar un tema que se encuentra en pleno apogeo dentro del mundo de la informática, y de la tecnología en general. Vamos a tratar el tema del Cloud Computing, una herramienta que en la actualidad se encuentra presente prácticamente en todos los entornos tecnológicos.

A pesar de que el Cloud Computing está presente en el día a día de cualquier usuario que pueda tener un acceso a un terminal móvil, un ordenador, etc. se trata de una herramienta muy ambigua y que por lo general, no tiene una definición exacta y que esté extendida.

Es por ello, que en este proyecto, se intentará dar una definición concreta de esta tecnología, y todo lo que envuelve, abarcando todo lo más relativo a ella.

Además, no se quedará únicamente en un aspecto teórico, sino que se usará esta tecnología pasando a un aspecto práctico, en el que se deje claro las dudas teóricas que puedan surgir.

Antes de hablar en sí de lo que es la nube, y acometer este proyecto, es necesario distinguir entre los tres tipos de implementación presentes en la informática.

Nube: cuando se implementa una aplicación en la nube, toda esta se encuentra en la nube, haya sido implementada en ella, o posteriormente transferida a ella.

Solución híbrida: esta solución permite conectar los recursos existentes en la nube con el exterior de esta.

Implementación local: de esta forma se trabaja en el propio dispositivo local, y no se usa la tecnología de la nube.

1.2. Objetivos del proyecto

Encontramos dos objetivos principales en este proyecto, los cuales han sido mencionados anteriormente.

En primer lugar, un objetivo principal es proporcionar una definición concreta de lo que es la tecnología del Cloud Computing, abarcando todos los aspectos necesarios para ello.

En segundo lugar, el otro objetivo es crear una aplicación en el paradigma de Plataforma como Servicio (PaaS) del Cloud Computing **1**. A través de esta aplicación, podrán ser abordados todos los aspectos que la teoría no cubre, dando un enfoque distinto y que complete a esta.

1. La aplicación creada está disponible en la dirección: http://twsent.azurewebsites.net/

2. CLOUD COMPUTING: DEFINICIÓN. EVOLUCIÓN Y ACTUALIDAD

Este capítulo se centrará en dar forma al concepto de Cloud Computing, abordándolo desde distintos puntos.

2.1. ¿Qué es el Cloud Computing?

Wikipedia lo define de la siguiente forma: ”*La **computación en la nube** (del inglés cloud computing), conocida también como **servicios en la nube**, **informática en la nube**, **nube de cómputo**, **nube de conceptos** o simplemente «**la nube**» ”*.

Como se puede observar en este fragmento, hablar de nube (*cloud*) o hablar de cómputo en la nube (*Cloud Computing*) es lo mismo, por lo tanto, a lo largo de este documento, se mencionarán ambos, haciendo referencia a lo mismo.

Una vez aclarado esto, comenzamos a definir qué es el Cloud Computing.

Nos encontramos ante una de las tareas más complicadas, definir con exactitud y claridad qué es el fenómeno del que todos hablan y, en realidad muy pocos saben qué es con certeza.

La nube no es algo físico, este término de nube en el mundo de la Informática y la Tecnología hace referencia a una red mundial de servidores remotos, cada cual con una determinada función, que están interconectados para almacenar y administrar datos, ejecutar aplicaciones u ofrecer servicios tales como streaming de vídeos, correo web, etcétera. Únicamente, será necesario una conexión a Internet para poder acceder a estos datos/servicios, en lugar de hacerlo desde un equipo local, con la ventaja de que estos datos/servicios estarán disponibles en cualquier lugar.

Es decir, cuando hablamos de cloud computing hacemos referencia a un conjunto de servidores, generalmente, propiedad de empresas, que, en el negocio de cloud computing, se denominan proveedores de servicios de la

nube, ya que son los encargardos de aprovisionarnos con estos servidores, para nuestro propio uso. Estos proveedores, serán estudiados en profundidad más adelante.

Ilustración 0. Representación de la nube.

2.2. Características de la nube

Aun así, esta definición puede dejarnos algunos aspectos con cierta duda, pero para aclararlos, tenemos estos parámetros que debe tener una tecnología para ser considerada como nube:

2.2.1. Autoservicio bajo demanda

En primer lugar, servicio bajo demanda es la capacidad de satisfacer una necesidad cuando esta surge, realizar una petición y recibir acceso al servicio requerido en el momento.

Esto unido a la capacidad de realizar por uno mismo y sin un intermediario que conceda el acceso o autorización, hace esta característica fundamental y muy atractiva para el usuario, dado el dinamismo y velocidad de acceso que ofrece a la tecnología que la presenta. Aunque es necesario mencionar que la implementación de esta funcionalidad puede llegar a ser muy complicada.

2.2.2. Amplio Acceso a la Red y a los Servicios

Al hablar de nube, es normal que a la cabeza se nos venga que tenemos que estar conectados a una red para poder disfrutar de sus servicios. Estos servicios, alojados en la red, además deberían ser accedidos fácilmente, es decir, el acceso a la red debe ser un acceso de calidad, que permita una conexión estable y eficaz, para poder aprovechar al máximo las ventajas que la nube nos ofrece.

Esta conexión normalmente será una conexión a Internet, y pese a que Internet está creciendo mucho en ancho de banda, sigue siendo relativamente lento si lo comparamos con las conexiones de las redes de área local (LANs). Por ello, esta conexión, como ya hemos mencionado debe ser eficaz y de calidad, pero no necesariamente debe exigirse un gran ancho de banda.

El limitado ancho de banda nos trae a otro punto de esta característica: los servicios de la nube no deberían exigir ningún cliente o en su defecto, un cliente ligero, ya que descargar un cliente pesado con una conexión débil puede tardar mucho tiempo, y si la aplicación cliente exige mucha comunicación entre el sistema cliente y los servicios, puede dar lugar a que los usuarios puedan experimentar problemas de latencia.

Esto nos lleva al último punto de esta característica: los servicios de la nube deberían ser accesibles por una gran variedad de dispositivos clientes. Esto significa básicamente que los usuarios deberían ser capaces de acceder desde tablets, smartphones y otros tipos de dispositivos, no solo desde ordenadores. Además, deberían ser servicios multiplataforma (Windows, Mac, iOS, Android, …), por lo tanto, un servicio sin cliente será mucho más fácil de construir, ya que no habrá que hacer una implementación para cada dispositivo y plataforma.

2.2.3. Pool de recursos dinámicos

Esta característica es la que permite la gran eficiencia de la nube, además permite llevar un registro de los costes y dotar de una gran flexibilidad por parte

del proveedor. Se basa en el hecho de que los clientes no tienen una constante necesidad de todos los recursos disponibles para ellos, por lo que estos recursos que no están siendo aprovechados por ese consumidor, están disponibles para ser usados por otro, aumentando la eficiencia y evitando desaprovechar recursos.

Esto es posible gracias a la virtualización, ya que permite a los proveedores incrementar la densidad de sus sistemas. Esta tecnología más adelante será descrita.

2.2.4. Escalabilidad y elasticidad

Cuando un cliente de la nube está cerca de sobrepasar los recursos que le han sido asignados, la nube debe ser capaz de buscar una solución. Esta solución es implementada mediante la escalabilidad y una rápida elasticidad que permite expandir con facilidad su capacidad de servicio, consiguiendo satisfacer la demanda del usuario. La clave se encuentra en que todos los recursos estén disponibles, pero no en uso hasta que no son necesitados, lo que permite medir el consumo (como se explicará en el punto 2.2.5).

Cuando el uso de un recurso necesita más, un disparador se activa y automáticamente comienza el proceso de expansión, asignando los recursos que se solicitan. Y cuando el uso del recurso ha finalizado, esta expansión comienza a desvanecerse para asegurar que no se desperdician los recursos, y que puedan ser usados por otros clientes. Esto es posible gracias al uso de la automatización y la orquestación.

Esta característica de la nube, está muy ligada a la anterior (Pool de recursos dinámicos), debido a que para efectuar un proceso de escalado es necesario que haya un conjunto de recursos disponibles, es por ello, que este característica es muy dependiente de que existan recursos disponibles para ser usados, o viceversa, un lugar donde depositar aquellos recursos que debido a la reducción del uso, no son necesarios en ese momento.

2.2.5. Medición de los servicios

Esta característica está muy estrechamente relacionada con las dos anteriores. Como sabemos los servicios en la nube son dinámicos, es decir, pueden crecer o decrecer según el uso que necesitemos de ellos, y la nube debe ser capaz de medir el uso de estos servicios, con el objetivo de que entiendas exactamente qué servicios has usado y sobre todo, que pagues solo por lo que has usado, siendo esto un gran atractivo de la nube.

2.3. Modelos de despliegue de la nube

La forma en la que se proporciona el servicio o la forma en la que se accede a la nube varía, y depende de cada usuario u organización. Estos tienen sus propios requisitos, y en algunos casos necesitarán más control o prioridades en un aspecto determinado. Pues bien, es por ello, que existen varios tipos de despliegue de la nube, cada uno con sus ventajas y desventajas.

2.3.1. Nube pública

Cuando hablamos de nube, por lo general lo asociamos a la nube pública, el modelo más extendido y al que el usuario medio tiene acceso.

Se trata del modelo en el cual todos los servicios están proporcionados por un proveedor de servicios externo, el cual es el responsable de la gestión y la administración de los sistemas que los proporcionan. El cliente es responsable únicamente del software instalado para acceder a ella.

Básicamente, el funcionamiento es el siguiente: el cliente contrata servidores de forma instantánea a través generalmente, de un servicio web centralizado. Estos servidores, se encuentran alojados en datacenters pertenecientes al proveedor de servicios, en los cuales se comparten recursos entre todos los clientes. Dependiendo del proveedor, en algunos casos el uso es gratis, pero lo por lo general, se pagará por el uso de estos servidores.

Otra característica de la nube pública es que el acceso a esta es igual para todos los usuarios y es accesible desde cualquier dispositivo con conexión a la red.

Cuando hablamos de servicios proporcionados, no hablamos únicamente de servidores (máquinas físicas), sino que también hablamos de bases de datos, balanceadores, etcétera.

De esta forma, la nube pública te permitirá aumentar y reducir el número de servidores en tiempo real, dependiendo tus las necesidades.

La nube pública es perfecta para empresas que no pueden permitirse un desembolso inicial muy grande, startups o servicios con una carga que varía mucho en el tiempo.

Ventajas:

- Costos inferiores.
- Sin mantenimiento.
- Escalabilidad casi ilimitada.
- Gran confiabilidad.

2.3.2. Nube privada

La nube privada es una gran opción para empresas que necesiten una gran protección de los datos. En este caso, el hardware, la red y las aplicaciones son dedicadas para una sola empresa. El hardware puede estar en la misma empresa, o en el datacenter del proveedor de servicios, pero en este caso, en el datacenter colocas tus propios servidores.

Aquí el cliente asume más responsabilidades, ya que es el responsable de la gestión y administración de los sistemas que proporcionan los servicios. Es decir, la nube privada necesita más mantenimiento, y además requiere de un coste inicial elevado.

Como se trata de una nube proporciona escalabilidad, pero a nivel inferior que la nube pública, ya que el número de servidores que tienes es limitado. Pero en su favor juega que puedes configurar el entorno según las necesidades específicas que tengas, posees más control. Además, por lo general, el nivel de seguridad es mucho mayor.

La nube privada se presenta muy apetecible para empresas que posean su propio datacenter o empresas que necesiten una gran personalización y control.

Ventajas:

- Más flexibilidad.
- Mejor seguridad.
- Mayor escalabilidad.

2.3.3. Nube híbrida

Tanto la nube privada como la nube pública nos proporcionan unas ventajas, pero el hecho de usar una u otra, deja fuera características que nos gustaría que pudiesen ser complementarias. Para ello, surge la nube híbrida, que no es otra cosa, que una combinación entre la nube pública y la privada, consiguiendo combinar la mayoría de los beneficios de ambas.

En una nube híbrida los datos que se almacenan pueden moverse entre nubes privadas y públicas, obteniendo más flexibilidad y opciones de implementación. Es por ello, que las nubes no están mezcladas entre sí, cada una de las nubes que componen la nube híbrida son nubes independientes, pero que están conectadas entre ellas.

En este tipo de nubes, se suele usar la nube privada para operaciones confidenciales, y la nube pública para tratar un gran volumen de datos que no necesita tanta seguridad. Además, la nube pública se puede usar, cuando la nube privada esté llegando a su límite, aprovechando los recursos que ofrece esta.

Ventajas:

- Control.
- Flexibilidad.
- Rentabilidad.
- Facilidad de migración.

2.3.4. Nube comunitaria

Es un tipo de nube que genera controversia en las distintas bibliografías, que en alguna de estas no aparece (por ejemplo en *"Handbook of Cloud Computing"* o *"The Definitive Guide To Cloud Computing"*) y en otras como el NIST o en el libro "*The basics of Cloud Computing: understanding the fundamentals of Cloud Computing in theory and practice"* sí que lo hace. Pero es necesario mencionarla, ya que posee características que la diferencian de los tres tipos anteriores.

Se trata de una nube semipública, un tipo de nube híbrida, que posee más características de la nube pública. Es creada por un grupo de organizaciones con un objetivo en común que necesitan la nube pública, y a su vez necesitan más privacidad de la que esta le ofrece.

En la nube comunitaria, los datos que precisan esta privacidad no se encuentran físicamente en centro de datos, sino que se encuentran en los ordenadores de los usuarios que la componen, es decir, en los ordenadores de las organizaciones que la implementan.

Ventajas:

- Las de la nube pública, con un grado de seguridad mayor.

2.3.5. Comparativa entre nube pública, privada e híbrida

	NUBE PÚBLICA	NUBE PRIVADA	NUBE HÍBRIDA
Uso exclusivo	NO	SI	SI
Gran nivel de seguridad y privacidad	NO	SI	NO
Localización dentro de la empresa	NO	SI/NO	NO
Mínima inversión	SI	NO	SI

Tabla 1. Comparativa entre nube pública, privada e híbrida

2.4. Modelos de servicio de la nube

Cuando nos adentramos más allá e intentamos saber qué servicios pueden ser proporcionados por los proveedores de la nube, hablamos de los modelos de servicio de la nube. Según el NIST (National Institute of Standars and Technology: https://www.nist.gov/) existen tres modelos básicos de servicio en la nube, los cuales son Infraestructura como servicio, Plataforma como servicio y Software como servicio. A pesar de esta definición, según avanza la tecnología y surgen nuevas necesidades, también estos modelos van cambiando y dando lugar a otros tipos de modelos que se adaptan a la actualidad y dan soluciones a necesidades específicas, es por ello, que se tratarán también en este documento.

2.4.1. Servicios ofrecidos

Antes de entrar a tratar cada uno de los modelos, es necesario definir los servicios existentes, para ver cuales serán ofrecidos por cada uno de los modelos. Para definir estos servicios, nos basamos en un desarrollo tradicional On-Premises, es decir, localmente.

- Aplicaciones
- Información
- Tiempo de ejecución
- Middleware
- Sistema Operativo
- Virtualización
- Servidores físicos
- Almacenamiento
- Networking

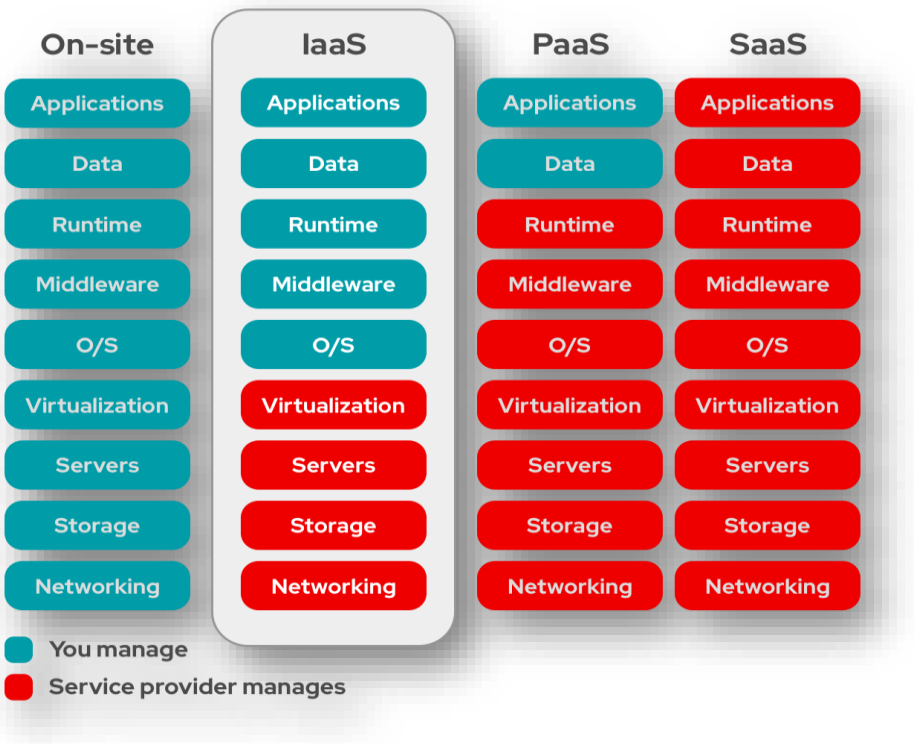


Ilustración 1. Servicios ofrecidos por cada uno de los modelos frente al desarrollo On-Site (también conocido como On-Premise). En azul lo que el cliente gestiona y en rojo lo que el proveedor gestiona.

2.4.2. Infraestructura como servicio (IaaS)

Se trata del modelo en el cual el proveedor de servicios de la nube, es responsable únicamente de la parte hardware, es decir, el networking, almacenamiento y servidores, y además de la virtualización. Es lo mínimo de lo que un proveedor puede encargarse, es decir, se trata del modelo a más bajo nivel, y en el cual el cliente/usuario de la nube, es responsable de más servicios, partiendo del sistema operativo hasta las aplicaciones.

Es por ello, que una vez se recibe todo esto por parte del proveedor, el usuario es el que decide que tipo de servidor quiere, pudiendo elegir si quiere que corra sobre Linux, Windows, etc. Por ello, también se considera a este servicio como el servicio más flexible.

DEFINICIÓN

> "IaaS (Infrastructure as a Service) es un servicio informático en la nube que da acceso a una **infraestructura de TI altamente flexible** a través de Internet. El alojamiento, la administración y el mantenimiento del hardware que la sustenta recaen completamente en manos del proveedor del servicio. El modelo típico de facturación de las ofertas de IaaS sigue el principio ***pay per use*** por el que los abonados solo pagan por lo que usan."

Definición de Ionos: https://www.ionos.es/digitalguide/servidores/know-how/que-es-iaas/

RESPONSABILIDADES Y FUNCIONAMIENTO

Por parte del proveedor, este será el encargado de la instalación, funcionamiento y seguridad en el hardware (el cual le va a proveer al cliente acceso a recursos de computación -procesador, memoria RAM, disco duro- y estructuras de red integradas -cortafuegos, routers y sistemas de seguridad y backups-), para que el cliente pueda disfrutar en cualquier momento de los servicios. Para ello, se encargará de montar y mantener la infraestructura en el centro de datos, proteger este de cualquier evento que pueda pasar, dar

potencia de cálculo a la CPU y RAM, tener estructuras de red, proveer estructuras de servidores, red y bases de datos y crear un entorno de virtualización eficiente y eficaz, con un software apropiado a este.

Básicamente, a través de este modelo, adquieres un servidor virtual dentro del centro de datos del proveedor (siempre y cuando no hablemos de nube privada, en este caso, tendremos nuestro propio servidor físico en nuestro centro de datos), y a raíz de este, puedes crear tu propio entorno.

VENTAJAS Y DESVENTAJAS

Las ventajas que nos ofrece el uso de IaaS son las siguientes. En primer lugar, no es necesario preocuparse por el hardware, haciendo desaparecer los costes y los problemas derivados de este; la implementación de proyectos es más rápida; y la escalabilidad rápida hace que sea más flexible frente al desarrollo On-Premise.

En cuanto a las desventajas, destacan las siguientes: es muy dependiente de la conexión a la red y al proveedor, por lo que el total control que el usuario poseía desaparece. Además, en este modelo de servicio, el cambio de proveedor es muy costoso.

2.4.3. Plataforma como servicio (PaaS)

Sus siglas son PaaS, del inglés *Platform as a Service*, es decir Plataforma como Servicio. Como su propio nombre indica se le ofrece al cliente una plataforma para sus necesidades de cómputo, la cual puede ir desde un simple sistema operativo hasta una plataforma de desarrollo completa con servidor web y bibliotecas de desarrollo.

Se trata del modelo intermedio entre IaaS y SaaS, ya que el proveedor de servicios aprovisiona al cliente de más servicios y lo hace menos responsable que en IaaS, pero lo aprovisiona de menos y lo hace más responsable que en SaaS.

DEFINICIÓN

> "La **plataforma como servicio** (PaaS) permite a los clientes alquilar una plataforma virtual en la que pueden desarrollar, probar y desplegar aplicaciones web para los usuarios."

Definición de Ionos: https://www.ionos.es/digitalguide/servidores/know-how/paas/

RESPONSABILIDADES Y FUNCIONAMIENTO

Entre estos servicios de los que es responsable el proveedor, se encuentran los pertenecientes a IaaS, además de un paquete de herramientas que permite al usuario desarrollar aplicaciones de inmediato. Es decir, el proveedor se encarga de la infraestructura básica como los servidores, almacenamiento, networking, sistemas operativos y middleware, y además de recursos como herramientas de desarrollo, lenguajes de programación, sistemas de gestión de bases de datos y técnicas de contenedores. Permite al usuario desarrollar su aplicación de una manera más sencilla, centrándote únicamente en la programación de esta.

Por ejemplo, a la hora de desarrollar una aplicación, la plataforma de bases de datos será suministrada por el proveedor, y este será responsable de mantenerla actualizada y en buen estado para ser usada, pero será el cliente el responsable de los datos que haya en ella, sin que el proveedor se entrometa en esto.

Al hablar de lenguajes de programación, hay que destacar que en PaaS solo se dispondrán de unos cuantos lenguajes de programación, dependiendo del proveedor, por lo tanto, es importante elegir bien el proveedor.

El funcionamiento de Paas permite que desarrolles tu aplicación de la misma forma en la que lo harías en un entorno propio, y una vez el código este listo, la plataforma en la nube se encargará de desplegarlo y ejecutarlo en un contenedor, permitiendo hacerlo tantas veces como quieras e incluso a la

misma vez. De esta forma, no tienes que preocuparte por la falta de memoria de tu dispositivo, el poco espacio en disco o un procesador saturado.

En este proyecto, usaremos este modelo de servicio de la nube para desarrollar una aplicación, y mostraremos más adelante (CAPITULO TAL) paso a paso cómo se realizará. Siendo este (desarrollo de aplicaciones) el área de aplicación principal de PaaS, también existen otros tales como: desarrollar o ampliar nuevas APIs, analizar datos de gran volumen o como plataforma de comunicación (mensajería de voz y vídeo) entre otros.

VENTAJAS Y DESVENTAJAS

La ventaja más destacada y que hace tan llamativo el uso de PaaS es que el desarrollo es mucho más sencillo y veloz, dejando de lado la parte de la infraestructura, que tantos quebraderos de cabeza puede ocasionar. Otra ventaja destacada, y una ventaja heredada de las ventajas generales de la nube es la escalabilidad, que permite aumentar o disminuir con mucha rapidez los servicios que se demandan.

En cuanto a las desventajas, el no poder modificar la infraestructura puede llegar a limitar el desarrollo en algún punto. Por ejemplo, solo se permite usar los lenguajes de programación proporcionados por el proveedor, lo que en ciertos proyectos puede ser una traba.

Es por ello, que decidirse por PaaS depende mucho del tipo de proyecto que se quiera realizar y las necesidades de este.

2.4.4. Software como servicio (SaaS)

En mucha bibliografía, se considera a SaaS como el modelo de servicio original, ya que el concepto de Cloud Computing está muy arraigado con este modelo.

El origen de este modelo está en los proveedores de servicios de aplicación o ASP, ya que por fuera, tienen muchísimas similitudes, pero cuando adentras en ambas, se aprecian diferencias significantes, que hacen de los

ASP una tecnología diferente y cada vez más anticuada. Estas diferencias claves comienzan en que las aplicaciones de los ASP se basaban en la tecnología cliente/servidor, mientras que las aplicaciones de SaaS no necesariamente necesitan de un cliente (como ya se observó anteriormente) ya que son aplicaciones basadas en la web.

Como su propio nombre indica, en este modelo, *Software as a Service* (Software como servicio), lo que se le ofrece al usuario es un software, al que podrá acceder a través de internet y no será necesario realizar una instalación local. Las responsabilidades en este modelo, recaen prácticamente en su totalidad de lado del proveedor, ya que el cliente solo es encargado del acceso y uso del software. Por lo tanto, el proveedor de servicios de la nube será el responsable tanto de la creación del hardware (servidores, networking, almacenamiento…) y del software (información, aplicación…) como del mantenimiento y actualización de los mismos, sin que el usuario sea necesariamente consciente de ello.

DEFINICIÓN

> **"Software as a Service** es un modelo de distribución de software donde el software y los datos se alojan en servidores del proveedor y se accede con un navegador web a través de Internet."

