Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

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

Alumno: Jesús Pereira Ibáñez

Tutor: Prof. D. Víctor Manuel Rivas Santos
Dpto: Departamento de Informática

Junio, 2023

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Víctor Manuel Rivas Santos , tutor del Trabajo Fin de Grado titulado: Cloud Computing: Iniciación a la creación de aplicaciones en la nube mediante el paradigma Platform as a Service, que presenta Jesús Pereira Ibáñez, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Junio de 2023

El alumno:

JESÚS PEREIRA IBÁÑEZ

El tutor:

VÍCTOR MANUEL RIVAS SANTOS

Índice de contenido

Índice de ilustraciones

Índice de tablas

1. INTRODUCCIÓN

En este primer capítulo desarrollaremos una introducción al trabajo de Fin de Grado "Cloud Computing: iniciación a la creación de aplicaciones en la nube mediante el paradigma Platform as a Service". Durante el desarrollo de este trabajo describiremos en detalle el problema que Cloud Computing intenta resolver, así como una primera aproximación al concepto, abordándolo desde una perspectiva temporal e histórica. Del mismo modo se introducirá la motivación que nos ha llevado a elegir este proyecto, y los objetivos que se plantean y se intentan conseguir con la elaboración del mismo. Finalmente desarrollaremos una aplicación de prueba con la que pretendemos demostrar algunas de las virtudes y problemáticas que presenta crear aplicaciones en la nube mediante el paradigma Platform as a Service.

1.1. Motivación

El Cloud Computing, o Computación en la Nube, ha revolucionado la forma en que las organizaciones y grandes empresas gestionan sus recursos. La nueva realidad del mundo empresarial, así como las cada vez más demandantes necesidades que se plantean, han provocado un tsunami en forma de transformación digital a gran escala, proporcionando al tejido empresarial el poder de adaptación que demandaba.

El propósito de este trabajo de fin de grado será explorar el Cloud Computing en todas sus vertientes, centrándonos llegado el punto en el paradigma Platform as a Service. Más adelante y conforme vayamos estudiando esta fascinante tecnología, comprenderemos mejor estas palabras y podremos poner en perspectiva el enorme cambio que supone.

Llegados a este punto consideramos que comprender las profundas implicaciones que se derivan de la computación en la nube, y cómo transforma todo aquello que toca, ha adquirido un nivel de importancia inusitado hasta no hace tanto tiempo. A

través de este escrito intentaremos analizar las ventajas y, porqué no decirlo, algunos problemas que plantea su uso, así como los desafíos a los que se enfrenta y enfrentará en un futuro cercano.

1.2. Objetivo del proyecto

Pasamos ahora a presentar aquellos