Definición de IEB School: https://www.iebschool.com/blog/que-es-saas-definicion-ventajas-digital-business/#:~:text=Un%20sistema%20SaaS%20o%20Software,servidor%20externo%20a%20la%20empresa.

RESPONSABILIDADES Y FUNCIONAMIENTO

El funcionamiento de SaaS se basa en el aprovisionamiento por parte del proveedor de una herramienta centralizada que puede ser accesible de manera online por los usuarios a través de cuentas individuales. Es decir,

Se puede decir que este modelo está muy de moda ya que se trata de una herramienta fácil de usar y que la mayor parte de los usuarios de Internet

han usado (por ejemplo, en el entorno universitario, hay muy pocos casos que no haya usado Google Docs, una herramienta muy sencilla y con un gran potencial, sobre todo a la hora de realizar un trabajo en grupo, ya que, en este caso, está muy por encima de cualquier herramienta de ofimática local). Su popularidad crece cada día, ya que está alcanzando un rendimiento muy alto, igualando (en algunos casos hasta superando) el rendimiento de los softwares similares locales, y además de aprovechar las ventajas que la nube ofrece sobre estos.

Aun así, la aceptación de SaaS en algunos casos es difícil por la desconfianza que genera la seguridad en el Cloud Computing, algo que día tras día se va erradicando y hacen que SaaS esté creciendo mucho y sea una tecnología de presente y sobre todo de futuro (cuando se solucionen los problemas que hoy día presenta).

VENTAJAS Y DESVENTAJAS

Como ventajas de SaaS destaca lo siguiente. El SaaS puede llegar a ser muy ventajoso en ciertos entornos, ya que posee características muy atractivas como una rápida puesta en funcionamiento, evitando instalaciones costosas y falta de espacio en disco; el usuario no tiene que preocuparse por el mantenimiento, además de poder disfrutar en todo momento de las últimas versiones de las herramientas del software sin preocuparse de actualizarlas, y esto incluye las actualizaciones de seguridad; el acceso se realizará desde cualquier dispositivo (por lo general, suele ser multiplataforma); y obviamente hereda una de las ventajas de la nube, y solo pagarás por lo usado, omitiendo el gran coste de las licencias de la mayoría de los softwares locales.

Centrándonos en las desventajas que posee el modelo de servicio SaaS, volvemos a hablar de una desventaja generalizada, y es que SaaS es muy dependiente de la calidad de la conexión a la red, por lo tanto, una caída de esta por el factor que fuere, imposibilitaría la capacidad de acceso a la aplicación de SaaS. Además, y a pesar de las medidas de seguridad existentes, la cesión de los datos al proveedor de la nube sigue siendo una desventaja frente al software instalado de manera local. Por último, y a menor

nivel, se trata de que los proveedores, en alguna ocasión usan el software que se ofrece a los usuarios como campos de pruebas, y acaban ofreciendo software inacabado que puede dar lugar a bugs.

EJEMPLOS DE SAAS

Google posee una gran cantidad de software disponible a través de la nube, hemos mencionado anteriormente uno de ellos, Google Docs, una herramienta de ofimática, además posee Google Drive, una herramienta de almacenamiento, o entre otra también se encuentra Google Calendar, etc.

Podemos hablar también de otra herramienta de almacenamiento como Dropbox, o Mega, o herramientas de correo electrónico, las cuales son las más conocidas (como Gmail de Google u Outlook de Microsoft). También existen herramientas para el mundo laboral como Slack o Trello, sistemas de gestión de contenido (CMS) y un sinfín de finalidades variadas como Netflix o HBO.

2.4.5. Otros modelos de servicio

En este apartado vamos a hablar de otros modelos de servicio no tan importantes como los tres anteriores, pero que cada día aumentan su importancia. Además, poco a poco surgen nuevos modelos, a raíz de las necesidades de los usuarios. A continuación, se definirán los más populares:

XaaS (Todo como servicio)

Este modelo de servicio no es un modelo en sí, sino que se trata de una alternativa diferente. El cliente contrata XaaS para beneficiarse de todas las soluciones de la nube. XaaS proporciona al usuario infraestructura, plataforma, software, etc.

A pesar de las ventajas que ofrece, el poder tener todo en uno, XaaS puede ocasionar problemas, debido a la no definición clara de lo que en realidad es.

DBaaS (Base de datos como servicio)

Este modelo se podría encajar perfectamente como un tipo de IaaS, es por ello, que se encuentra al mismo nivel. Aun así, al tratarse de algo tan específico y tener sus propias características, en muchas bibliografías es considerado como un modelo a parte.

Como su propio nombre indica, DBaaS permite al cliente obtener las ventajas de la nube a la hora de crear y gestionar una base de datos. El proveedor de servicios se encargará de mantener la estructura física y la base de datos, siendo el cliente el encargado de los datos de esta y las operaciones que se realicen en ella.

Como bien es sabido, existen dos tipos de bases de datos: SQL y NoSQL. Las bases de datos SQL son bases de datos con poca escalabilidad, es por ello, que les cuesta encajar en la tecnología de la nube; por el contrario, las bases de datos NoSQL, las cuales poseen un gran número de lecturas/escrituras y por ello una gran capacidad de escalabilidad, se adaptan con mucha facilidad a la nube, aprovechando eficientemente las ventajas de esta.

FaaS (Función como servicio)

Faas es un modelo que ofrece a los usuarios una abstracción para poder ejecutar aplicaciones en respuesta a ciertos eventos en contenedores sin estado, es decir, se trata de una tecnología serverless.

En principio, se podría confundir con PaaS, pero la principal diferencia es que se trata de un modelo de ejecución basado en eventos, es decir, está siempre disponible, pero ningún proceso está ejecutándose en segundo plano, a diferencia de PaaS.

HaaS (Hardware como servicio)

Este tipo de “modelo de servicio” poco tiene que ver con la nube. Es todo lo contrario a IaaS, ya que se “alquila” un equipo físico perteneciente al proveedor.

DRaaS (Recuperación ante desastres como servicio)

Es un modelo de servicio enfocado a la seguridad del usuario. Basado en la replicación, mediante el alojamiento de los datos en los datacenters pertenecientes al proveedor, está diseñado para garantizar la continuidad de las operaciones de una empresa.

Además de la replicación de los datos, incluye aplicaciones y bases de datos, junto con técnicas que permiten reestructurar la actividad de la empresa en caso de que suceda un evento y se pierda parte o la totalidad de los medios.

Para poder disfrutar de DRaaS, en primer lugar es necesario realizar un análisis de los riesgos que puedan suceder, además de fijar los puntos críticos los cuales habría que recuperar en caso de pérdida.

Se trata de un modelo esencial para las empresas en las cuales la información sea un activo importante, y un modelo muy atractivo para cualquier empresa.

CaaS (Contenedor como servicio)

Cuando se habla de CaaS, se habla de *Container as a Service*, un modelo de servicio de la nube que permite la virtualización a través de contenedores, sin tener que instalar toda la infraestructura que se necesita. Al usuario se le ofrece la capacidad de desarrollar, probar y ejecutar software en contenedores ofrecidos por los proveedores.

Los contenedores son una herramienta de virtualización que permite aislar totalmente la ejecución de aplicaciones, siendo esta virtualización a nivel de sistema operativo.

Es por esto, que se encuentran en un nivel intermedio entre IaaS y PaaS, diferenciándose de estos por el concepto de virtualización usado, un concepto de virtualización basado en contenedores, el cual se encuentra a más alto nivel que el concepto usado en IaaS.

STaaS (Almacenamiento como servicio)

Este modelo se basa únicamente en alquilar una infraestructura de almacenamiento al proveedor de servicios para almacenar datos. Es muy útil en pequeñas empresas para realizar copias de seguridad de sus datos.

CaaS (Comunicaciones como servicio)

El proveedor de servicios ofrece servicios de comunicación como Voz sobre IP (VoIP) , mensajería instantánea o herramientas de videoconferencia. El proveedor de servicios es el responsable de todo el hardware y el software, incluyendo calidad de servicio.

De igual forma que sucede con IaaS y DBaaS, este modelo podría ser incluido como un tipo de SaaS, pero de igual forma, posee características que pueden hacerlo independiente de SaaS.

NaaS (Red de Trabajo como servicio)

Del inglés *Networking as a Service*, en este modelo se ofrece la capacidad de crear y manejar redes desde la nube. Permite al cliente disfrutar de la red sin tenerla en propiedad. Este modelo incluye tecnologías tales como las redes virtuales (VPN), ancho de banda bajo demanda (BoD) y redes virtuales móviles.

Se trata de un modelo no muy extendido, pero al que se le augura una gran expansión.

SECaaS (Seguridad como servicio)

SECaas o *Security as a Service* es un modelo el cual ofrece la provisión de servicios de seguridad desde la nube, con las ventajas de ser más rápido y ágil, escalable y en algunos casos incluso automatizada.

Este modelo abarca los siguientes campos:

- Gestión de Identidades y Accesos.
- Soluciones de prevención de pérdida de datos (DLP).
- Seguridad Web.
- Seguridad del Correo Electrónico.
- Evaluación de la Seguridad.
- Gestión de la Intrusión.
- Gestión de la Información de la Seguridad y Gestión de eventos.
- Encriptación.
- Continuidad de negocio y Recuperación ante desastres.
- Seguridad de las Redes.

2.5. Evolución a lo largo del tiempo

Al igual que la definición de Cloud Computing, decir cuándo surgió con exactitud esta tecnología, es difícil. Esto es así por el hecho de que se trata de una tecnología que no tiene un origen exacto, sino que ha sido fruto de diversos avances en la tecnología que han concluido en la aparición de la nube.

El término Cloud Computing lleva presente con nosotros desde hace más de 20 años, pero los orígenes de este, se remontan varios años y décadas atrás.

El origen de esta tecnología se remonta a los años 50, cuando las empresas precisan el acceso a la información desde varios puntos. Sin

embargo, debido al gran tamaño y coste de la infraestructura de un ordenador, se hacía inviable el hecho de tener uno en cada oficina.

Fue ya en la década de los 60, cuando el profesor John McCarthy, pionero en el campo de la Inteligencia Artificial, concretamente en 1961, durante su discurso para celebrar el centenario del MIT, dio a conocer su teoría del tiempo compartido (Time-Sharing), en la que sugería apostar por la computación en sistema compartido, es decir, diferentes recursos tecnológicos trabajaban de manera conjunta. De esta forma, sugería poder vender la capacidad de procesamiento de los ordenadores como un servicio público, tal como se proporciona el agua.

El tiempo compartido del que hablaba John McCarthy, hace referencia a la capacidad de compartir un recurso de cómputo de forma concurrente a través de software.

Durante esta década, la idea de “alquilar tiempo de computación” se hizo muy popular, pero las limitaciones técnicas existentes en la época impidieron que se materializara esta idea. A finales de esta época, se ofrecían paquetes que incluían editores de texto, entornos de desarrollo para lenguajes de programación, almacenamiento de archivos, paquetes de ofimática y soluciones de impresión a cambio de un alquiler en función del tiempo y uso de los recursos.

Un año más tarde, en 1963, sería Joseph Carl Robnett Licklider (J.C.R. Licklider, para muchos el verdadero “creador” del Cloud Computing), quien tras la idea de una informática compartida, introduce la idea de una red intergaláctica de computación, mediante la cual podríamos compartir información con cualquier usuario y acceder a los datos y programas desde cualquier ubicación.

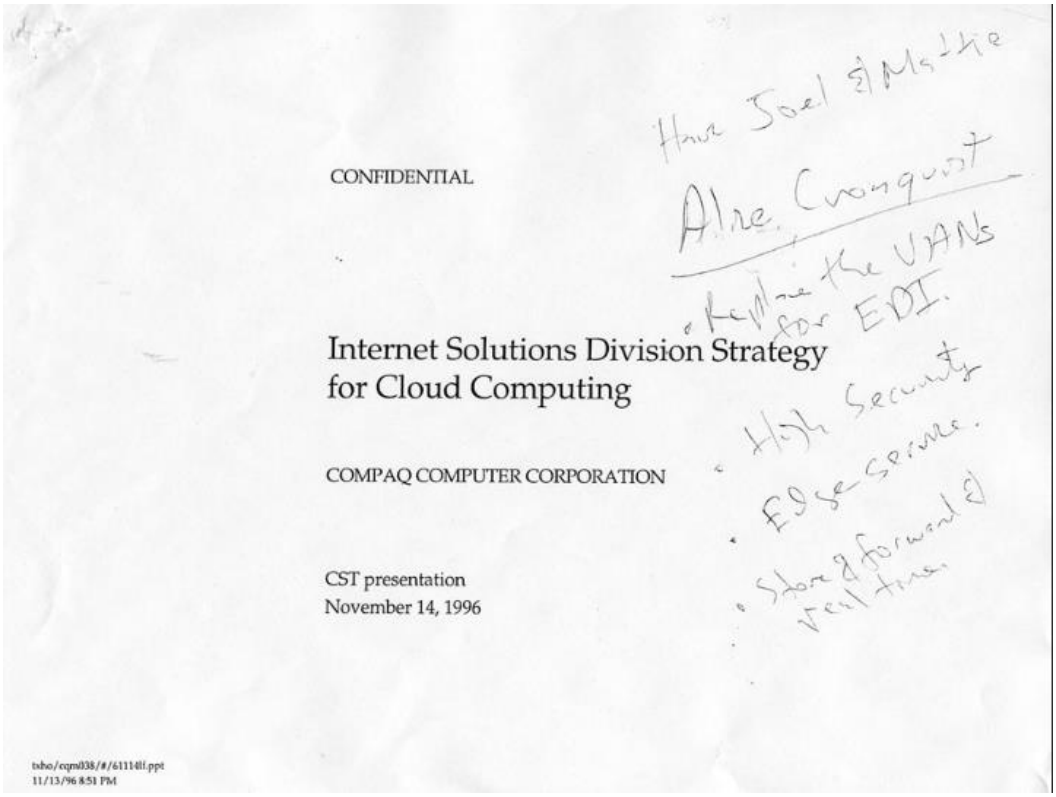
«*Consideren un caso en el que diferentes centros de datos están conectados, con su propio lenguaje y su forma de hacer las cosas. ¿No sería deseable, incluso necesario que todos se pongan de acuerdo para usar el mismo lenguaje o, al menos, tener una convención para preguntar en el*

lenguaje que habla el otro?» afirmó Licklider en un discurso que ahora se torna casi profético.

Unos meses después, en abril de 1963, J.C.R. Licklider da forma a un esbozo de una red de computadores que presenta a ARPA (Agencia de Investigación de Proyectos Avanzados), y acaba en el Pentágono, donde, en plena guerra fría convence a los mandos militares de que desarrollar esta idea les daría ventaja sobre los soviéticos. Así, comenzó el desarrollo de ARPAnet, precursora de Internet, una tecnología que lo cambió todo, y base del Cloud Computing.

Sin embargo, ambos proyectos quedaron en suspensión, pero sin quererlo, pusieron la semilla de lo que hoy conocemos como Cloud Computing. El estado de estos proyectos es debido a la caída de precios que experimentó la informática en los años setenta y ochenta, que dejaron de lado los avances experimentados en Cloud Computing.

No fue hasta la década de los 90, cuando Internet ya era una tecnología madura y estaba asentada, que comenzó la expansión del Cloud Computing. El ancho de banda existente permitía intentar desarrollar esta tecnología. Y justo en esta década, en 1996, apareció el término “cloud computing”, de la mano de George Favaloro y Sean O’Sullivan.


CONFIDENTIAL

Internet Solutions Division Strategy
for Cloud Computing

COMPAQ COMPUTER CORPORATION

CST presentation
November 14, 1996

Ilustración 2. Portada del proyecto presentado por George Favaloro y Sean O'Sullivan en 1996. https://s3.amazonaws.com/files.technologyreview.com/p/pub/legacy/compaq_cst_1996_0.pdf **enlace del documento.**

Es a finales de los 90, cuando Salesforce inventa lo que hoy se conoce como aplicaciones empresariales en Internet, a través de la web Salesforce.com, convirtiéndose en la primera empresa que utiliza cloud computing.

Tras esto, Amazon sorprende y comienza a usar la nube dentro de su empresa. Es en 2002, y tras percatarse de que solo usa el 15% de toda su estructura informática, que lanza AWS (Amazon Web Services), un sistema novedoso de almacenamiento en la nube en el que puede ejecutar aplicaciones y manejar información y así ponían todo su hardware al servicio de la nube.

Es en 2006, cuando Amazon comienza a prestar estos servicios a pequeñas y medianas empresas, desarrollando EC2 (Elastic Computer Cloud). A través de este sistema, permitía alquilar servidores a las empresas para almacenar sus datos y ejecutar aplicaciones. Debido a esto muchas organizaciones prestaron gran atención a la tecnología del Cloud Computing y comenzó una gran expansión.

Tres años más tarde, en 2009 Google jugó un papel muy importante en la historia del Cloud Computing, lanzando su paquete de aplicaciones en la nube (Gmail, Google Docs, Google Drive, etc.). Tras esto, otras grandes empresas de informática se sumaron a esta tecnología como Microsoft (Microsoft Azure) o Apple con iCloud.

A día de hoy el Cloud Computing sigue creciendo y se siguen añadiendo funcionalidades que hacen que vaya evolucionando. Todos estos avances en la nube han creado un ecosistema que crece de manera increíble y que está presente en nuestro día a día, donde gran parte de lo que hacemos en Internet, o incluso fuera de este, depende en gran medida de la tecnología del Cloud Computing.

3. TECNOLOGÍAS ASOCIADAS AL CLOUD COMPUTING

El objetivo de este punto es definir las tecnologías relacionadas con el Cloud Computing, que han ayudado y ayudan al desarrollo y crecimiento de esta tecnología, tanto como base para el nacimiento o apoyo para desarrollar nuevas funcionalidades dentro de la misma.

3.1. Virtualización

Cada vez que hablamos de Cloud Computing, es imposible no hablar implícitamente de virtualización. Son dos tecnologías que van cogidas de la mano, especialmente, de lado del Cloud Computing.

En primer lugar, ¿qué es la virtualización?

La virtualización se trata de una tecnología que permite crear de forma virtual a través de software algún recurso tecnológico, como un servidor, sistema operativo, disco duro, etc. Por ejemplo, permite ejecutar varios sistemas operativos sobre otro sistema operativo.

Es decir, a través de la virtualización, se aprovechan los recursos hardware de un dispositivo para emular la existencia de otros o de partes de otros sobre este hardware. Por lo tanto, estos dispositivos/entornos virtuales, están limitados a las características del propio hardware.

Cada uno de estos entornos virtuales, suelen ser denominados guests, mientras que el hardware que hospeda a los visitantes o guests, se denomina host.

Para la existencia de la virtualización, es necesario que exista algo llamado hipervisor, una capa intermedia entre el hardware y el software de virtualización. El hipervisor es el encargado de repartir según lo indicado, los recursos del hardware entre los guests o los entornos virtuales.

Existen muchos tipos de virtualizaciones, pero en el caso del Cloud Computing, la virtualización de servidores, redes, sistemas operativos y escritorios y recursos de almacenamiento son las que se usan.

La virtualización es una de las tecnologías más importantes sobre la que se basa el funcionamiento básico del Cloud Computing. Es parte de la infraestructura técnica que permite, entre otras cosas, que se pueda aumentar la eficiencia del Cloud Computing y ofrecer servicios alojados en la nube de alto rendimiento.

Es gracias a la virtualización que podemos crear pools de recursos, algo básico en la tecnología de la nube, que permite que varios usuarios puedan acceder a la capacidad de almacenamiento o la potencia de cálculo que requieren al mismo tiempo.

Decir que el Cloud Computing puede existir sin la virtualización es atrevido, pero es concebible, no es un requisito imprescindible, pero sí fundamental, ya que sin esta, el eficiencia del Cloud Computing caería tanto, que no sería una alternativa en ningún caso.

3.2. Contenedores

Hablar de contenedores, es hablar de la herramienta anterior, la virtualización. Se trata de una forma de virtualización, que al contrario de las máquinas virtuales tradicionales - las cuales permiten la virtualización de la infraestructura de computación -, habilitan la virtualización de las aplicaciones de software, es decir, utilizan el sistema operativo de su host en lugar de proporcionar uno.

Los contenedores dividen las aplicaciones en paquetes de código más pequeños, más fáciles de administrar y autónomos, además de los requisitos previos que las aplicaciones de software tienen que operar de manera independiente del host.

Se trata de una tecnología que está cogiendo mucha relevancia debido a que necesitan unos recursos informáticos mínimos, son rápidos y muy fáciles de manejar e instalar. Además, poseen una adaptación a la nube muy buena, y su combinación se está poniendo de moda en los últimos años, ya que esta posee muchas ventajas. Incluso existe un propio modelo de servicio específico para los contenedores (Caas – *Container as a Service*), como mencionamos anteriormente.

Según la investigación de ESG, los contenedores constituyeron aproximadamente el 19% de las cargas de trabajo de la nube híbrida de 2018, y se prevé que para este año 2020, sea un tercio de las cargas de trabajo de la nube híbrida.

Estos datos avalan el gran crecimiento que están teniendo estas dos tecnologías cuando se combinan, ofreciendo una gran cantidad de beneficios:

- Portabilidad en la nube: ahorra tiempo a los programadores, ya que no tienen que reescribir código nuevo para cada sistema operativo y plataforma de la nube.
- Plataforma común para la distribución de aplicaciones.
- Poseen una forma estándar de desarrollar e implementar servicios.
- Proporcionan consistencia a las plataformas, lo que mejora la portabilidad.
- Ayudan a que las aplicaciones en construcción sean más rápidas y sencillas.

Si a todas estas ventajas se aplican dentro de un sistema en la nube, se podrá tener capacidad para poder desarrollar aplicaciones de una manera más sencilla e intuitiva, se crearán contenedores para poder lanzar partes de estas aplicaciones, y a su vez, se podrán gestionar y orquestar a través de la propia plataforma de la nube (por ejemplo, a través de Kubernetes en Google Cloud https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/), entre otras aplicaciones de esta tecnología.

3.3. Servicios web

Se trata de una tecnología que permite la intercomunicación e interoperabilidad entre máquinas conectadas en red. Generalmente, suele basarse en el envío de solicitudes por parte de un cliente a un servidor, y este responde con los datos que el cliente solicitaba.

Por un lado existe el protocolo SOAP que usa XML, siendo esta combinación la primera introducción de los web services en el mundo de Internet, y por el otro lado se encuentra REST, que usa la codificación JSON para mandar sus datos, siendo esta última opción la más moderna y la más simple, con un gran potencial.

Pues bien, los servicios web y la nube están muy relacionados, ya que a través de estos, es como principalmente se puede acceder a los servicios que ofrece la nube. Cada proveedor ofrece una API (Interfaz de programación de aplicaciones) mediante la cual se pueden realizar todas las operaciones que ofrezca este. Es por ello, que se trata de una tecnología fundamental para el despliegue de la nube.

3.4. Microservicios

Se trata de una tecnología relativamente reciente, que permite la implementación y desarrollo de una aplicación como un conjunto de pequeños servicios, cada uno ocupándose de su parte y comunicándose entre ellos con mecanismos sencillos (por lo general una API a través de HTTP).

Esta forma de enfocarse, se enfrenta a la forma clásica o arquitectura monolítica de desarrollar aplicaciones. Esta forma lo que sugiere es crear una aplicación como un bloque compacto, siendo muy dependiente del hardware sobre el que se instala.

Los microservicios son el futuro, y gran parte de esto es gracias a la nube, ya que esta ha favorecido la proliferación de aplicaciones basadas en microservicios, debido al enfoque ofrecido por los proveedores de servicios en la nube.

Para poder disfrutar de todos los beneficios de rentabilidad, escalabilidad y disponibilidad, es necesario disponer de una plataforma de microservicios, y es aquí donde entra la nube, ya que la mayoría de las plataformas de microservicios se encuentran en la nube (por ejemplo OpenShift o Pivotal Clouf Foundry).

Pero no solo la nube aporta beneficios a esta tecnología, sino que los microservicios son una herramienta clave en el desarrollo de las plataformas sobre las que son ofrecidas la nube. Por ejemplo, Netflix se trata de una aplicación construida sobre microservicios, y a su vez, es un ejemplo de *Software as a Service* (SaaS), por lo que el impacto es mutuo.

4. VENTAJAS, INCOVENIENTES Y RETOS DEL CLOUD COMPUTING

El tema a tratar en este punto se centra en la repercusión que tiene el Cloud Computing, analizado desde el punto de vista de las ventajas y desventajas existentes, y los retos que aun le faltan por cumplir.

4.1. Ventajas

- **Reducción de costos**

El uso de Cloud Computing permite reducir una gran cantidad de costos debido al ahorro que supone no tener que adquirir hardware (servidores físicos) ni software (licencias), y no tener que realizar un mantenimiento periódico de este. Es una ventaja muy llamativa de Cloud Computing y que atrae a las empresas, ya que les permite ganar dinero con unos gastos mínimos.

- **Flexibilidad, escalabilidad y movilidad**

A medida que el uso de la nube aumenta o disminuye, esta automáticamente tiene la capacidad de aumentar y disminuir en proporción al uso que se hace de ella. Esto es gracias a la capacidad de escalabilidad que nos ofrece la nube, ya que por ejemplo si una aplicación necesita una herramienta en un momento específico, esta se añadirá automáticamente.

Además, la nube nos ofrece la posibilidad de acceder en cualquier momento y desde cualquier lugar, a nuestros datos alojados en esta, ya que estos se alojan en la red de servidores de la nube. Esta es una ventaja que hace muy apetecible el uso de Cloud Computing, ya que aporta independencia del hardware donde se han realizado las operaciones.

- **Confiabilidad, innovación y eficiencia**

Usar Cloud Computing es una decisión que aporta fiabilidad y consistencia, ya que se trata de un servicio que se encuentra disponible 24 horas al día, 7 días a la semana, 365 días al año, con un 99,99% de

disponibilidad. Es más, esta disponibilidad a medida que avanza la tecnología va en aumento. Esto es posible, ya que los proveedores de servicios actualizan las soluciones de manera regular para ofrecer a los usuarios la tecnología más actualizada posible.

El hecho de poseer la tecnología más innovadora del mercado, repercute del lado del usuario aportando un valor estratégico a las soluciones que propone este en la nube. Además de este hecho, la nube ofrece la posibilidad de lanzar las aplicaciones al mercado rápidamente, sin necesidad de preocuparse por la infraestructura o el mantenimiento, lo que hace que la aplicación sea más competente dentro del mercado.

- **Green IT**

Dejando de lado las ventajas que ofrece la nube desde el punto de vista tecnológico, desde el punto de vista ecológico, el Cloud Computing presenta grandes ventajas.

Green IT se define como “la máxima optimización de los recursos informáticos en las empresas pero con el mínimo impacto en el medioambiente.” Definición de Nubit: https://www.nubit.es/que-es-el-green-computing/

El Cloud Computing es un gran aliado del Green IT, ya que como se ha comentado, el uso de esta tecnología implica la reducción del hardware, que a su vez, conlleva el beneficio de la reducción de materiales de plástico (formados principalmente por hidrógeno y carbono, silicio, fósforo y azufre) y de metal (que contiene hierro en forma de acero y aluminio), que ayuda a reducir la contaminación. A este beneficio de reducir materiales, también se le añade el proceso de fabricación de estos, que conlleva una gran cantidad de costes y de contaminación en el proceso.

Todo esto, gracias a que estos componentes físicos, se convierten en virtuales.

Esta información ha sido sacada del siguiente libro: *Green Cloud Computing: An Energy Efficient Methodology for Cloud Computing Enviroment.*

Es por ello, que si se desea saber más sobre este tema, es recomendable la lectura del mismo.

4.2. Inconvenientes

- **Seguridad y privacidad**

Es la principal desventaja del uso de Cloud Computing, ya que ceder los datos a un red de servidores cuyo lugar físico desconocemos o es inaccesible genera cierta desconfianza por parte del cliente. Además, el hecho de ceder datos confidenciales y sensibles como planes de mercado, lanzamientos de productos o datos financieros, a una empresa supone para la mayoría una miedo ante la posible pérdida de la privacidad, además de una sensación de descontrol sobre tus datos. Aun así, se trata de un punto que va creciendo y mejorando día a día, aplicando las nuevas actualizaciones.

- **Dependencias**

El uso de Cloud Computing lleva implícito las dependencias. En primer lugar, la dependencia de la conectividad, ya que sin una buena conexión no sería posible acceder a los datos de la nube. Además, el buen estado de la red es sensible a diversos factores como la climatología, o grandes masas de usuarios.

En segundo lugar, la dependencia del proveedor de servicios en la nube, ya que a pesar de ofrecer una gran flexibilidad y control, hay ciertos aspectos que no quedan lejos del alcance del usuario, ya que la infraestructura de la nube sigue siendo propiedad de estos, y tienen ciertas políticas sobre el uso que hay que cumplir. De hecho, migrar de un proveedor a otro es una tarea ardua ya que existen incompatibilidades entre distintos proveedores.

4.3. Retos

- Computing everywhere: Permitir el acceso desde cualquier lugar y dispositivo, llevando la computación en la nube a lugares.

- Cloud – client computing: Integrar la nube con los dispositivos móviles inteligentes para así potenciar las propias aplicaciones de la nube.
- Asignar a cada aplicación de manera automática y precisa la infraestructura y comunicaciones necesarias en cada momento.
- Aumento de la seguridad y privacidad, aplicando las nuevas tecnologías existentes para evitar así el robo de datos o la suplantación de cuentas.
- Incremento de la disponibilidad, intentando que aumente lo máximo posible para que cada vez se acerque más al 100%.
- Integración rápida y eficaz de nuevas tecnologías, para mantener al cliente de la nube a la vanguardia de la tecnología.

5. PROVEEDORES DE CLOUD COMPUTING

En este punto, el tema a tratar son los proveedores de Cloud Computing. En primer lugar, se realizará una definición de estos, seguido de un análisis de los principales proveedores existentes en el mercado.

5.1. ¿Qué son los proveedores de Cloud Computing?

A lo largo de este documento se ha mencionado proveedores de Cloud Computing o proveedores de servicios en la nube, pues bien, se va a definir qué son.

Según la página de Red Hat, "*los proveedores de nube son empresas que proporcionan infraestructura, plataformas y software a través de una red. Generalmente los servicios se agrupan en nubes, que son conjuntos de recursos virtuales organizados como software de gestión y automatización*."

En resumen, los proveedores de Cloud Computing son las empresas que hay tras estas, que brindan todos los servicios, ofreciendo sus instalaciones y sus recursos al servicio de los clientes, encargadas del mantenimiento y el buen funcionamiento, de añadir nuevas funcionalidades y prestar buen servicio a los usuarios.

Existe una gran cantidad de proveedores de nube, y cada día surgen nuevas empresas que brindan esta característica. Pero por encima de todas ellas destacan tres: Google con **Google Cloud Platform**, Amazon con **Amazon Web Services** y Microsoft con **Microsoft Azure**.

5.2. Principales proveedores

5.2.1. Google Cloud Platform

Ilustración 3. Logo de Google Cloud Platform.

Google Cloud Platform es la plataforma de Cloud Computing que Google brinda a sus clientes. Permite la creación y desarrollo de aplicaciones y el alojamiento de estas en su plataforma gracias a una gran cantidad de programas que proporciona. Es posible desarrollar desde un simple sitio web hasta una aplicación con una complejidad alta. Ofrece servicio tanto de PaaS como de IaaS.

Google Cloud Platform es un conjunto de servicios basados en la nube con múltiples herramientas de desarrollo tales como servicios de hosting y computación, almacenamiento en la nube (Cloud Storage, Cloud Datastore y Cloud SQL), big data y APIs específicas. Todos los servicios que ofrece esta plataforma tienen una interfaz web, herramienta para la línea de comandos y una API REST.

Cuando se habla de SaaS, Google ofrece una gran cantidad de programas que ha reunido en lo que se denomina como la Nube de Google (Google Cloud). En este espacio se ofrecen servicios como un paquete completo de ofimática (Google Docs), servicios de correo electrónico (Gmail), calendario (Google Calendar), almacenamiento de datos (Google Drive) y un sinfín más.

La solución PaaS que ofrece Google Cloud Platform se denomina App Engine, una plataforma totalmente gestionada que permite crear y desplegar

aplicaciones, con la capacidad de escalarlas sin complicaciones, sin preocuparse por la infraestructura subyacente. Permite centrar únicamente en el desarrollo de la aplicación, sin la sobrecarga de la gestión. Los lenguajes que están permitidos en esta plataforma son Java, PHP, Node.js, Python, C#, .NET, Ruby y Go) y tiene soporte para múltiples frameworks. Permite crear SaaS y alojarlo en la misma plataforma, además de la posibilidad de desarrollar de manera local la aplicación, gracias al SDK disponible.

Compute Engine es la plataforma desarrollada por Google para IaaS. Permite a los clientes la creación y manejo de máquinas virtuales que se ejecutan en los centros de datos de Google.

5.2.2. Amazon Web Services

Ilustración 4. Logo de Amazon Web Services.

Amazon Web Services es la plataforma de Cloud Computing ofrecida por la empresa Amazon. Se trata de la plataforma en la nube más adoptada en el mundo que ofrece más de 175 servicios integrales de centros de datos a nivel global y una de las pioneras en esta tecnología. Posee la infraestructura en la nube más amplia del mundo con 76 zonas de disponibilidad en 24 regiones geográficas del mundo.

AWS cuenta con una gran cantidad de servicios y de características incluidas en ellos, "*ofreciendo desde tecnologías de infraestructura como cómputo, almacenamiento y bases de datos hasta tecnologías emergentes*

como aprendizaje automático e inteligencia artificial, lagos de datos y análisis e internet de las cosas." Fuente: https://aws.amazon.com/es/what-is-aws/

Dentro de AWS la plataforma que ofrece IaaS se trata de Elastic Compute Cloud (EC2). Proporciona capacidad de computación escalable en la nube, con la ventaja de evitar la inversión inicial en hardware. Permite crear entornos informáticos virtuales conocidos como instancias, configurar la seguridad y las redes y administrar el almacenamiento. Además, para las instancias, existen plantillas preconfiguradas, distintas configuraciones de CPU, memoria, almacenamiento y capacidad de red, firewall o direcciones IPv4 estáticas para informática en la nube dinámica entre otras.

Como solución a PaaS, AWS ofrece Elastic Beanstalk. A través de esta se pueden desarrollar y administrar aplicaciones en la nube de forma rápida, reduciendo la complejidad de administración y con las ventajas de no tener que aprender como funciona la infraestructura en la que se ejecutan y una gran capacidad de elección y control. De forma automática, al lanzar una aplicación en Elastic Beanstalk, la propia plataforma se encarga de crear los recursos necesarios para que su funcionamiento sea el correcto como instancias de Amazon EC2. Los lenguajes compatibles con esta plataforma son los siguientes: Go, Java, .NET, PHP, Ruby, Python y Node.js.

Existen otras soluciones dentro de Amazon Web Services para PaaS (aunque no constituyen un PaaS tradicional): AWS Cloud9, AWS CodePipeline y AWS CodeDeploy.

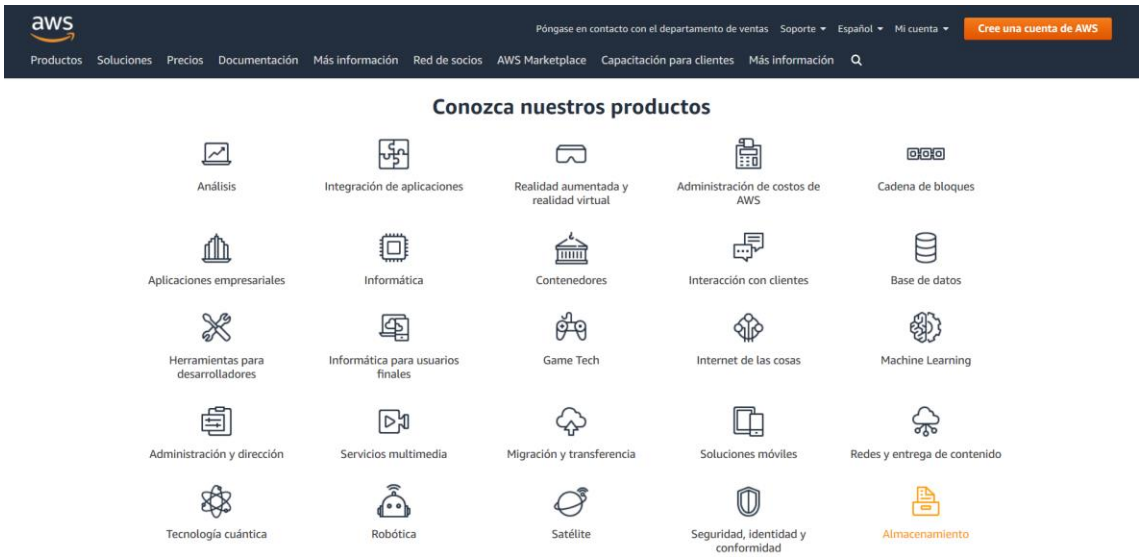


Ilustración 5. Productos ofrecidos dentro de Elastic Beanstalk. Fuente: https://aws.amazon.com/es/products/

Para las soluciones en SaaS, AWS proporciona una base segura de la que se han aprovechado empresas como Netflix.

5.2.3. Microsoft Azure

Ilustración 6. Logo de Microsoft Azure.

Microsoft Azure es la solución que ofrece la empresa Microsoft en la nube, y en su página web se define así: “Microsoft Azure es conjunto en constante expansión de servicios en la nube para ayudar a su organización a satisfacer sus necesidades comerciales. Le otorga la libertad de crear, administrar e implementar aplicaciones en una red mundial enorme con sus herramientas y marcos favoritos.”.

Anteriormente fue conocida como Windows Azure o como Azure Services Platform. Se expande a lo largo de 24 países por todo el mundo, y posee servicios propios como alojamiento para aplicaciones e incluso comunicaciones seguras.

Existen dos soluciones de PaaS en Microsoft Azure: App Service y Cloud Services. La primera de ellas, se define como una plataforma en la que se permite implementar y escalar aplicaciones web a través de lenguajes como .NET, .NET Core, Node.js, Java, PHP, Ruby o Python, que se ejecutan en contenedores. Cloud Services permite desarrollar, empaquetar e implementar aplicaciones eficaces de manera simple, con la ventaja de poseer un SDK. Los lenguajes compatibles con esta plataforma son .NET, Node.js, Java, PHP, Python y Ruby. Además, a través de Azure Emulator se puede probar la aplicación antes de implementarla en la nube.

Para IaaS, Microsoft Azure ha desarrollado la plataforma Azure Virtual Machines, una plataforma que permite la creación y configuración de máquinas virtuales que nos permitan desarrollar en ellas. Estas máquinas virtuales pueden tener hasta 416 vCPU y 12 TB de memoria.

5.2.4. Comparativa de los tres proveedores principales

	Amazon Web Services	Microsoft Azure	Google Cloud Platform
IaaS	Elastic Compute Cloud (EC2)	Azure Virtual Machines	Compute Engine
PaaS	Elastic Beanstalk	App Service y Cloud Service	App Engine Standard Environment y App Engine Flexible Environment
Lenguajes permitidos	Go, Java, .NET, PHP, Ruby, Python y Node.js	.NET, .NET Core, Node.js, Java, PHP, Ruby, Python	Java, PHP, Node.js, Python, C#, .NET, Ruby y Go
Servidores Virtuales Privados	Lightsail	Virtual Machine Images	-
Servicio de contenedores	EC2 Container Service (ECS)	-	-
Servicio de Kubernetes	Elastic Container Service para Kubernetes (EKS)	Azure Kubernetes Service (AKS)	Kubernetes Engine
Registro de contenedores Docker	EC2 Container Registry (ECR)	Azure Container Registry	Container Registry
Contenedores Serveless	Fargate	Container Instances	-
Microservicios	Service Fabric	App Engine	-
Serveless	Lambda	Functions	Cloud Functions
Bases de datos relacionales	DynamoDB	Azure Cosmos DB	Cloud Datastore y Cloud Bigtable
NoSQL	DynamoDB y SimpleDB	Table Storage	Cloud Datastore
Almacenamiento	Simple Storage Services (S3)	Blolb Sorage	Google Cloud Storage
Entornos virtuales de red	Virutal Private Cloud	Virtual Network	Virtual Private Cloud
DNS	Route 53	Azure DNS	Google Cloud DNS
Administración	Application Discovery Service y System Manager	Log Analytics, Operations Management Suite y Storage Explorer	Cloud Console
Autenticación y autorización	Identity and Access Management (IAM) y Organizations	Active Directory (y versión Premium)	Cloud IAM y Cloud Identity-Aware Proxy
Cifrado	AWS Key Management Service y Cloud HSM	Key Vault	Cloud Key Management Service

Tabla 3. Comparativa de los servicios ofrecidos por los tres principales proveedores de la nube

5.2.5. Otros proveedores destacados

Existen una gran cantidad de proveedores en la nube, pero entre todos ellos vamos a hablar de algunos destacados, que aunque no están al nivel de los tres principales, tienen un buen rendimiento y destacan por diversos factores.

SAP Cloud Platform

Ilustración 7. Logo de SAP.

Es la solución que ofrece SAP en el mundo del Cloud Computing. Se trata de una Plataforma como servicio (PaaS), con todas las ventajas que ofrece tener esta tecnología.

Heroku

Ilustración 8. Logo de Heroku.

Es propiedad de Salesforce, y se trata de una solución de PaaS capaz de soportar distintos lenguajes de programación. Fue desarrollada en 2007, para

el lenguaje Ruby, pero con el tiempo se ha ido ampliando con nuevos lenguajes como Java, Node.js, Scala, Clojure, PHP y Python.

Permite crear aplicaciones de forma sencilla y estas se corren desde un servidor del propio Heroku usando Heroku DNS Server para apuntar al dominio de la aplicación (suele ser *nombreaplicacion.herokuapp.com*). Esto se hace a través de Heroku Platform, donde las aplicaciones son ejecutadas en contenedores virtuales.

OpenShift

Ilustración 9. Logo de OpenShift.

Pertenece a Red Hat, y se define así: “Red Hat OpenShift es una plataforma de contenedores de Kubernetes empresarial con operaciones automatizadas integrales para gestionar implementaciones de nube híbrida y multicloud”. Por lo tanto se puede hablar que a través de esta plataforma se ofrecen soluciones de tipo CaaS (Contenedores como servicio).

IBM Cloud

Ilustración 10. Logo de IBM Cloud.

Se trata de la plataforma que ofrece el servicio de Cloud Computing de la empresa IBM. Esta solución puede acercarse al nivel de los 3 principales proveedores, ya que ofrece características similares. En esta plataforma, IOBM combina PaaS con IaaS, con una red de centros de datos que se extiende por todo el mundo. IBM Cloud permite desplegar aplicaciones nativas a la vez que garantiza la portabilidad de la carga del trabajo.

6. CLOUD COMPUTING FRENTE A OTROS TIPO DE COMPUTACIÓN

Cloud Computing no es la única forma de computación existente, es por ello, que en este capítulo se van a analizar las demás formas de computación en comparación con la nube.

6.1. Computación tradicional

Con computación tradicional hacemos referencia a una computación de manera local, en la que todos nuestros datos se encuentran en un disco duro de un ordenador o en un servidor. Depende mucho del hardware del equipo, y si este es potente, podremos obtener una gran eficiencia, aun así, el software hay que pagarlo y conlleva un proceso de instalación, que en algunos casos es complejo y realizarlo puede ocasionar muchos problemas. Además la información solo es accesible desde el mismo equipo, con lo que se corre el riesgo de perder la información en caso de que falle el equipo.

Esta forma de computación día tras día va quedando obsoleta frente a la computación en la nube.

6.2. Egde Computing

Es una forma de computación que se relaciona mucho con el Internet de las Cosas (IoT, por sus siglas en inglés *Internet of Things*). En la actualidad, la forma en la que actúa IoT es la siguiente: los dispositivos captan la información y la envían a la nube, donde es procesada para un determinado fin.

Egde Computing tiene el propósito de que sean los propios dispositivos al borde de la red los que procesan la información, sin tener que enviarlos a la nube, reduciendo las comunicaciones existentes, y de esta forma acelerar el funcionamiento del IoT. Esta tecnología es muy beneficiosa en distintos campos en los que necesitan procesar datos en tiempo real, tales como la fabricación, la salud, las telecomunicaciones...

6.3. Fog Computing

En español, se define como “Computación en la niebla”, un paso intermedio entre la nube y el Edge Computing. No son los propios dispositivos que captan la información los que la procesan, sino que existe un nodo intermedio o varios, que reciben la información de estos y la procesan. La cantidad de información que procesan es mucho menor que la que se procesaría en la nube, ya que los dispositivos de Fog Computing se hacen cargo de procesar la información de un número reducido de los dispositivos en la frontera, intentando obtener un equilibrio, para no sobrecargarlos, ya que de esta forma, los beneficios no se reflejarían.

Es una tecnología que fundamentalmente es usada en el Internet de las Cosas.

No es necesario que toda la información sea enviada a los dispositivos de la niebla, en ocasiones, solo parte de la información será procesada por dispositivos de la niebla, y habrá otra que necesariamente tenga que ser procesada por la nube, pero está claro que son tecnologías que se complementan bien y se benefician mutuamente.

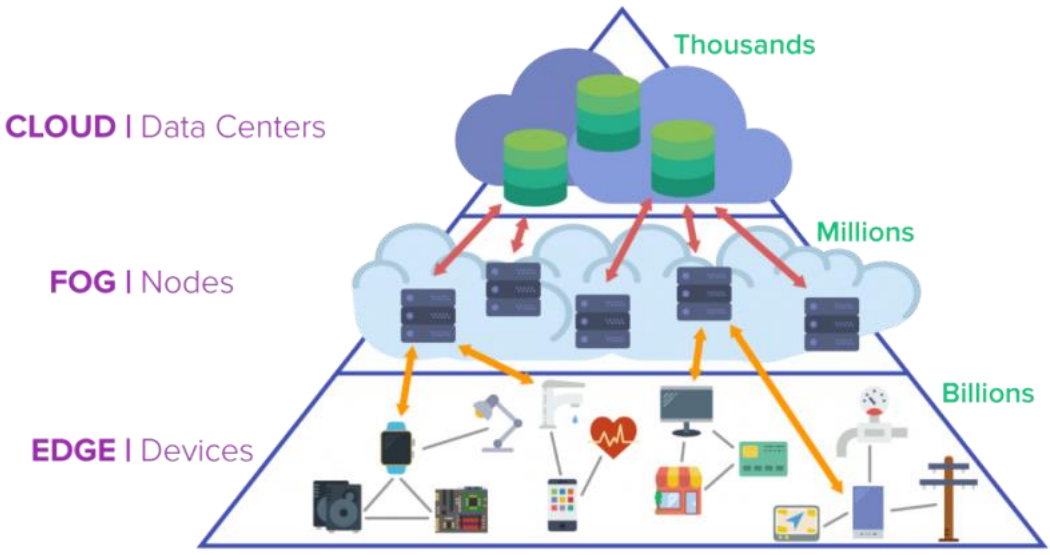


Ilustración 11. Cloud vs Fog vs Edge.

Fuente: https://itmastersmag.com/noticias-analisis/que-es-edge-computing-y-por-que-es-relevante-para-las-empresas/

6.4. Grid Computing

Se trata de una tecnología que permite coordinar distintos ordenadores a través de hardware y software para realizar una tarea común, que generalmente posee un gran nivel de complejidad. Es decir, un equipo que realiza una computación grid, se encuentra dentro de una red de trabajo y realiza operaciones dentro de ella.

En muchas ocasiones se puede confundir con el Cloud Computing por el hecho de que ambos están basados en la tecnología red, donde intercambian recursos y tienen capacidad de multitarea. Pero las diferencias son de gran dimensión, ya que el Cloud Computing tiene una gran elasticidad frente a un servicio rígido del Grid Computing, la seguridad no es el punto fuerte del Grid, algo muy necesario en la tecnología de la nube, además de que la gestión de los recursos en el caso de la nube suele ser centralizada frente a una gestión de recursos distribuida en el Grid Computing, entre otras muchas diferencias.

7. TWSENT: CASO PRÁCTICO. DESPLIEGUE DE UNA APLICACIÓN WEB EN LA NUBE SIGUIENDO EL MODELO PAAS.

Para demostrar de forma práctica como funciona la nube y cómo se realiza una implementación de una aplicación en la nube, vamos a realizar un caso práctico de una aplicación web en PaaS.

Esta aplicación web se denomina TwSent y se trata de una aplicación que permite realizar en tiempo real un análisis de sentimientos en Twitter a través de un hashtag. Se trata de una aplicación de una complejidad media, ya que su función principal es mostrar cómo se realiza el desarrollo de una aplicación web en PaaS.

7.1. Resumen

TwSent, como bien se ha mencionado arriba, realiza un análisis de sentimientos en Twitter en tiempo real a través de un hashtag. A continuación, se explicará el modo en el que funciona.

Un usuario accede a la aplicación, rellena un formulario con el hashtag que quiere que se analice, el número de tweets que quiere que se recolecten cada cierto tiempo, el pasaje “Minutos” que hace referencia al número de minutos que el usuario esperará para recolectar y analizar nuevos tweets, y la granularidad, que mostrará los resultados de una manera más o menos precisa a través de emoticonos.

Lo que hace la aplicación en sí, es recolectar los últimos tweets que se han publicado con un hashtag, los analiza y genera la media de la polaridad de todos ellos, guardando el más positivo y el más negativo, para mostarlos.

Conocemos por polaridad en este contexto a la medida, comprendida entre -1 y 1, resultante del análisis de sentimientos realizado a un tweet. Siendo -1 la marca más negativa, 0 el valor neutro y 1 la más positiva.

Una vez muestra los resultados, esta aplicación sigue ejecutándose, esperando un cierto tiempo (indicado por el usuario), para recolectar nuevos

tweets y actualizar la información mostrada, generando un análisis en tiempo real, ya que siempre que recolecta los tweets, recoge los últimos que se han publicado.

Esta aplicación estará ejecutándose hasta que el usuario decida pararla, tratándose de una aplicación muy oportuna para tenerla de segundo plano en una oficina o en un lugar en el que aúne a muchas personas y se esté tratando un tema concreto, ya que el análisis que se esté realizando puede servir de apoyo.

El enlace a la aplicación es el siguiente:

http://twsent.azurewebsites.net/

7.2. Análisis de la Aplicación

7.2.1. Perfiles de usuario

En primer lugar, se definirán los perfiles de usuario existentes para crear una idea general del funcionamiento de la aplicación.

- **Usuario**

En este proyecto es el único participante, y se trata de cualquier usuario que acceda a la aplicación. Tiene la capacidad de establecer los parámetros de la ejecución, ejecutar y ver los resultados.

7.2.2. Casos de uso

Una vez definidos los perfiles de usuario, vamos a definir los casos de uso. Un caso de uso se trata de la descripción de una acción o actividad. En este proyecto son los siguientes:

Actor	**Casos de Uso**	**Descripción**
como	*quiero*	*por lo tanto*
Usuario	establecer parámetros	puedo establecer el hashtag, la granularidad, el número de tweets y los minutos
Usuario	ejecutar	pulsar el botón para ejecutar la aplicación una vez rellenos los parámetros
Usuario	visualizar resultados	ver los resultados una vez se han analizado los tweets
Usuario	ver y acceder a los tweets más influyentes	ver y pinchar en el tweet más positivo y en el tweet más negativo
Usuario	para ejecución	pulsar el botón para parar una vez que se desee

Tabla 3. Casos de Uso.

7.2.3. Diagramas de casos de uso

Antes de mostrar los diagramas de casos de uso, es necesaria su definición. Según la Wikipiedia, *"Los diagramas de casos de uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas. O lo que es igual, un diagrama que muestra la relación entre los actores y los casos de uso en un sistema."* Es decir, muestran los casos de uso del Sistema asociados a los actores presentes en el mismo.

- **Diagrama de casos de uso General**

En este punto se puede observar el diagrama de casos de uso general, en el cual se muestran todos los actores que intervienen (en este caso solo uno, el actor Usuario) y todas las funcionalidades a las que pueden acceder cada uno de ellos.

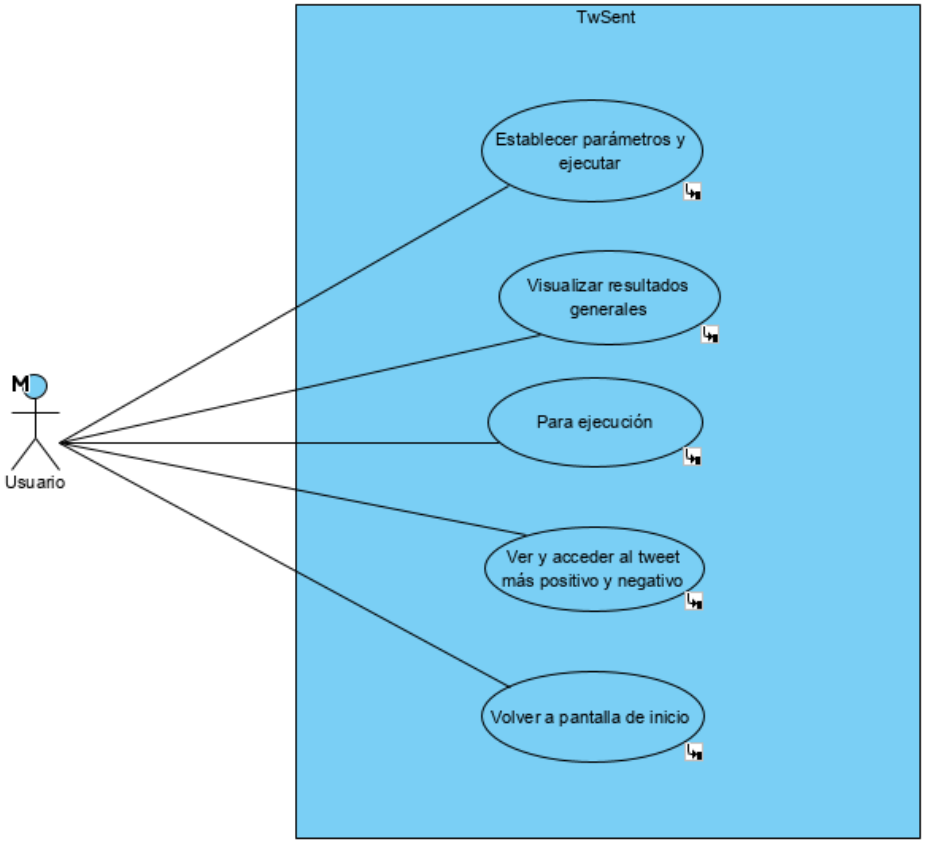


Ilustración 12. Diagrama de casos de Uso General.

- **Diagrama de casos de uso "Establecer parámetros y ejecutar"**

Este diagrama de casos de uso se trata del primero asociado a una funcionalidad: "Establecer parámetros y ejecutar". Participa el actor Usuario, al igual que en todos los siguientes. Se establece paso a paso lo que el Usuario puede hacer, desde el primer paso "Entrar en la aplicación" hasta el último "Ejecutar", pasando por el relleno de parámetros como "Rellenar hashtag". Se establece además, que en el caso de saltar un paso y no rellenar un parámetro, al momento de ejecutar, se vuelve al punto del parámetros sin rellenar, para que se complete y se pueda seguir el flujo normal.

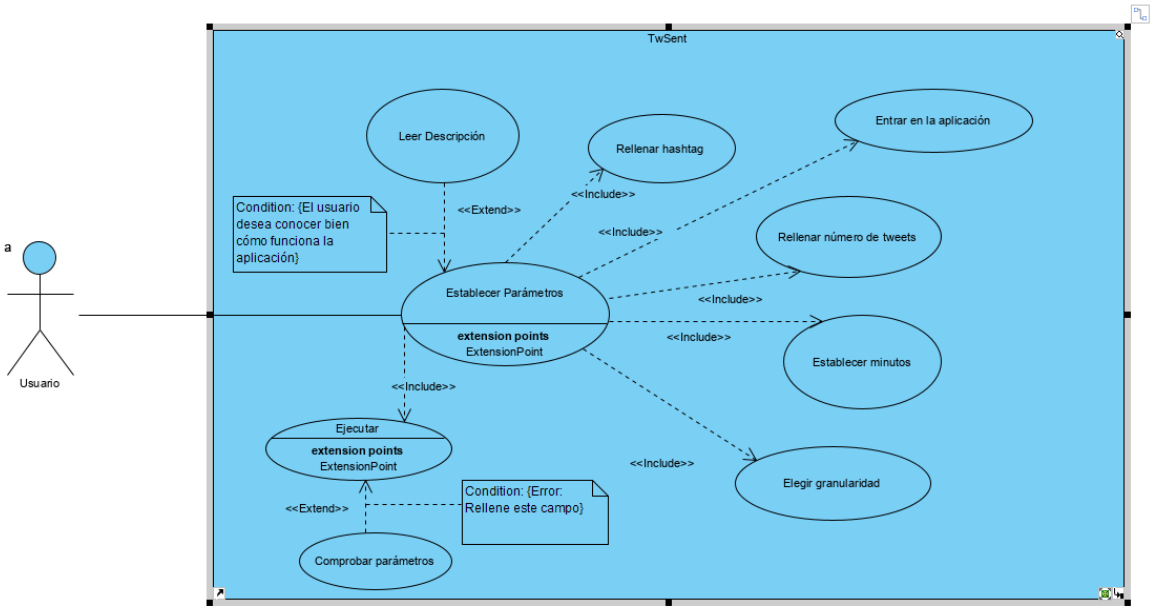


Ilustración 13. Diagrama de casos de Uso "Establecer parámetros y ejecutar".

CASO DE USO	Establecer parámetros y ejecutar
Actor primario	Usuario
Sistema	TwSent
Participantes	Usuario
Nivel	Objetivo Usuario
Precondiciones	Acceder a la aplicación
Flujo básico	1. Entrar en la aplicación
	2. Rellenar hashtag
	3. Rellenar número de tweets
	4. Establecer minutos
	5. Elegir granularidad
	6. Ejecutar
Flujos alternativos	1A. Si el usuario lo desea leer descripción
	6B. Si falta algún parámetros volver a este y seguir el proceso (volver a 2,3,4 o 5) #TODO dividir cada uno

Tabla 4. Caso de Uso "Establecer parámetros y ejecutar"

- **Diagrama de casos de uso "Visualizar resultados generales"**

En la siguiente imagen se muestra la funcionalidad "Visualizar resultados generales".

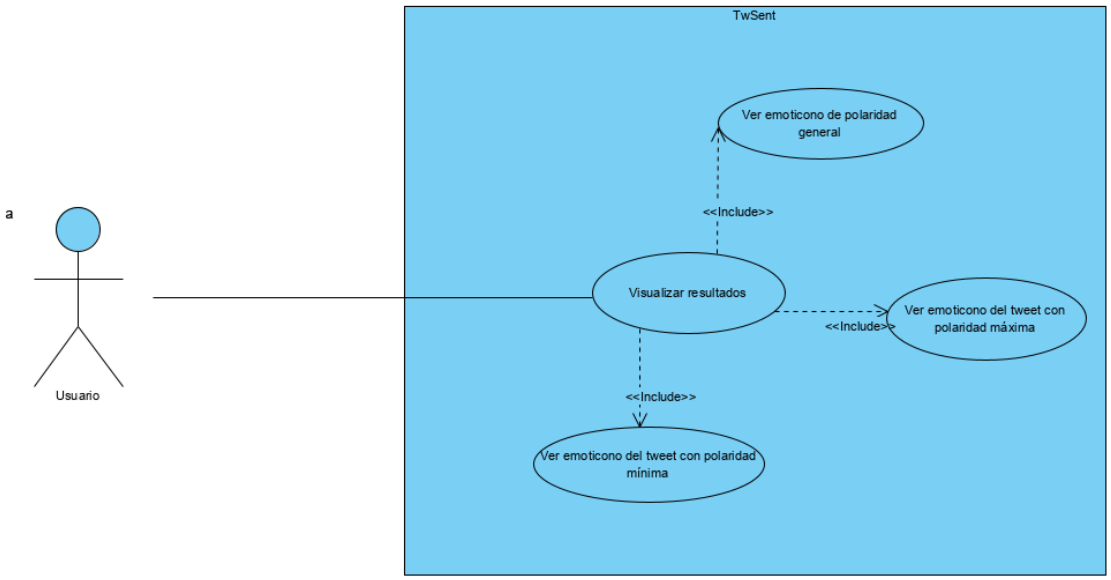


Ilustración 14. Diagrama de casos de Uso "Visualizar resultados generales".

CASO DE USO	Visualizar resultados generales
Actor primario	Usuario
Sistema	TwSent
Participantes	Usuario
Nivel	Objetivo Usuario
Precondiciones	Ejecutar la aplicación y esperar a obtener los primeros resultados
Flujo básico	1. Ver emoticono de polaridad general
	2. Ver emoticono del tweet con polaridad máxima
	3. Ver emoticono del tweet con polaridad mínima

Tabla 5. Caso de Uso "Visualizar resultados generales"

- **Parar ejecución**

En esta imagen se muestra la funcionalidad “Parar ejecución”, mediante la cual, el usuario tras haber visualizado resultados, puede pinchar en el botón que permite parar la ejecución, y de esta forma congelar la imagen, impidiendo que los emoticonos se refresquen.

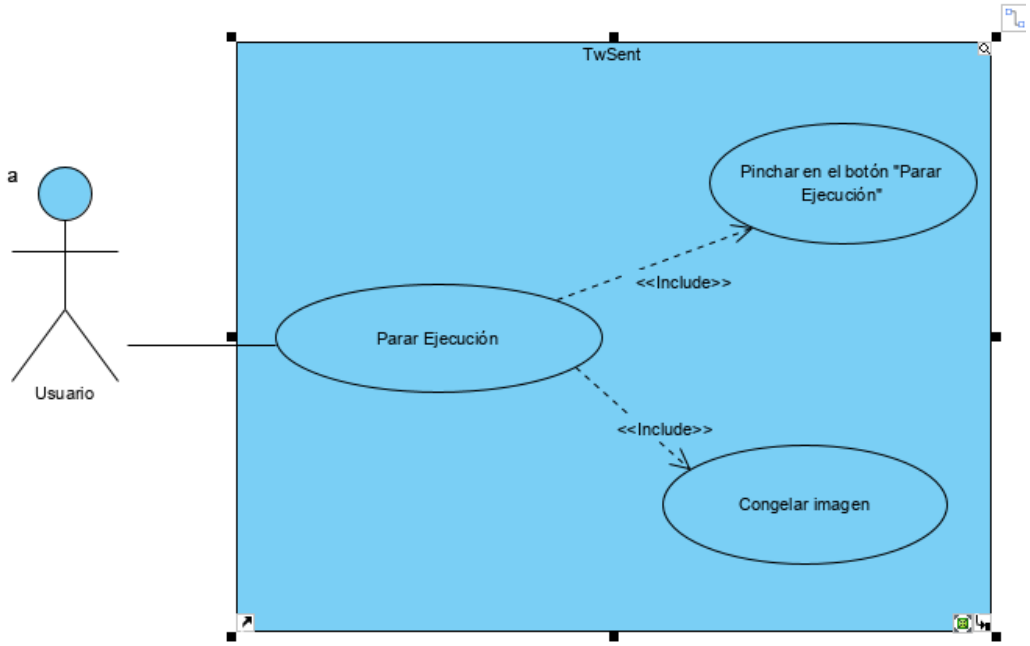


Ilustración 15. Diagrama de casos de Uso “Parar ejecución”.

CASO DE USO	Parar ejecución
Actor primario	Usuario
Sistema	TwSent
Participantes	Usuario
Nivel	Objetivo Usuario
Precondiciones	Ejecutar la aplicación y haber visualizado resultados
Flujo básico	1. Pinchar en el botón “Parar Ejecución”
	2. Congelar imagen

Tabla 7. Caso de Uso “Parar ejecución”

- **Ver y acceder al tweet más positivo y negativo**

En este diagrama de caso de uso se muestra al funcionalidad “Ver y acceder al tweet más positivo y negativo”. Esta permite al usuario, una vez visualizados los resultados, pinchar en el propio tweet mostrado, ya sea el más positivo o el más negativo de los seleccionados, y acceder a Twitter, donde podrá obtener más información del tweet, con lo que puede retroalimentarse y ponerse en situación de lo que está sucediendo en la actualidad.

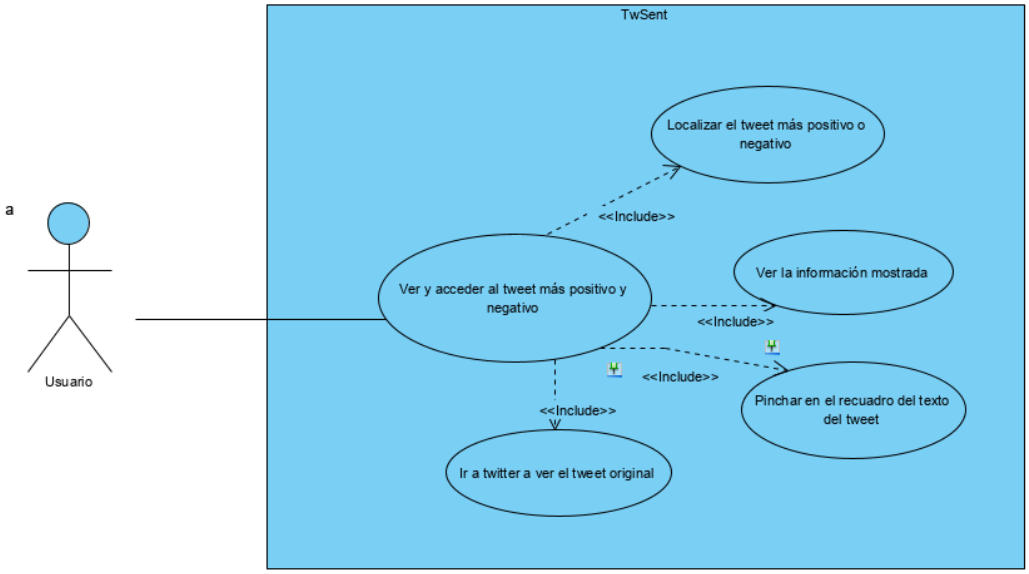


Ilustración 16. Diagrama de casos de Uso “Ver y acceder al tweet más positivo y negativo”.

CASO DE USO	Ver y acceder al tweet más positivo y negativo
Actor primario	Usuario
Sistema	TwSent
Participantes	Usuario
Nivel	Objetivo Usuario
Precondiciones	Visualizar resultados
Flujo básico	1. Localizar el tweet más positivo o negativo
	2. Ver la información mostrada
	3. Pincha en el recuadro del texto del tweet
	4. Ir a Twitter a ver el tweet original

Tabla 8. Caso de Uso “Ver y acceder al tweet más positivo y negativo”

- **Volver a pantalla de inicio**

A través de esta imagen de la funcionalidad “Volver a pantalla de inicio”, el usuario, una vez ejecutada la aplicación, puede volver a la pantalla de inicio, donde puede establecer nuevos parámetros para una nueva ejecución.

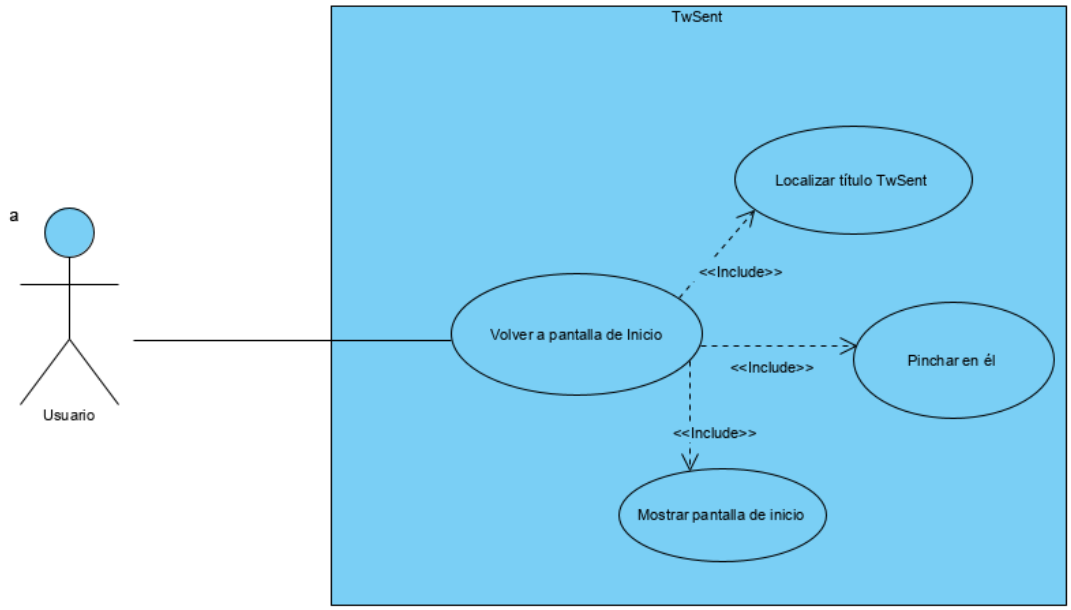


Ilustración 17. Diagrama de casos de Uso “Volver a pantalla de inicio”.

CASO DE USO	Volver a pantalla de inicio
Actor primario	Usuario
Sistema	TwSent
Participantes	Usuario
Nivel	Objetivo usuario
Precondiciones	Ejecutar la aplicación
Flujo básico	1. Localizar título TwSent
	2. Pinchar en él
	3. Mostrar pantalla de inicio

Tabla 9. Caso de Uso “Volver a pantalla de inicio”

7.2.4. Diagramas de secuencia

Para poder entender este punto, es necesario definir qué es un diagrama de secuencia. Se trata de un tipo de diagrama de interacción que pretende describir cómo se comporta de manera dinámica el sistema de información, centrándose en los mensajes que se intercambian entre los objetos.

- **Establecer parámetros y ejecutar**

Aquí se muestran los mensajes y el orden en el que se envían para que un usuario pueda establecer los parámetros y ejecutar la aplicación.

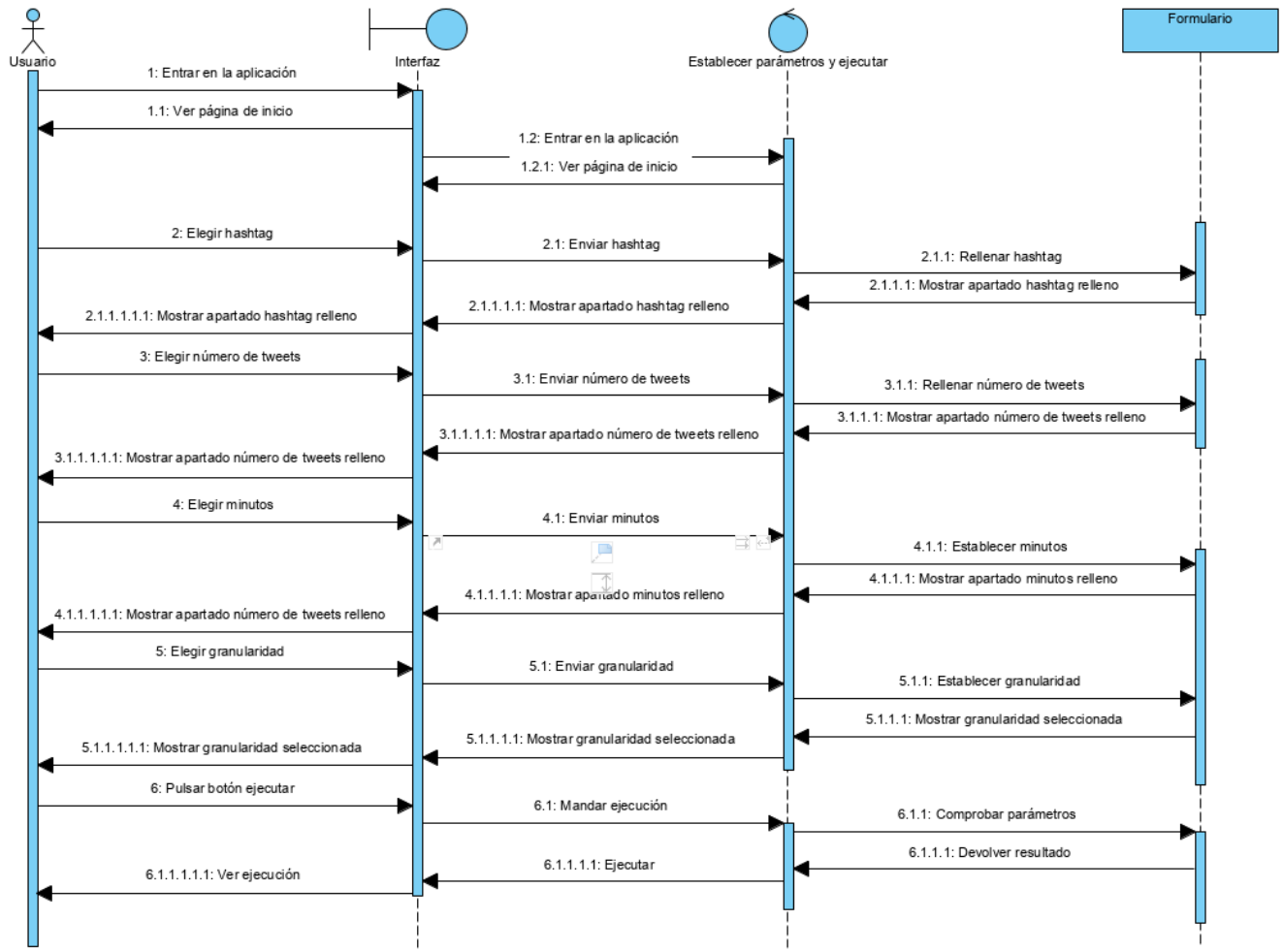


Ilustración 18. Diagrama de secuencia "Establecer parámetros y ejecutar".

- **Visualizar resultados generales**

  En este diagrama se muestran los mensajes y la secuencia para que un usuario pueda visualizar los resultados generales de una ejecución.

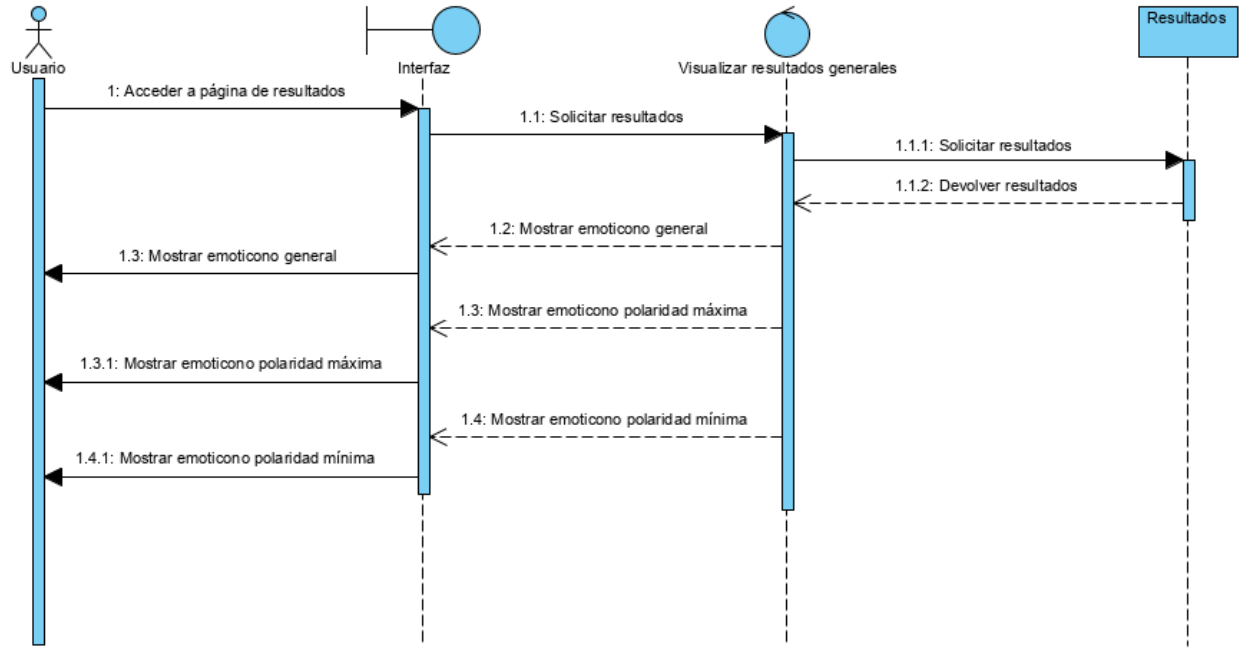


Ilustración 19. Diagrama de secuencia "Visualizar resultados generales".

- **Parar ejecución**

  Este diagrama de secuencia muestra los mesnajes intercambiados y la secuencia de estos para poder parar la ejecución de la aplicación.

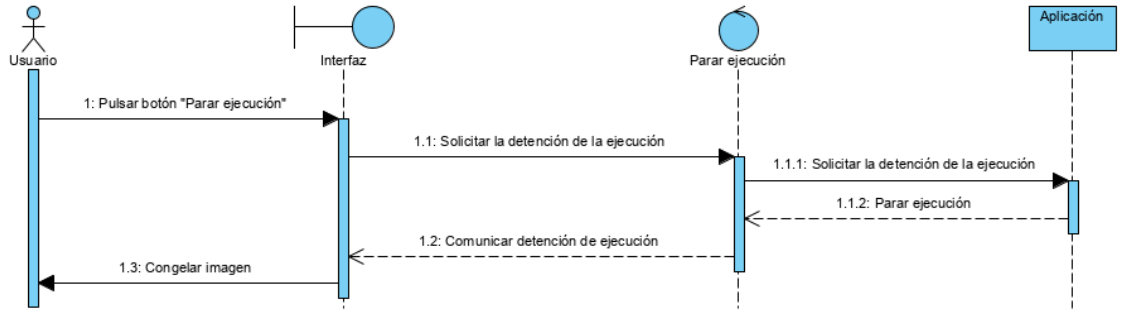


Ilustración 20. Diagrama de secuencia "Parar ejecución".

- **Ver y acceder al tweet más positivo y negativo**

  En la siguiente imagen se pueden visualizar los mensajes que debe haber para poder ver y acceder al tweet más positivo y negativo de una ejecución.

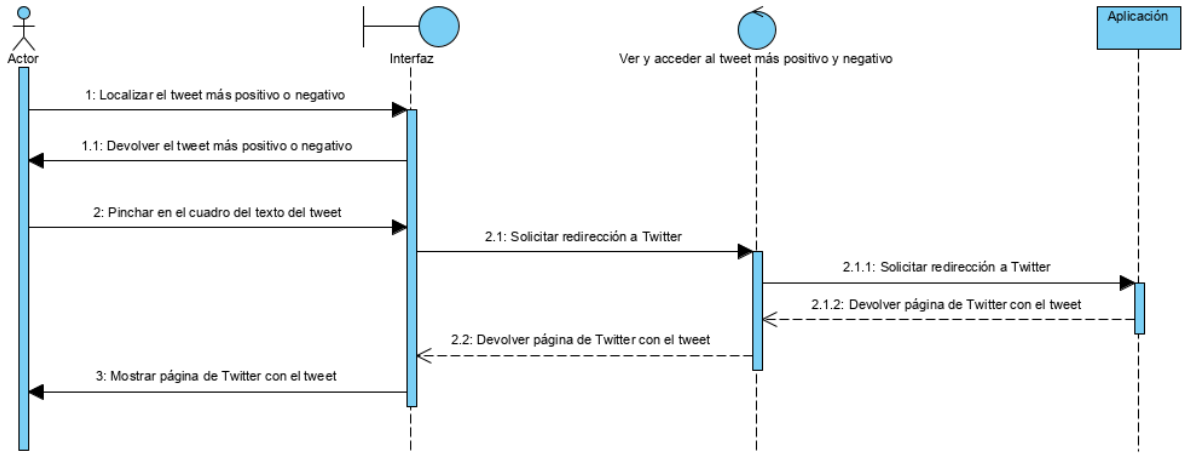


Ilustración 21. Diagrama de secuencia "Ver y acceder al tweet más positivo y negativo".

- **Volver a pantalla de inicio**

  Por ultimo, en este diagrama de secuencia se pueden apreciar los mensajes y la secuencia necesarios para poder volver a la pantalla de inicio.

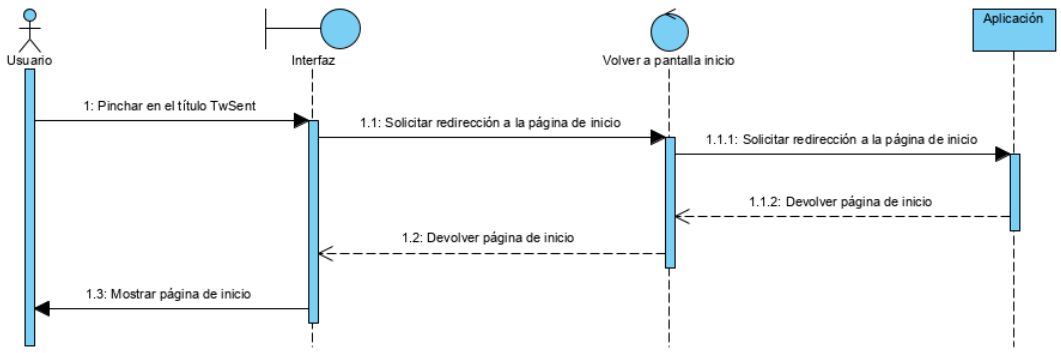


Ilustración 22. Diagrama de secuencia "Volver a pantalla de inicio".

7.3. Diseño de la Aplicación

En este punto del proceso de Ingeniería del Software, nos encargaremos de analizar las características más resultantes de la aplicación.

7.3.1. Diseño arquitectónico

En esta imagen se puede apreciar el diseño arquitectónico de nuestra aplicación. Se entiende como diseño arquitectónico a la concepción original de la Arquitectura Software de un sistema a fin de construirlo con la máxima claidad y claridad, intentando obtener un alto grado de abstracción que será refinado sucesivamente hasta nivel de componente.

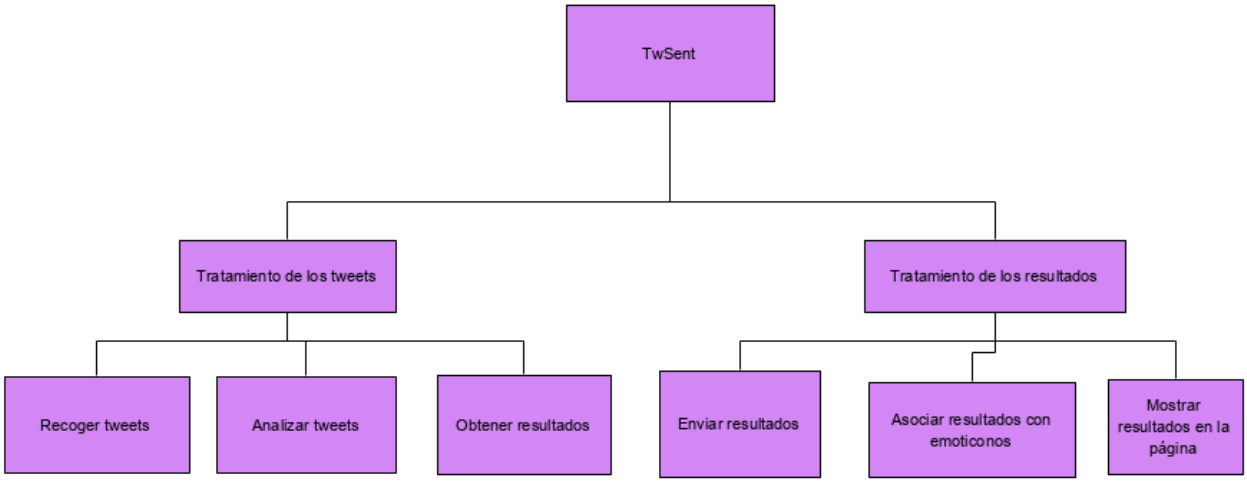


Ilustración 23. Diseño arquitéctonico de TwSent.

7.3.2. Modelo de datos

Una vez analizado el problema, y propuesta una solución de desarrollo, es necesario realizar un modelo de datos. Un modelo de datos es un tipo de modelo que determina la estructura lógica de una base de datos. En nuestro caso, el uso de una base de datos es mínimo, tratándose además de un tipo de base de datos no relacional (NoSQL).

Únicamente en esta base de datos se almacenará información de cada ejecución, es decir, por cada vez que el usuario pulse el botón de ejecutar, se creará una nueva rama dentro de la estructura creada en la base de datos, ya que la estructura que presenta es en forma de árbol.

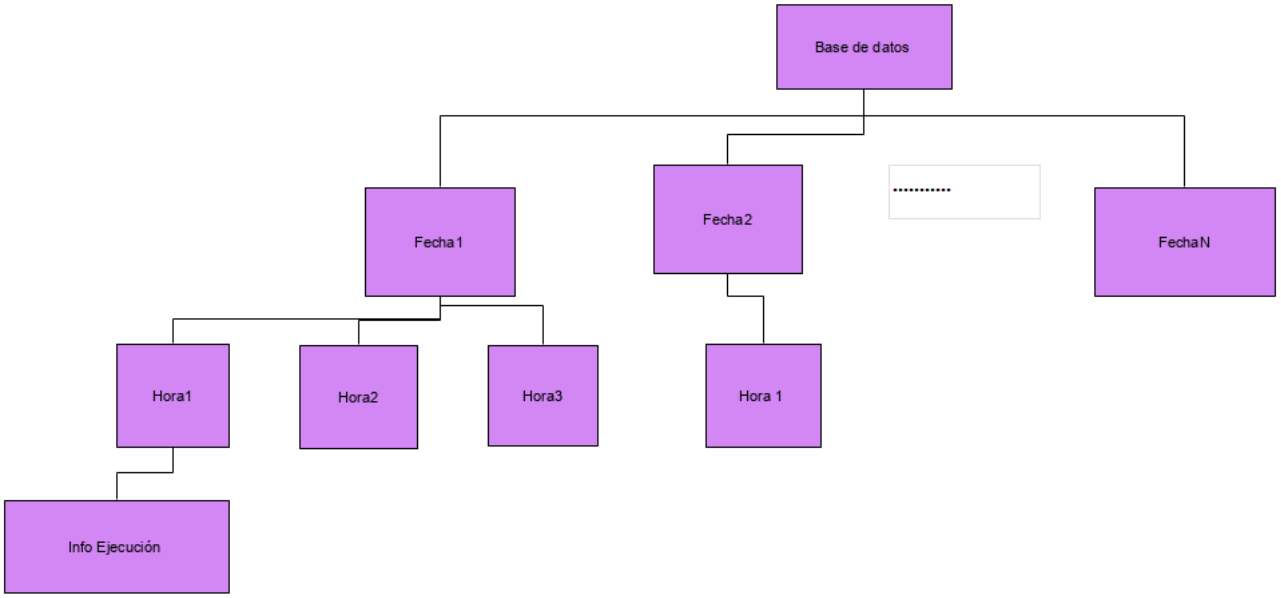


Ilustración 24. Modelo de datos de TwSent.

En el nivel superior se encuentra la fecha, y por cada ejecución que se realiza en el mismo día, se agrega una nueva entrada a ese día.

Esta entrada, en el nivel inferior, tiene como clave la hora, con los minutos y segundos, por lo que cada entrada va a ser única.

Y cada entrada que se crea, con la fecha y el día como identificadores únicos, almacena los siguientes datos:

- **hashtag:** en este campo se almacena el hashtag con el que se realizará el análisis.
- **num-tweets:** en este campo se almacena el número de tweets que se van a recolectar nuevos para realizar el análisis.
- **minutos:** en este campo se almacena el número de minutos que hay que esperar para realizar una nueva recolecta y análisis de tweets.
- **granularidad:** en este campo se almacena la granularidad seleccionada a la hora de mostrar los resultados.
- **pos-usuario:** en este campo se almacena el usuario que ha puesto el tweet más positivo hasta el momento.
- **pos-texto:** en este campo se almacena el texto del tweet más positivo hasta el momento.
- **pos-id:** se trata del ID del tweet más positivo hasta el momento.
- **pos-fecha:** en este campo se almacena la fecha del tweet más positivo hasta el momento.
- **pos-polaridad:** aquí se almacena la polaridad del tweet más positivo hasta el momento.
- **neg-usuario:** en este campo se almacena el usuario que ha publicado el tweet más negativo hasta el momento.
- **neg-texto:** en este campo se almacena el texto del tweet más negativo hasta el momento.
- **neg-id:** se trata del ID del tweet más negativo hasta el momento.
- **neg-fecha:** en este campo se almacena la fecha del tweet más negativo hasta el momento.
- **neg-polaridad:** aquí se almacena la polaridad del tweet más negativo hasta el momento.

En casi todos los apartados como final aparece “hasta el momento”, y eso quiere decir que no son datos que vayan a estar fijos, es decir, en el caso de que durante la primera elección de los tweets, el tweet más positivo contenga una polaridad de 0.5, y en siguientes recolecciones de tweets, exista alguno

con una polaridad mayor a 0.5, todos los datos del tweet más positivo serán actualizados con la información de este último tweet. Cuando la ejecución termine, los datos que se mantendrán en la base de datos serán los del tweet más positivo y los del tweet más negativo de esa ejecución en particular.

Cabe destacar, que al tratarse de una base de datos no relacional, no es necesario que cada campo sea de un tipo y de una longitud concreta. En cada campo se puede escribir y almacenar texto plano, y de hecho, en un mismo campo, en distintas entradas, un valor puede ser numérico y en otro puede ser un string, ya que todos van a ser tratados como cadenas de texto.

Esta es la forma en la que se vería en la interfaz de Firebase:

Ilustración 25. Datos almacenados en la interfaz de Firebase.

7.4. Diseño de la interfaz y storyboards

En este punto vamos a hablar de la interfaz de la aplicación y a analizar su diseño. Esta aplicación posee dos vistas, a las cuales se ha dado forma a través de storyboards.

7.4.1. Storyborad Vista Home

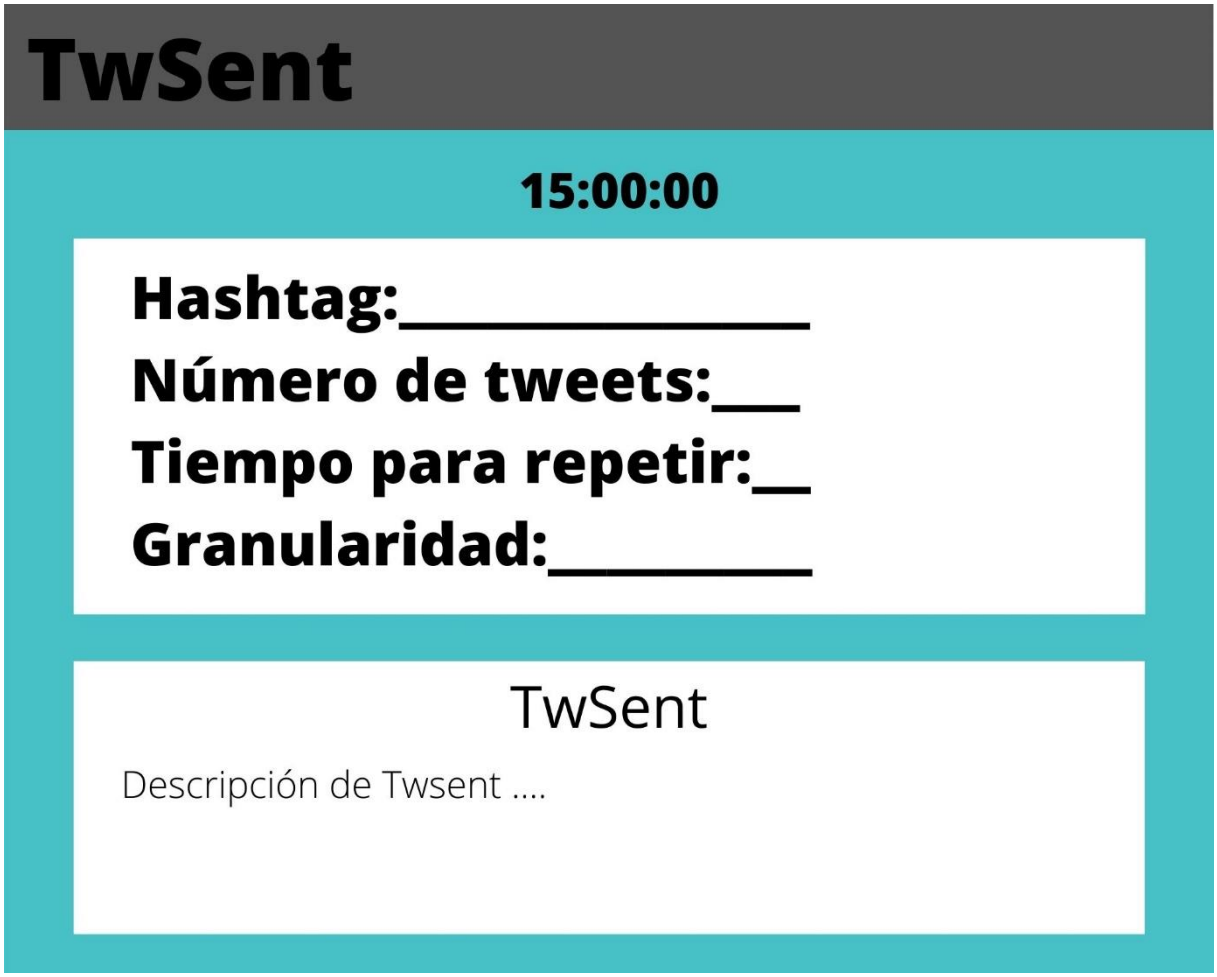


Ilustración 26. Storyboard Vista Home.

En primer lugar, temenos la vista denominada “home”, cuyo enlace es “/”. Se trara de la página principal o de inicio, la cual incluye los siguientes elementos:

En primer lugar, en la barra de navegación se incluye el título de la aplicación.

Más abajo, se encuentra un reloj, y justo debajo, una sección con fondo en blanco, en la cual se incluirá el formulario a rellenar por el usuario.

Por ultimo, en otra sección con fondo blanco, una descripción de la aplicación, para que el usuario pueda saber qué hace TwSent y cómo funciona.

7.4.2. Storyboard Vista Hashtag

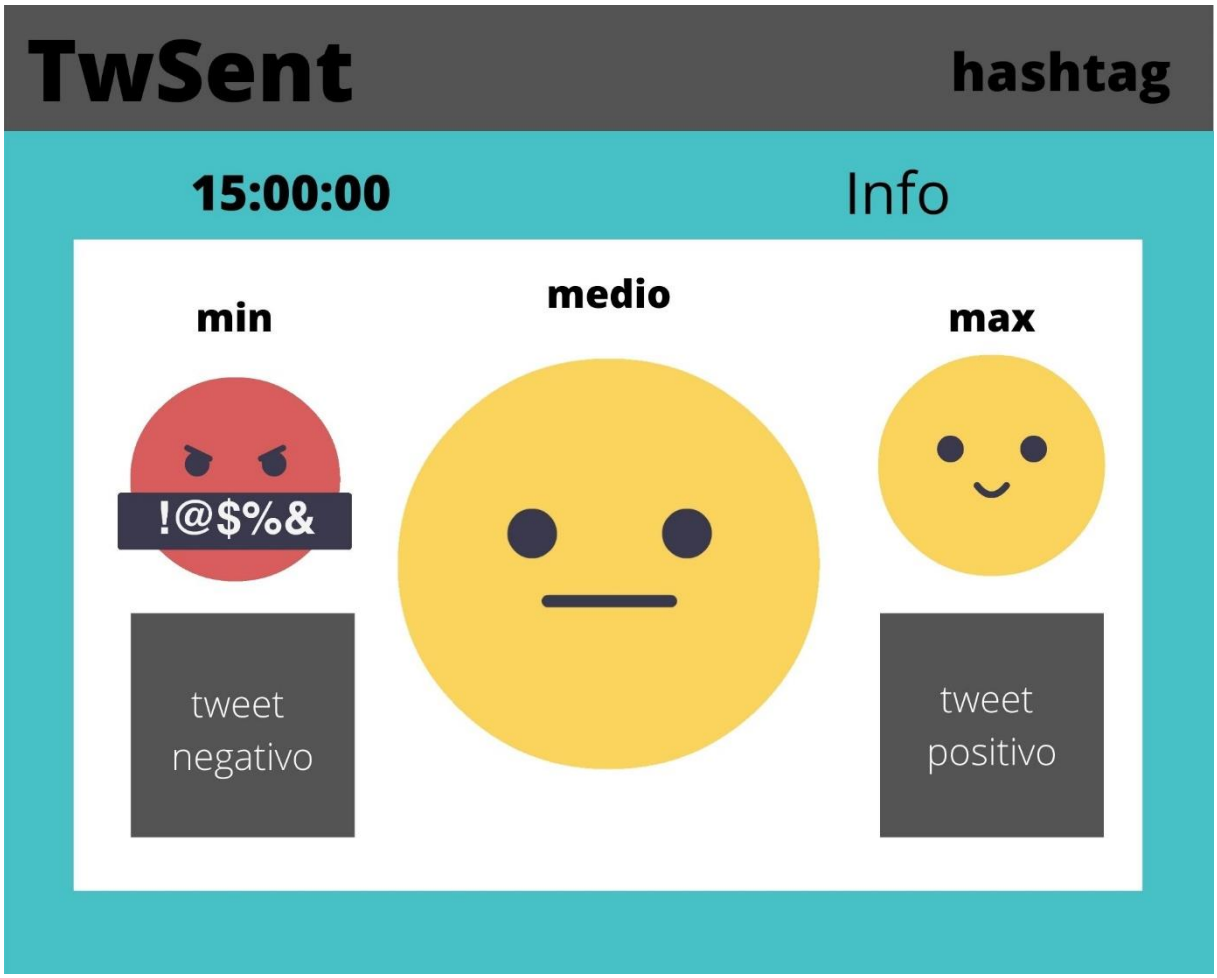


Ilustración 27. Storyboard Vista Home.

Como segunda vista, se encuentra la vista denominada “hashtag”, cuya ruta es “/hashtag”. En esta vista se aprecian los resultados de la ejecución.

Esta vista muestra la información sobre la ejecución, en primer lugar en la barra de navegación (donde muestra el hashtag), y a continuación a la altura del reloj (donde muestra el resto de información).

Justo debajo, muestra los resultados, en una sección con fondo blanco. Aquí se muestran tres emoticonos, el central, que representa la media de todos los tweets analizados, el de la izquierda que muestra el tweet más negativo, junto con un recuadro en el que se cita el tweet, y a la derecha, donde se muestra el emoticono asociado al tweet más positivo, junto con un recuadro donde se cita el tweet más positivo.

7.5. Herramientas y tecnologías existentes

Las tecnologías existentes para poder realizar una aplicación web que realice un análisis de sentimientos son las siguientes.

- **Herramientas de la web (HTML5, CSS y JavaScript)**

En primer lugar, nos vamos a centrar en qué es la aplicación, y para ello nos quedamos con que se trata de una aplicación web, y por ello, estas tres tecnologías están presentes (en este caso, sobre todo HTML5 y CSS).

Ilustración 29. Logo de HTML5.

HTML5 (HyperText Markup Language, versión 5) es la quinta revisión importante del lenguaje básico de la World Wide Web, HTML. HTML5 especifica dos variantes de sintaxis para HTML: una «clásica», HTML (text/html), conocida como HTML5, y una variante XHTML conocida como sintaxis XHTML 5 que deberá servirse con sintaxis XML (application/xhtml+xml). Esta es la primera vez que HTML y XHTML se han desarrollado en paralelo. La versión definitiva de la quinta revisión del estándar se publicó en octubre de 2014. Definición de Wikipedia: https://es.wikipedia.org/wiki/HTML5

Ilustración 28. Logo de CSS.

CSS (siglas en inglés de Cascading Style Sheets), en español «Hojas de estilo en cascada», es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado.2 Es muy usado para establecer el diseño visual de los documentos web, e interfaces de usuario escritas en HTML o XHTML; el lenguaje puede ser aplicado a cualquier documento XML, incluyendo XHTML, SVG, XUL, RSS, etcétera. Definición de Wikipedia: https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada

Ilustración 30. Logo de JavaScript.

JavaScript (abreviado comúnmente JS) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Definición de Wikipedia: https://es.wikipedia.org/wiki/JavaScript

- **Lenguaje de Back-end**

  - **Python**

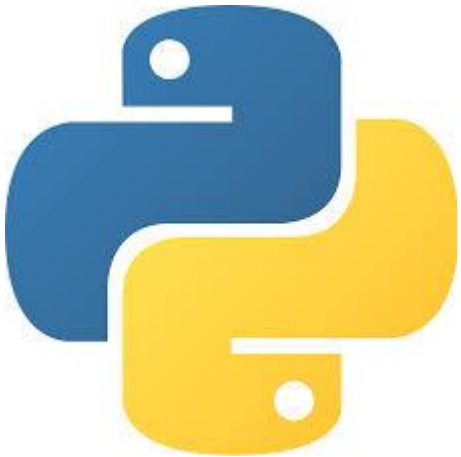


Ilustración 31. Logo de Python.

"Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de

programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma." Definición de Wikipedia: https://es.wikipedia.org/wiki/Python

En este caso, Python se usaría como lenguaje de back-end, que junto con las librerías existentes permititía realizar un análisis de los sentimientos completo.

Se trata del único lenguaje de programación válido como back-end para realizar un análisis de sentimientos, ya que a través de sus librerías permite realizarlo de una manera sencilla y muy eficaz. Además, posee distintas alternativas que lo convierten en un lenguaje válido como back-end: (Flask y Django).

- **PHP**

Ilustración 32. Logo de PHP.

Se trata del lenguaje de back-end por excelencia ("*PHP es un lenguaje de programación de uso general que se adapta especialmente al desarrollo web*" según Wikipedia: https://es.wikipedia.org/wiki/PHP), pero pese al conocimiento del lenguaje por parte del desarrollador, se descartó debido a que *"los recursos en PHP para el desarrollo de herramientas de procesamiento del lenguaje natural y de aprendizaje automático son escasos y poco conocidos"* (pág. 2 del TFG Análisis de Sentimientos aplicado a la opinión política en Twitter: un sistema de clasificación en tiempo real: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/96770/6/pllorenteayusoTFG0619memoria.pdf).

- **Framework para el Desarrollo web**

  - **Flask**

Ilustración 33. Logo de Flask.

Para poder usar Python como back-end, se ha hecho uso de un framework que lo hiciese posible. De Flask destacaa la sencillez que presenta frente al resto de frameworks, y unido a su sencillez, el gran potencial que nos dota. Cabe decir, que Flask en sí no es un framework, sino más bien un microframework al cual se le pueden ir dotando de funcionalidades según sean necesarias.

"Flask es un framework minimalista escrito en Python que permite crear aplicaciones web rápidamente y con un mínimo número de líneas de código. Está basado en la especificación WSGI de Werkzeug y el motor de templates Jinja2 y tiene una licencia BSD." Definición de Wikipedia: https://es.wikipedia.org/wiki/Flask

- **Django**

Ilustración 34. Logo Django.

"Django es un framework web Python de alto nivel que fomenta el desarrollo rápido y un diseño limpio y pragmático. Creado por desarrolladores experimentados, se encarga de gran parte de la molestia del desarrollo web, por lo que puede concentrarse en escribir su aplicación sin necesidad de reinventar la rueda. Es gratis y de código abierto." Definición de Django: https://www.djangoproject.com/

Es el principal framework web de Python, más complejo que Flask al tratarse de un framework completo, y con un gran potencial. Permite que el desarrollo web con Python sea de un gran nivel.

- **Proveedores de la nube**

De todos los proveedores de PaaS (vistos anteriormente en el capítulo 5), las posibilidades oscilaban entre los tres grandes proveedores existentes (GCP, AWS o Azure). En este caso, el elegido ha sido Azure, gracias a su sencillez de integración con aplicaciones web realizadas con Flask, y a la existencia de una interfaz gráfica muy intuitive que permite realizar todo el proceso sencillamente.

- **Bases de Datos**

En este punto, las opciones son múltiples. Pero partiendo de la base de que se va a tratar de una base de datos tipo NoSQL, y para cerrar el círculo aun más, se trata de un proyecto en la nube por lo tanto, nos centraremos en opciones de bases de datos NoSQL en la nube.

- **Firebase**

Ilustración 35. Logo Firebase.

Se trata de una plataforma de desarrollo de aplicaciones web ubicada en la nube, propiedad de Google.

Dentro de las opciones que hay en Firebase, existe la opción de base de datos en tiempo real (Firebase Realtime Database), que presenta grandes ventajas en proyectos que se realizan en tiempo real. La interfaz que nos aporta es sencilla e intuitiva.

"*Firebase Realtime Database es una base de datos NoSQL alojada en la nube que te permite almacenar y sincronizar datos entre tus usuarios en tiempo real*". Definición de Google: https://firebase.google.com/?hl=es

- **Azure Cosmos DB**

"*Azure Cosmos DB es un servicio de base de datos NoSQL totalmente administrado para el desarrollo de aplicaciones modernas, con tiempos de respuesta garantizados de menos de diez milisegundos y una disponibilidad del 99,999 % con el respaldo de acuerdos de nivel de servicio, escalabilidad*

automática instantánea y API de código abierto para MongoDB y Cassandra. Disfrute de operaciones de escritura y lectura rápidas en cualquier parte del mundo gracias a la distribución global de la arquitectura multimaestro llave en mano." Definición de Azure: https://azure.microsoft.com/es-es/services/cosmos-db/

Se trata de una opción muy válida para este proyecto, ya que como Firebase, tiene la característica de lectura y escritura de datos en tiempo real.

7.6. Implementación

Se trata de una aplicación web realizada en Python, que posee 260 líneas de código, ya que como ya hemos mencionado, esta aplicación tiene como objetivo medir la capacidad de la nube en la actualidad, frente a los servidores on-premise.

El framework usado es Flask, por lo que es necesario realizar una breve introducción. En sí, Flask es un "micro" framework escrito en Python y concebido para facilitar el desarrollo de aplicaciones web bajo el patron MVC.

El patron MVC es una manera de trabajar que permite diferenciar el modelo de datos (la base de datos), la vista (página HTML) y el controlador (donde se gestionan las peticiones de la app web).

En nuestro caso, únicamente poseemos un controlador, que será el que recolecte la información, la analice y la mande a las vistas. Este controlador, escrito en Python realiza la siguiente función:

1. Carga la página de inicio.
2. Recibe las peticiones del usuario para realizar el análisis. Se tratan de peticiones de tipo "POST", que son recogidas y alamcenadas, para que la aplicación pueda continuar su uso.
3. Recolecta uno a uno cada tweet, lo analiza y lo procesa, obteniendo resultados individuales de cada tweet. Todo esto teniendo en cuenta que la librería tweepy permite obtener un número limitado de tweets

cada cierto tiempo, es por ello, que en el caso de sobrepasar ese límite, la aplicación se pondrá en reposos hasta poder volver a recolectar más tweets.

4. Almacena los resultados individuales en la base de datos, y manda los resultados globales a la vista “hashtag”, la cual será la encargada de mostrarla.

Y como base de datos, la seleccionada ha sido Firebase, debido a que el desarrollador contaba con experiencia previa con esta tecnología.

Las librerías de Python usadas (contando que la librería de Flask es una de ellas) son las siguientes:

- **tweepy**: es una librería de Python que permite acceder a la API de Twitter de una manera fácil y sencilla. La versión usada es la 3.8.0.
- **pyrebase**: es una librería de Python que permite conectar con la API de Firebase. La versión usada es la 4.3.0
- **TextBlob**: “*TextBlob es una biblioteca de Python (2 y 3) para procesar datos textuales. Proporciona una API simple para sumergirse en tareas comunes de procesamiento de lenguaje natural (PLN), como etiquetado de parte del discurso, extracción de frases nominales, análisis de sentimientos, clasificación, traducción y más*.” Definición de la página de TextBlob: https://textblob.readthedocs.io/en/dev/ En este caso, la librería TextBlob ha sido utilizada para análisis de sentimientos.

7.7. Interfaz

En este punto, nos centramos en la interfaz una vez creada completamente, analizando en profundidad cada uno de los aspectos que la componen.

Todas estas vistas han sido realizadas con los lenguajes HTML5, CSS y JavaScript desde cero, siguiendo un modelo web responsive.

7.7.1. Vista Home

Una vez desarrollada, así quedaría de primer plano en el caso de superar los 992px de anchura, donde se sigue la idea de los storyboards, salvo que en la barra de navegación hay un elemento más que será descrito.

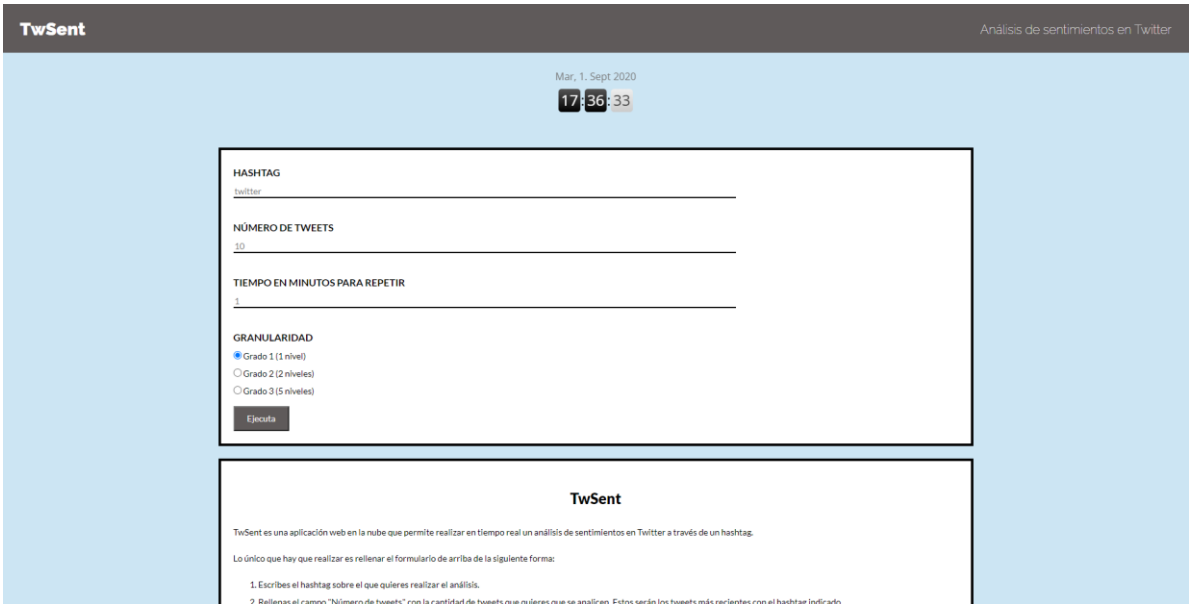


Ilustración 36. Vista Home más de 992px.

Esta vista presenta una barra de navegación superior, en la que se aprecia a la izquierda el título de la aplicación, que a su vez es un enlace a esta propia vista (es decir, a la página de inicio) y a la derecho una breve descripción de lo que esta aplicación es.

Justo debajo, nos encontramos un reloj, que nos marca la fecha y la hora, ya que para esta aplicación, que analiza y procesa datos en tiempo real, es bastante funcional.

Más abajo, nos encontramos una sección, que se trata de la sección principal de esta página, ya que contiene el formulario a rellenar, junto con el botón para ejecutar la aplicación. Posee varios campos, con unos valores sombreados, que bien podrían ser válidos para una ejecución, aunque es el usuario el que debe rellenarlos, y pulsar el botón.

Debajo de esta sección, aparece cortada la siguiente sección, la cual se trata de un resumen de lo que la aplicación realiza, junto con los pasos para rellenar el formulario de manera correcta.

Debido a que esta aplicación presenta un diseño responsive, se han diseñado nuevos modos para estos casos:

- En el caso de estar entre 641px y 992px de anchura quedaría de esta forma:

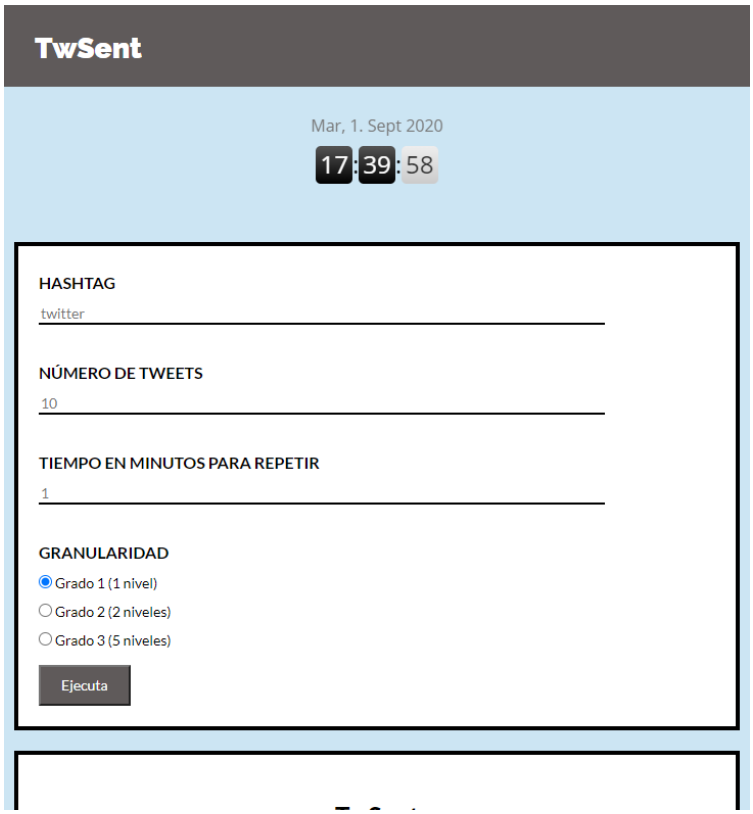


Ilustración 37. Vista Home entre 992px y 641px.

La principal diferencia que encontramos entre esta vista y la anterior es la ausencia de la descripción de la aplicación en la barra de navegación. Además, de tener una anchura menor tanto en las secciones como en la longitud de la barra de cada entrada del formulario.

- Y en el caso de ser menor de 641px de anchura:

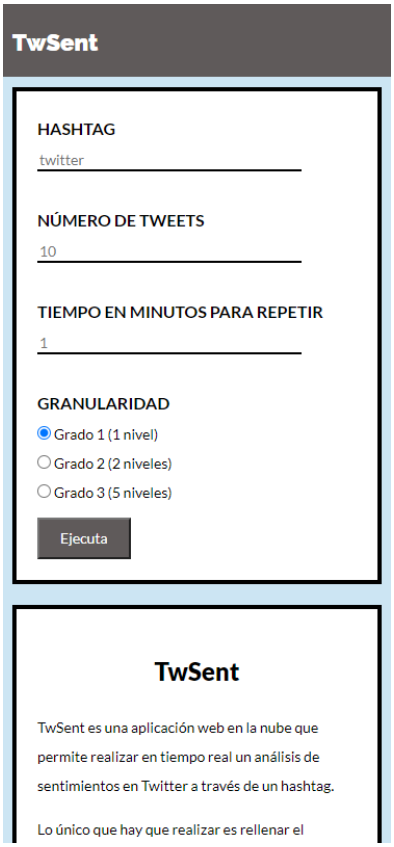


Ilustración 38. Vista Home menos de 641px.

En este caso la diferencia con la anterior vista simplemente radica en la anchura de los recuadros y la longitud de las barras, que obviamente es menor, para ajustarse de una manera estética y funcional a las anchuras principalmente de smartphones.

7.7.2. Vista Hashtag

- Una vez creada la vista hashtag, cuando la pantalla es más ancha que 992px, se ve de esta forma:

Ilustración 39. Vista Hashtag más de 992px.

Aquí se puede apreciar lo siguiente. En primer lugar, la barra de navegación es igual que la de la vista anterior con esta anchura, y con las mismas funcionalidades (es decir, al pulsar en el título “TwSent” nos lleva a la página de inicio). Pero además, aparece en el medio de ambos elementos el hashtag, el cual está siendo analizado. Este aparece con un grosor intermedio entre ambos elementos.

Justo debajo, aparece el mismo reloj, pero ya no centrado. Esto se debe a que a su misma altura aparece un recuadro pequeño que muestra los parámetros de la ejecución (excepto la granularidad, que es observada directamente con los emoticonos). Estos dos elementos aparecen centrados, pero separados, quedando dentro de la anchura del recuadro principal.

Más abajo, nos encontramos con la sección principal, un recuadro que engloba los resultados de la ejecución. Este recuadro se encuentra dividido en tres columnas, una central con una anchura superior, y dos a ambos lados. En la central se muestra un emoticono de tamaño superior, que simboliza el resultado del análisis general (como bien muestra el título que hay encima de este). A la izquierda aparece el emoticono que hace referencia al tweet con la polaridad minima, y justo debajo un recuadro en el cual se muestra el usuario que ha publicado el tweet y el texto de este. De igual forma, pero esta vez con el tweet con máxima polaridad ocurre en la columna de la derecha. Ambos recuadros al ser pinchados nos redirigen al enlace de Twitter de estos tweets, y al pasar el ratón por encima de los emoticonos se mostrará en un recuadro la polaridad.

Por último, el elemento que aparece en la parte inferior, se trata de un botón que el usuario puede pulsar en cualquier momento y que para la ejecución.

- Cuando la anchura de la pantalla se encuentra entre 641px y 992px aparece así:

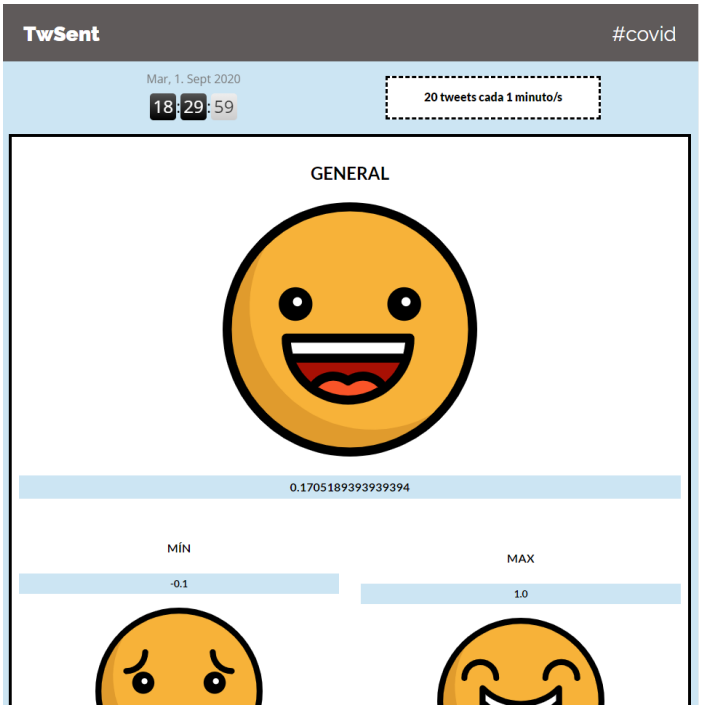


Ilustración 40. Vista Hashtag entre 992px y 641px.

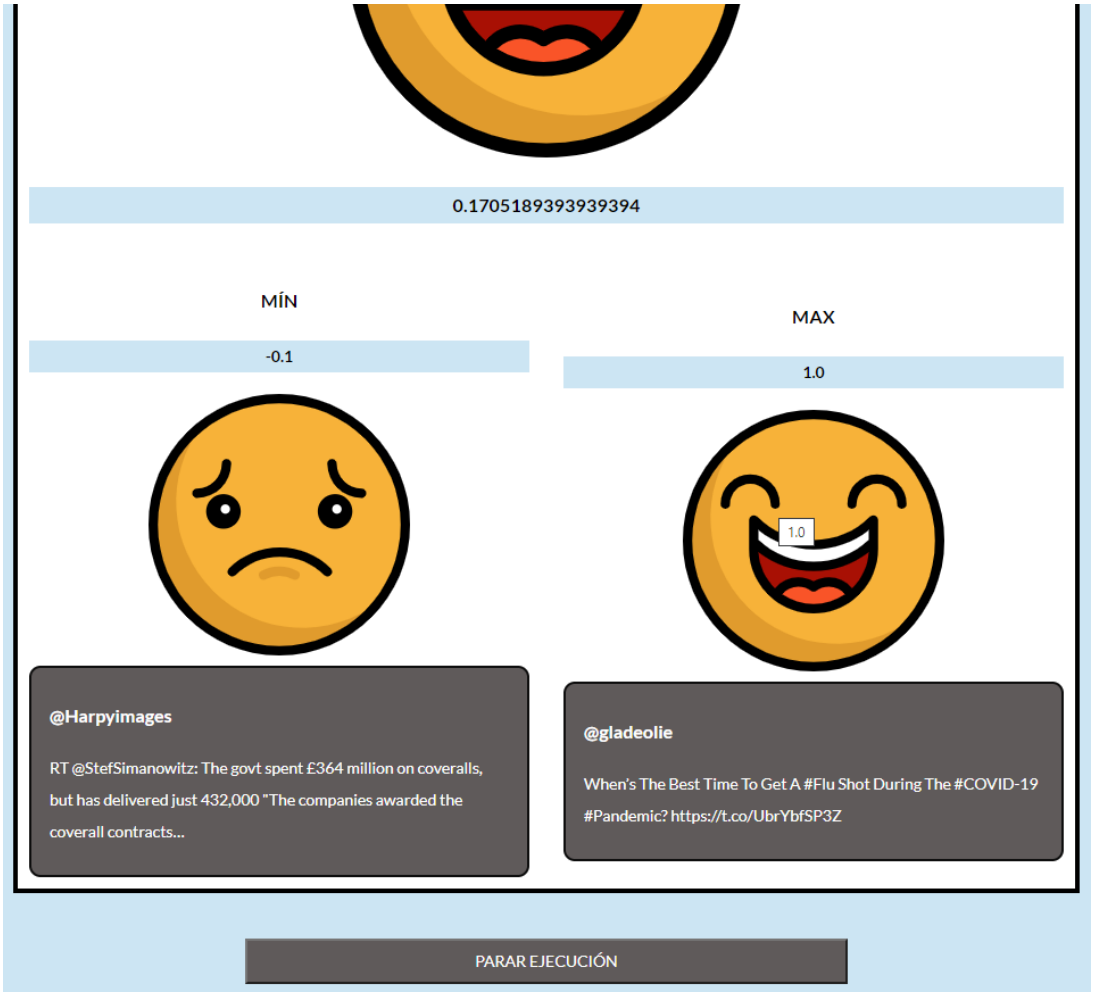


Ilustración 41. Vista Hashtag entre 992px y 641px (2)

.

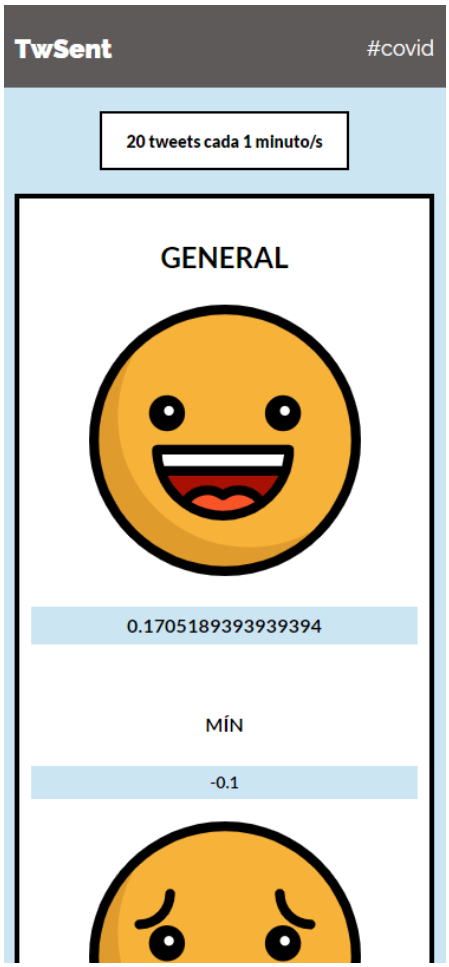
De estas imágenes a la anterior, podemos destacar las siguientes diferencias.

En la barra de navegación, la descripción desaparece y el texto con el hashtag se desplaza hacia la derecha.

En el recuadro principal, las tres columnas se convierten en dos, pero aparece otra fila inferior. La primera fila, con sus dos columnas está ocupada completamente por el emoticono con la polaridad general y un recuadro que muestra explícitamente la polaridad. La fila inferior se divide en dos columnas, la de la izquierda muestra el emoticono del tweet con la polaridad mínima, junto al recuadro con el texto del tweet y otro recuadro (por encima del emoticono) con la polaridad de manera explícita (al igual que sucede con el recuadro de la

polaridad general). La columna de la derecha es exactamente igual que la columna de la izquierda, excepto porque muestra el tweet con mayor polaridad. Ambos recuadros con el texto de los tweets siguen redirigiéndonos a Twitter, y esto será igual en el siguiente tamaño.

- Si la anchura es menor a 641px, la vista “hashtag” aparece así:



Ilustraciones 42 y 43. Vistas Hashtag entre menos de 641px.

Respecto a los dos tamaños anteriores, en este se puede observar que el reloj desaparece, la barra de navegación es igual que la barra del tamaño intermedio, el recuadro con los parámetros pasa de tener un borde con barras a tenerlo continuo, y sobre todo, que los resultados se muestran de forma distinta. Se dejan atrás las columnas, y aparecen tres filas, donde el tamaño del emoticono es exactamente igual en todas. En la primera se muestra el emoticono con la polaridad general, en el segundo con la del tweet con menor

polaridad junto al recuadro con el texto del tweet y en la tercera el emoticono asociado al tweet con mayor polaridad más el recuadro con el texto del tweet. En todas las filas, aparece un recuadro con la polaridad de manera explícita.

7.8. Despliegue de la aplicación a la nube de Azure

7.8.1. Cómo realizarlo

La nube seleccionada, como ya se ha dicho anteriormente, ha sido la de Microsoft Azure. La herramienta para el modelo PaaS que nos ofrece para aplicaciones web es App Service. App Service es el proveedor de alojamiento en la nube de Microsoft.

A continuación, se explicará paso a paso cómo ha sido el deploy de la aplicación TwSent a la nube de Azure, es decir, una aplicación Flask a Azure.

En primer lugar, es necesario preparar la aplicación. Como mínimo el proyecto necesita tener:

- Un archivo de Python.
- Un archivo *requirements.txt* que contenga la lista de librerías de las que el proyecto depende.

En este caso, nuestro proyecto presenta un archivo de Python, junto con otra serie de documentos HTML, CSS, imágenes, etcéteta. Además, fue necesario crear el archive requirements.txt con las librerías mencionadas en el punto 8.3.

Una vez hecho esto, es necesario crear un repositorio en GitHub y subir todo el código ahí, junto con todos los archivos, ya que usaremos GitHub para desplegar el proyecto en Azure.

Obviamente, es necesario crear una cuenta en Azure. Este punto es importante, ya que nos ofrecen 170€ gratis durante 12 meses. Es con ese crédito, con el que hemos usado distintos planes para la aplicación.

Una vez temenos la cuenta de Azure, navegamos hasta App Service, como en la imagen anterior. Una vez ahí, pinchamos en “Agregar” para crear una nueva. Nos aparecerá la siguiente pantalla.

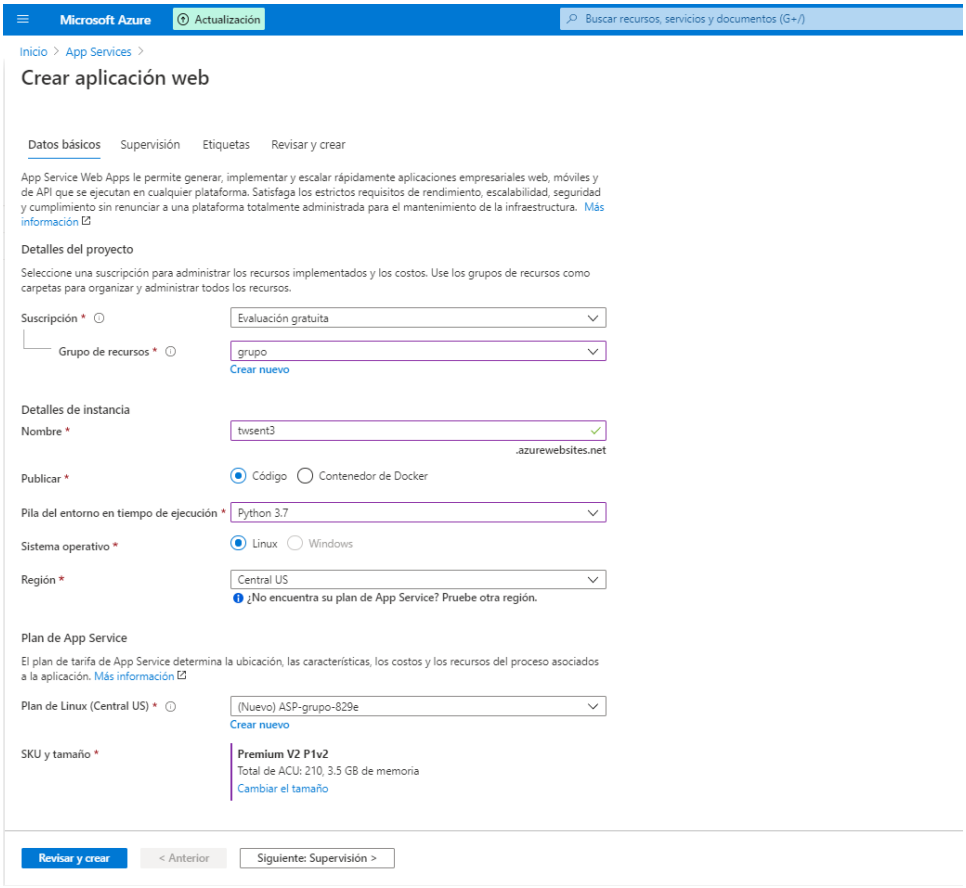


Ilustración 44. Creación de aplicación web en Azure.

En esta pantalla, seleccionamos lo que aparece en la ilustración 41, como se explica a continuación:

- En suscripción elegimos “Evaluación gratuita”, que hace referencia al crédito de 170€ gratuito que se nos ofrece.
- Una vez esto, creamos un grupo de recursos con el nombre que nos parezca oportuno, y le agregamos un nombre a la aplicación, en nuestro caso twsent.

- Elegimos la pila de entorno, en este caso será “Python 3.7” y la región, en nuestro caso “West Europe”.
- Seleccionamos un plan, que serán definidos en profundidad en el siguiente punto.
- Pulsamos “Revisar y crear” para crear la aplicación, y habrá que esperar unos segundos y estará lista.

Cuando ya está creada, iremos al menú de App Service, y pincharemos en nuestra aplicación. Se nos mostrará la siguiente página con la información general.

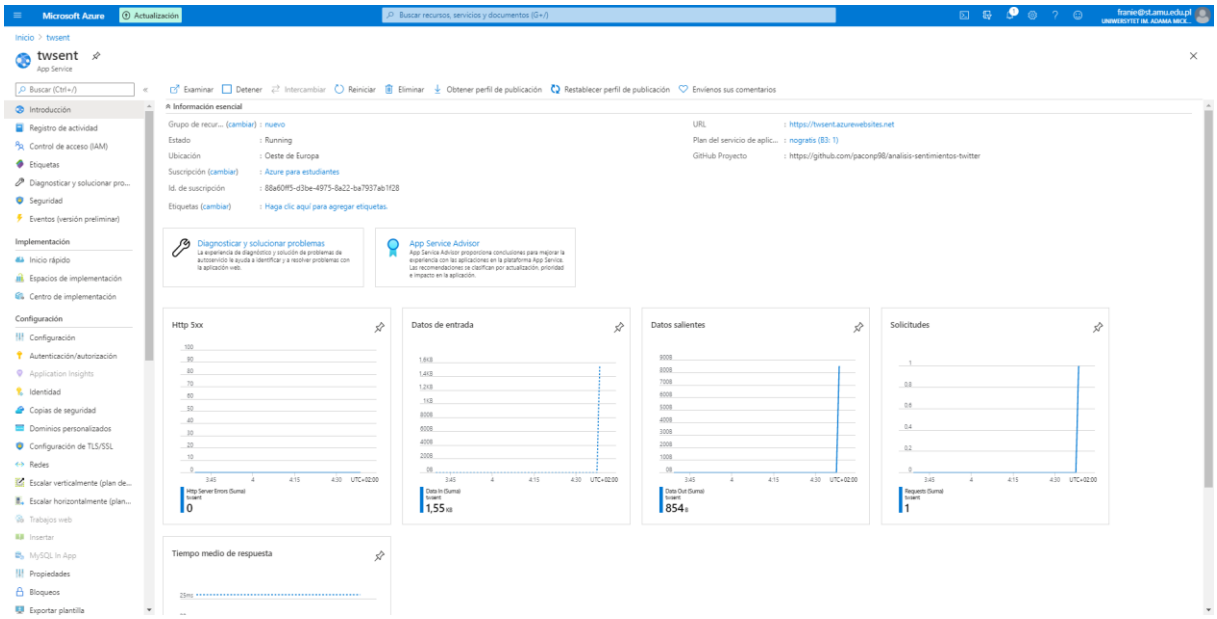


Ilustración 45. Página principal de TwSent en Azure App Service.

A la derecha de la ilustración 42 aparece un menú, con una gran variedad de entradas, que permiten realizar configuraciones en la aplicación y obtener información acerca de esta.

Para conectar nuestra aplicación con el repositorio de GitHub, pincharemos en la entrada del menú con el nombre “Centro de implementación”. Aquí nos dará algunas opciones de implementación, y elegiremos GitHub.

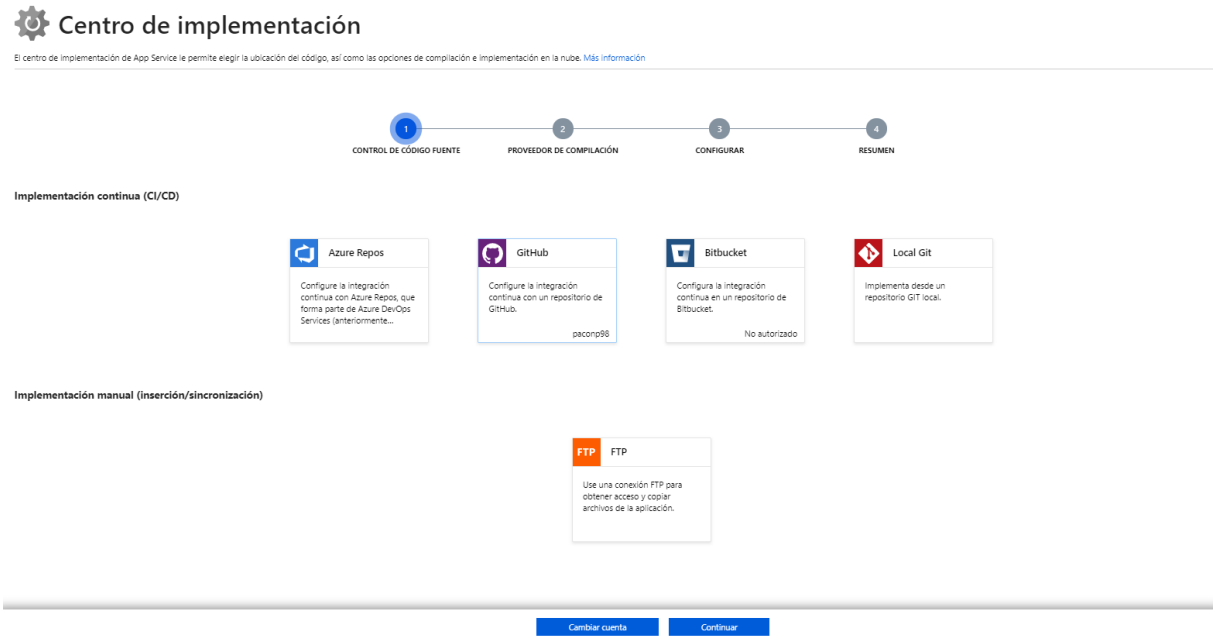


Ilustración 46. Centro de implementación en App Service (Paso 1).

Una vez seleccionado, iniciaremos sesión en nuestra cuenta de GitHub (en la imagen anterior aparece ya la sesión iniciada), y pulsaremos continuar. La siguiente pantalla nos muestra tres proveedores de compilación, y seleccionaremos “Servicio de compilación de App Service” (es decir, el motor de Kudu”), ya que funciona correctamente.

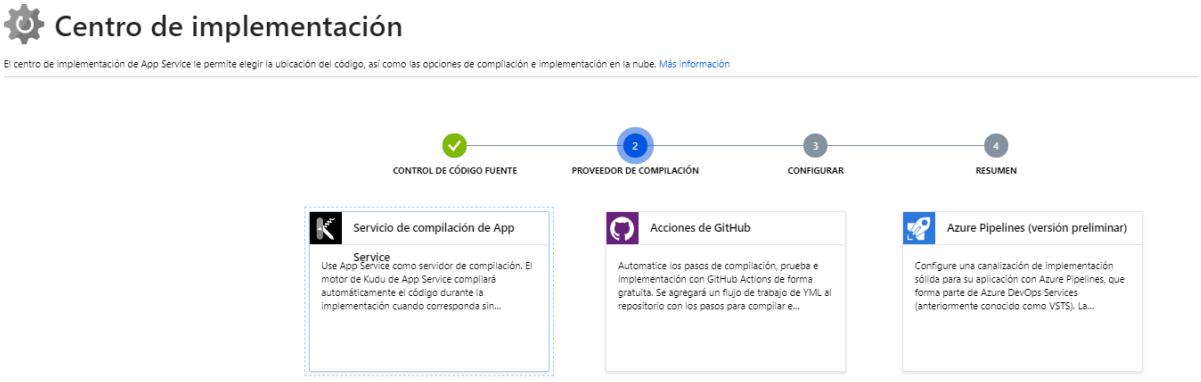


Ilustración 47. Centro de implementación en App Service (Paso 2).

Una vez pulsemos continuar, iremos al paso 3. Aquí ya hemos conectado con GitHub y tendremos que elegir tanto el repositorio en el que tenemos todo el código, como la rama de git en la que se encuentre el código correcto.

Ilustración 48. Centro de implementación en App Service (Paso 3).

Una vez seleccionamos esto, pasamos a la pantalla de resumen, que muestra lo seleccionado en los pasos anteriores. Una vez ahí, se pulsa “Finalizar” y ya estaremos conectados.

Para comprobar que estamos conectados, pulsamos de nuevo en “Centro de implementación” y nos tendrá que aparecer el último *commit* que aparezca en GitHub. Y cada vez que realicemos uno nuevo y lo subamos a GitHub aparecerá en este punto. Habrá que esperar a que compile, para que aparezca el estado “Correcto (Active)”, que indica que la nueva versión de la aplicación ha sido compilada con éxito y que funciona correctamente.

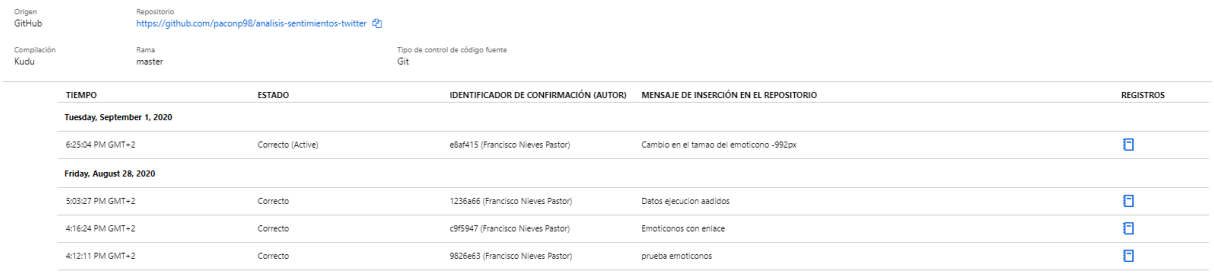


Ilustración 49. Centro de implementación en App Service (Finalizado).

Puede que suceda que la compilación produzca un error, y en vez de aparecer el estado “Correcto”, aparecerá el estado “Error”. En este caso, quedará activa la última versión que ha estado funcionando. Si esto ocurre, quiere decir que la última versión tiene un error en el código y hay que solucionarlo.

Solo nos queda un último paso para que la aplicación funcione. Para ello debemos ir a la entrada del menú lateral “Configuración”. Ahí, pinchamos en el menú de arriba en “Configuración general”.

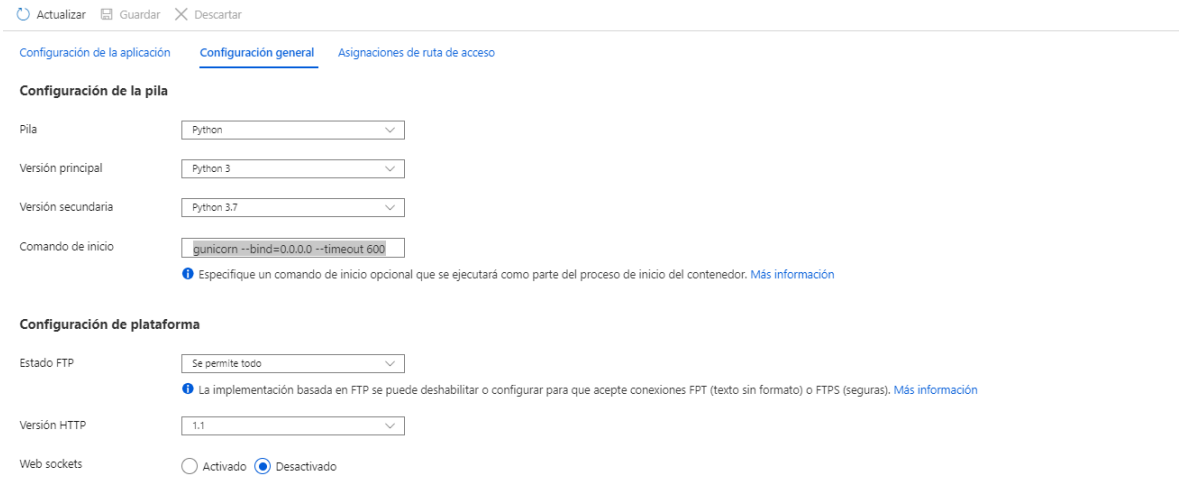


Ilustración 50. Configuración en App Service.

Ahí rellenamos el campo “Comando de inicio”, que aparece en blanco, con el siguiente texto:

gunicorn --bind=0.0.0.0 --timeout 600 main:app

Esto es necesario, ya que el archivo principal Python de nuestro proyecto se llama *main.py*, y es necesario indicárselo a Azure, ya que el que este detecta por defecto es aquel con el nombre *application.py*. Por lo tanto, este comando enlaza el archivo principal con Azure.

7.8.2. Planes de Azure y costes

Existen varios planes, cada cual con un precio asociado, y que se dividen en tres grupos (“Desarrollo y prueba”, “Producción” y “Aislado”). Cada uno de estos planes presenta unas ventajas y unas desventajas.

- **Desarrollo y prueba**

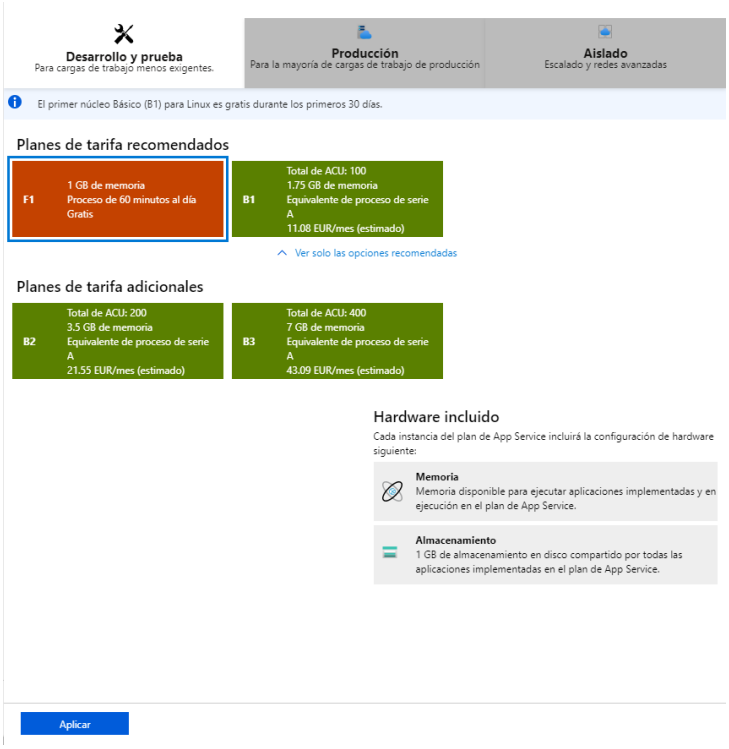


Ilustración 51. Plan F1.

Se aprecian 4 planes distintos, el primero (F1) gratis, es decir, aplicarlo no descontaría nada del crédito que poseemos. Incluye lo que aparece en la imagen.

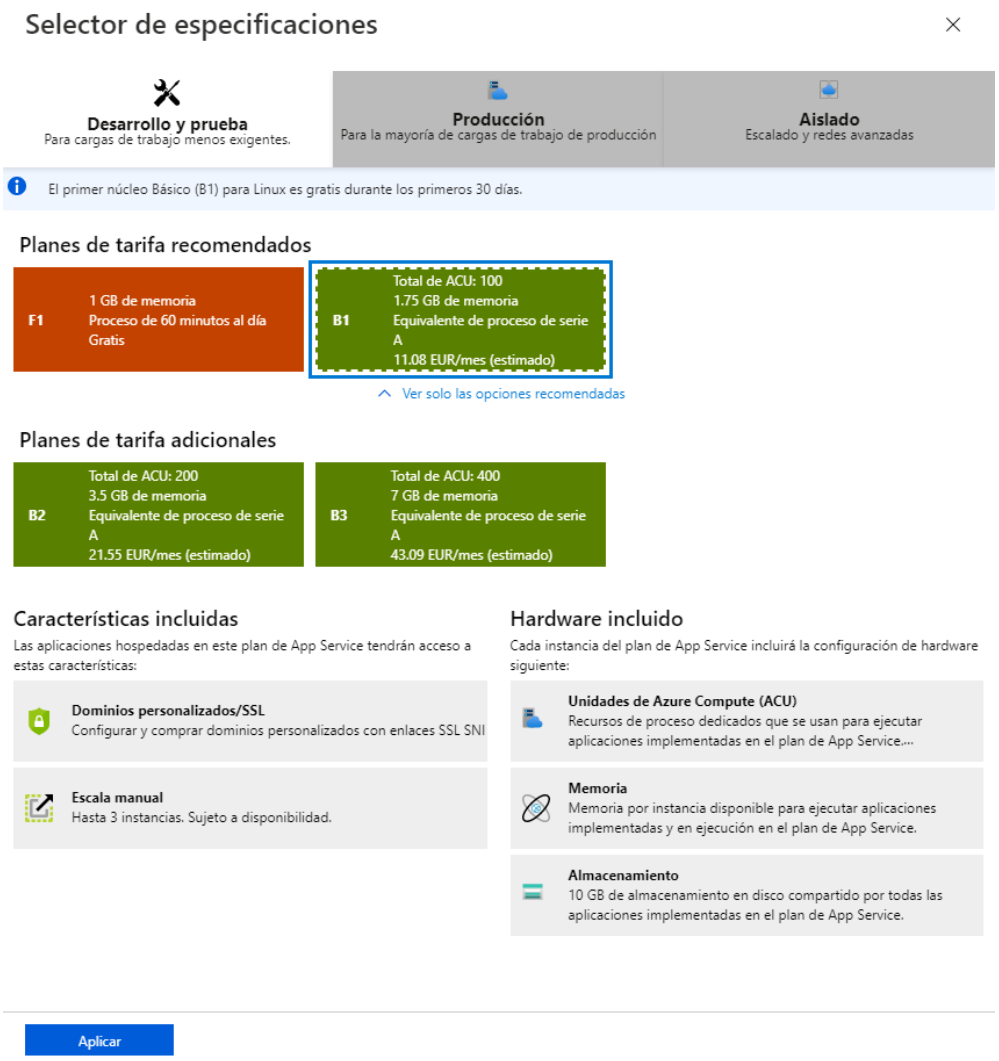


Ilustración 52. Planes B1, B2, B3.

Los demás planes, tanto el B1, B2 y B3 (de menos a más prestaciones), incluyen lo que aparece en la imagen, y tienen un costo mensual estimado, dependiendo de la capacidad que nos ofrecen. Frente al anterior plan gratis, estos nos ofrecen ACUs, más memoria, dominios personalizados/SSL y escala manual de hasta 3 instancias.

- **Producción**

Dentro de producción existen 6 planes, divididos en dos tipos. En primer lugar se encuentran los planes P1V2, P2V2 y P3V2, que poseen las siguientes características, mucho mejores que las de Desarrollo y prueba:

Ilustración 53. Planes P1V2, P2V2, P3V2.

A diferencia de los planes de Desarrollo y prueba, estos planes (cada uno con sus ACUs, su memoria y su precio asociado) tienen mejores características, que van a hacer que tu aplicación web funcione más eficientemente.

Poseen escalado automático de hasta 20 instancias, espacios de ensayo, 250gb de almacenamiento, copias de seguridad diarias y más características que aparecen en la imagen.

Pero también existen otro tipo de planes dentro de la categoría de Producción, los planes S1, S2, y S3. Estos planes están un paso por detrás de los mencionados anteriormente, ya que pasan de 250gb de almacenamiento a 50gb, de un escalado automático de hasta 20 instancias a 10 instancias, de 20 espacios de ensayo a 5 espacios y otras diferencias que se pueden aprecias entre esta imagen y la anterior.

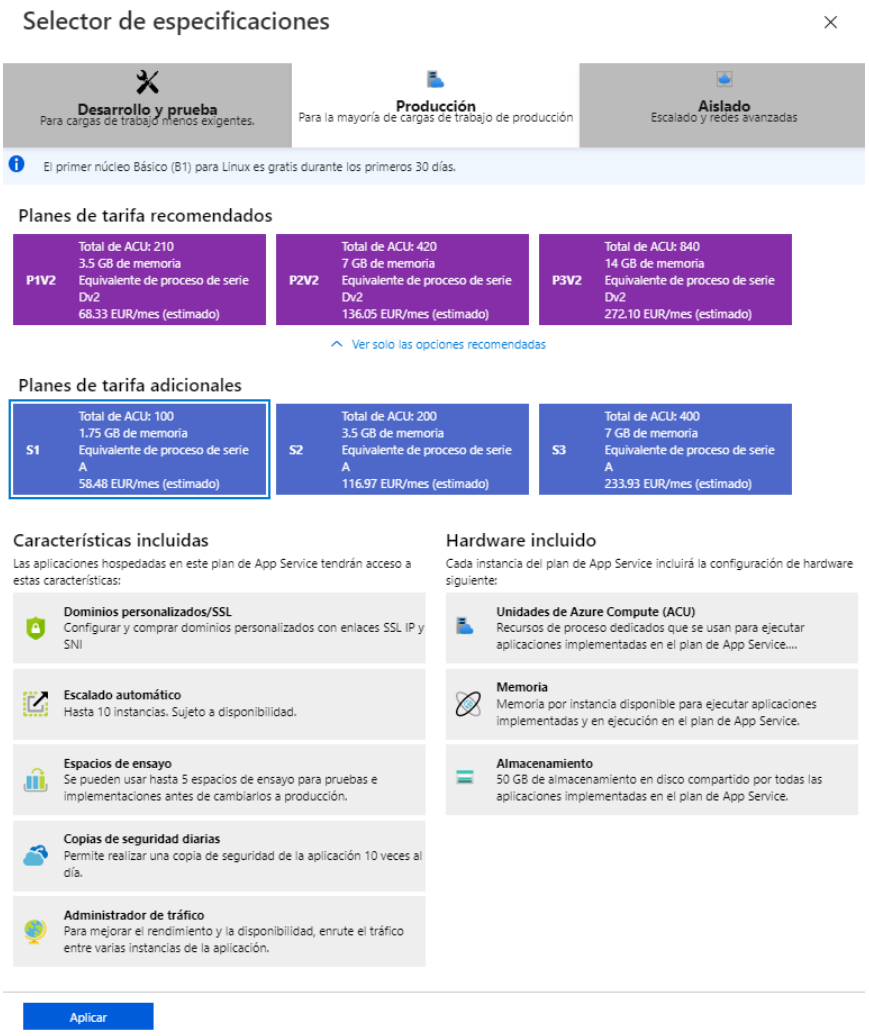


Ilustración 54. Planes S1, S2, S3.

- **Aislado**

Esta última categoría posee tres planes distintos, del mismo tipo, y está enfocada para Escalado y redes avanzadas. Estos planes son I1, I2, y I3. Se trata de los planes más fuertes de App Service, ya que supera las características de los planes de Producción. Posee una red aislada, pasa de 250gb a 1Tb… Una gran cantidad de mejoras que se aprecian en la siguiente imagen.

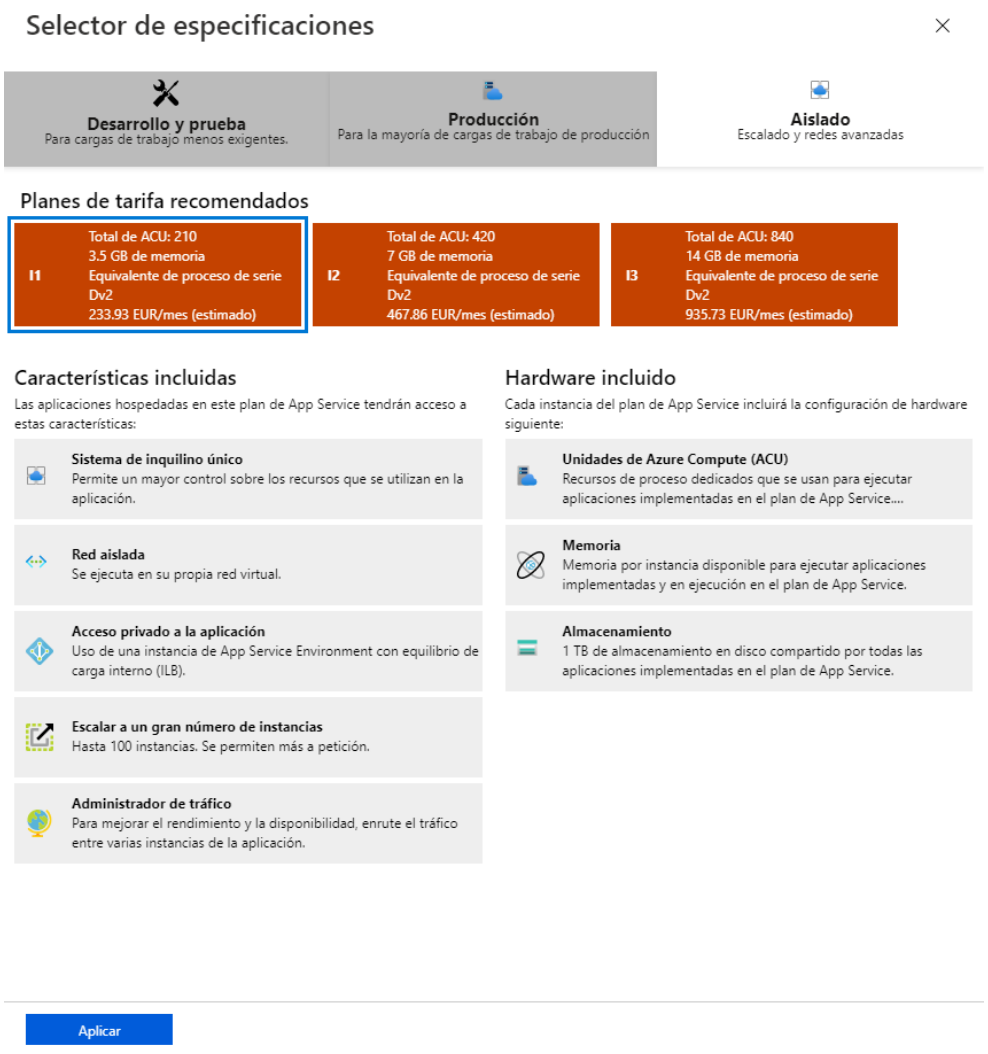


Ilustración 55. Planes I1, I2, I3.

Está claro que estos planes son muchísimos más caros, llegando a costar 935.73 €/mes el plan I3.

7.9. Pruebas en local

Una vez realizado el análisis, diseño e implementación, es necesario realizar una serie de pruebas para comprobar que la aplicación funciona correctamente.

Para ello, en primer lugar hemos realizado pruebas en local, antes de subirlo a la nube, para comprobar el funcionamiento de la aplicación. A lo largo de las ejecuciones, sobre todo en las primeras, aparecían problemas, que se han ido solucionando poco a poco, hasta llegar a un punto, en el que los errores han desaparecido casi al 100%. Al tratarse de Flask, los errores que ocurren aparecen en la web cuando se ejecuta, y los muestra en el lenguaje de jinja2, que al usar el método render_template() de Flask se activa.

Un defecto que aparece en la aplicación, tanto en local como en la nube, ya ha sido mencionado, y se trata de que al recibir muchas peticiones para extraer tweets de la API de Twitter, puede colapsarse y que la aplicación se quede en reposo hasta que estén disponibles los tweets. La solución a este problema, será propuesta como trabajos futuros para la mejora del proyecto.

Otro de los trabajos futuros para esta aplicación, se centra en el hecho de volver a la página de inicio. Cuando se solicita este recurso, no es posible realizarlo hasta que no se han procesado todos los tweets solicitados por el usuario. Es por ello, que en alguna ocasión, esta solicitud haga esperar al usuario un pequeño tiempo hasta que es completada.

7.10. Pruebas en la nube

Una vez testeado el funcionamiento de la aplicación en local, probando distintas cargas de trabajo, se ha pasado a testear la aplicación en la nube.

Aquí la interfaz de Azure nos provee de gráficas que nos dan una idea de la carga de trabajo y de lo que la aplicación está realizando en cada momento y cómo está gestionando estos recursos.

Las pruebas realizadas han sido con distintos planes que en Azure App Service existen. Las principales han sido con su plan gratuito, que permite una carga de trabajo minima durante 60 minutos al día, para así encontrar el límite que la nube de Azure nos permite. Este plan se denomina F1 y posee 1GB de memoria.

Los resultados obtenidos con este plan han sido desfavorables, ya que la página principal tardaba mucho en cargar, y una vez se mandaba a ejecución, las imágenes de los emoticonos en algunos casos no cargaban correctamente, y en muchas ocasiones ocurría un error 504, es decir, por parte del servidor, y este se debía a la carga excesiva de trabajo que se le pedía (cabe destacar que únicamente se ejecutaba por un usuario a la vez).

Ilustración 56. Error 504 en TwSent.

Una vez se pasó a un plan con mayor rendimiento (B3), con 7GB de memoria y con un total de ACU de 400, mejoró el rendimiento. Los problemas de carga de emoticonos se redujeron considerablemente, aunque la carga de la página principal, en algunas ejecuciones generaba problemas. Pero en general, para un solo usuario, funcionaba correctamente. El problema de este plan, era cuando varios usuarios lanzaban la aplicación a la vez. Aquí el servidor, al principio funciona bien, pero cuando pasa un tiempo, genera de nuevo el error 504.

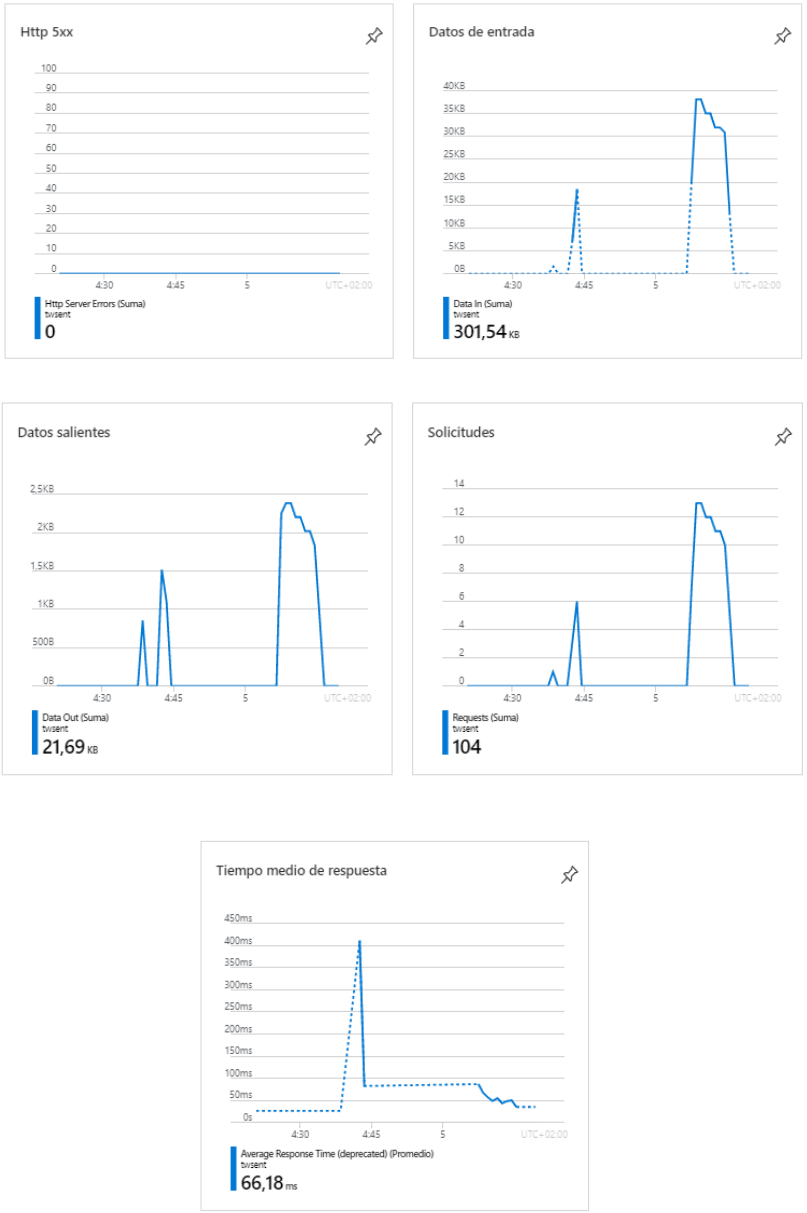


Ilustración 57. Gráficos de Azure App Service (plan B3).

Es por ello, que se ha aplicado una tarifa premium (P2V2), dentro de la categoría de Producción, que posee también 7GB de memoria y un total de ACU de 420, pero destinada no al desarrollo y pruebas, sino a la producción, siendo más eficiente que la anterior, y con la que se han obtenido mejores resultados, no solo para un usuario, sino para varios.

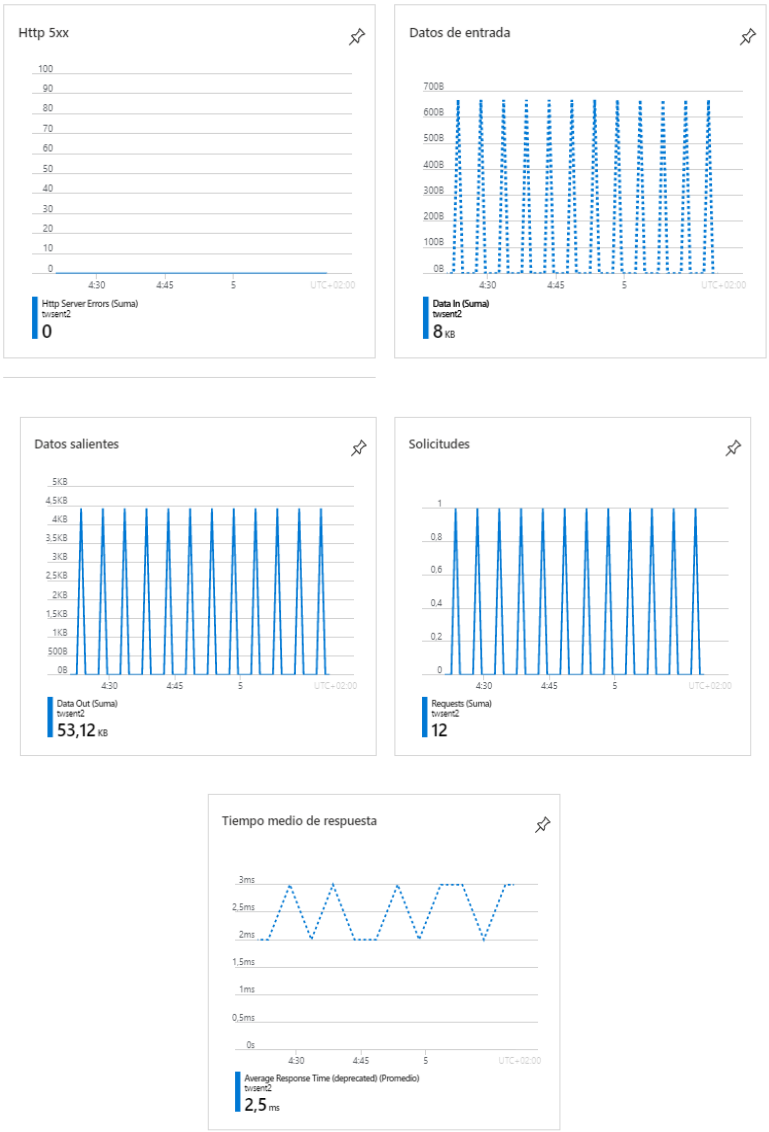


Ilustración 58. Gráficos de Azure App Service (plan P2V2).

En estas dos últimas imágenes, con una misma carga de trabajo y con dos planes distintos, se observan diferencias notables, especialmente en el tiempo de respuesta, que varía de 66,18 ms con el plan B3 a 2,5 ms con el plan P2V2.

8. CONCLUSIONES

Como se ha podido observar a lo largo de todo el proyecto, se han abordado todos los objetivos marcados:

1. En primer lugar, se ha realizado un análisis profundo de la tecnología Cloud Computing, abordando los puntos más importantes que esta tiene, creando una definición completa y concreta. Es por ello, que el objetivo de definir esta tecnología ha sido completado de manera satisfactoria en este proyecto.
2. Por último, el segundo objetivo de este proyecto era crear una aplicación y desplegarla en la nube bajo el paradigma Platform as a Service (PaaS). Este objetivo ha sido completado creando una aplicación web denominada TwSent, desplegada en la nube de Azure, y a través de esta, hemos podido medir la capacidad actual de la tecnología del Cloud Computing.

Una vez completados estos objetivos, cabe destacar que el Cloud Computing se trata de una tecnología relativamente nueva y emergente, aun así, ya está implantada en muchas empresas y en muchos ámbitos de la tecnología que cotidianamente es usada.

Los objetivos del Cloud Computing son muchos más grandes, y pretende ser la base de la tecnología que hoy en día tenemos. Es por ello, que aun le queda mucho camino por recorrer, no obstante, esto no quiere decir que sea una tecnología defectuosa. Depende más bien del ámbito en el que se emplee, ya que en ciertos ámbitos es una tecnología madura, y en otros está comenzando.

Y como ya hemos visto en este documento, esta tecnología genera una infinitud de ventajas, por lo cual, personalmente, recomiendo el uso de la tecnología del Cloud Computing en la mayoría de las empresas, porque a medio y largo plazo generará una gran cantidad de beneficios. Por lo tanto, creo que es la base de las tecnologías futuras, pero pienso que hay que ser paciente y confiar en el Cloud Computing.

Bibliografía

Libros

Rountree, D.; Castrillo, I.; Jiang, H. (2014). The basics of cloud computing: understanding the fundamental of cloud computing in theory and practice.

Jamsa, K.A. (2012): Cloud Computing.

J.K. Verma (2016): Green Cloud Computing: An Energy Efficient Methodology for Cloud Computing Enviroment.

Borko Furht; Armando Escalante (2015): Handbook of Cloud Computing.

Dan Sullivan (2009): The Definitive Guide to Cloud Computing.

Páginas Web:

Microsoft Azure, ¿qué es la nube? (10 octubre, 2019) https://azure.microsoft.com/es-es/overview/what-is-the-cloud/

Microsoft Azure, ¿Qué es la nube pública, privada e híbrida? (06 abril, 2020) https://azure.microsoft.com/es-es/overview/what-are-private-public-hybrid-clouds/

Tecnoinver, Características de los servicios en la nube (12 febrero, 2020) https://www.tecnoinver.cl/caracteristicas-de-los-servicios-en-la-nube/

Paradigma Digital, Cloud Pública vs Cloud Privada (01 enero, 2020) https://www.paradigmadigital.com/dev/cloud-publica-vs-cloud-privada/

Ionos, Cloud Computing: ¿qué se esconde detrás? (23 noviembre, 2020) https://www.ionos.es/digitalguide/servidores/know-how/cloud-computing/

Ionos, ¿Qué es IaaS? (11 abril, 2020) https://www.ionos.es/digitalguide/servidores/know-how/que-es-iaas/

Ionos, PaaS (11 abril, 2020) https://www.ionos.es/digitalguide/servidores/know-how/paas/

Ionos, ¿Qué es SaaS? (11 abril, 2020) https://www.ionos.es/digitalguide/servidores/know-how/que-es-saas/

Factum, Modelos “As a Service”, un salto decisivo a la transformación digital (17 abril, 2020) http://factum-it.es/blog/modelos-as-a-service-un-salto-decisivo-a-la-transformacion-digital

Grupo Garatu, DBaaS: El futuro en el éxito de nuestras empresas está en las bases de datos en la nube (17 abril, 2020) https://grupogaratu.com/dbaas-bases-datos-la-nube/

Red Hat, ¿Qué es FaaS? (17 abril, 2020) https://www.redhat.com/es/topics/cloud-native-apps/what-is-faas

Ionos, CaaS, virtualización a demanda (17 abril, 2020) https://www.ionos.es/digitalguide/servidores/know-how/caas-virtualizacion-a-demanda/

SearchDataCenter, CaaS (17 abril, 2020) https://searchdatacenter.techtarget.com/es/foto-articulo/2240224756/10-definiciones-de-modelos-de-servicios-en-la-nube-que-debe-conocer/6/Comunicaciones-como-servicio-CaaS

Channel Partner, NaaS o servicios de red bajo demanda (18 abril, 2020) https://www.channelpartner.es/negocios/especiales/1065846002202/naas-servicios-red-demanda.1.html

Einatec, Historia del Cloud Computing, ¿quién lo hizo posible? (20 abril, 2020) https://einatec.com/historia-cloud-computing/

Be Services, Historia del Cloud Computing (20 abril, 2020) https://www.beservices.es/cloud-computing-historia-n-5319-es

Be Services, La virtualización y el Cloud Computing: todo lo que debes saber (25 abril, 2020) https://www.beservices.es/cloud-computing-virtualizacion-n-5345-es

Hewlett Packard Enterprise, ¿Qué son los contenedores? (29 abril, 2020) https://www.hpe.com/es/es/what-is/containers.html

WikiDot, La nube, Ventajas y Desventajas de Cloud Computing (01 mayo, 2020) http://lanube.wikidot.com/ventajas-y-desventajas-de-cloud-computing

Zierzo, Cloud Computing. Ventajas y desventajas (01 mayo, 2020) https://zierzo.es/cloud-computing-ventajas-desventajas/

Red Hat, ¿Qué son los proveedores de nube? (01 mayo, 2020) https://www.redhat.com/es/topics/cloud-computing/what-are-cloud-providers

Google, App Engine (02 mayo, 2020) https://cloud.google.com/appengine?hl=es

Google, Compute Engine (02 mayo, 2020) https://cloud.google.com/compute

AWS, ¿Qué es AWS? (02 mayo, 2020) https://aws.amazon.com/es/what-is-aws/ 02/05/2020

AWS, ¿Qué es AWS Elastic Beanstalk? (03 mayo, 2020) https://docs.aws.amazon.com/es_es/elasticbeanstalk/latest/dg/Welcome.html

AWS, ¿Qué es AWS EC2? (03 mayo, 2020) https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/concepts.html

Microsoft Azure, ¿Qué es Azure? (03 mayo, 2020) https://azure.microsoft.com/es-es/overview/what-is-azure/

Microsoft Azure, App Service (03 mayo, 2020) https://azure.microsoft.com/es-es/services/app-service/

Microsoft Azure, Cloud Services (03 mayo, 2020) https://azure.microsoft.com/es-es/services/cloud-services/

Microsoft Azure, Virtual Machines (20 mayo, 2020) https://azure.microsoft.com/es-es/services/virtual-machines/

Paradigma Digital, AWS vs Azure vs GCP: todos los servicios cloud frente a frente (21 enero, 2020) https://www.paradigmadigital.com/dev/comparativa-servicios-cloud-aws-azure-gcp/

Altim, ¿Qué es SAP Cloud Platform? (04 mayo, 2020) https://www.altim.es/blog-noticias-tic/que-es-sap-cloud-platform/

Heroku, Heroku (04 mayo, 2020) https://www.heroku.com/

RedHat, OpenShift (04 mayo, 2020) https://www.redhat.com/es/technologies/cloud-computing/openshift

IBM, ¿Qué es la plataforma IBM Cloud? (05 mayo, 2020) https://cloud.ibm.com/docs/overview?topic=overview-whatis-platform&locale=es

Einatec, Computación en la nube vs Computación tradicional (07 mayo, 2020) https://einatec.com/computacion-en-la-nube-vs-computacion-tradicional/

Xataka, Edge Computing: qué es y por qué hay gente que piensa que es el futuro (07 mayo, 2020) https://www.xataka.com/internet-of-things/edge-computing-que-es-y-por-que-hay-gente-que-piensa-que-es-el-futuro

Tic Portal, Grid Computing (08 mayo, 2020) https://www.ticportal.es/glosario-tic/grid-computing

Open Webinars, ¿Qué es Flask? (03 julio, 2020) https://openwebinars.net/blog/que-es-flask/

GitHub, How to deploy your Flask app to Azure (15 julio, 2020) https://github.com/smartninja/example-azure-flask

Librería Tweepy https://www.tweepy.org/

Librería Pyrebase https://github.com/thisbejim/Pyrebase

Librería TextBlob https://textblob.readthedocs.io/en/dev/

Apuntes:

Apuntes Fundamentos de Ingeniería del Software 2017/2018

Apuntes Procesamiento del Lenguaje Natural 2019/2020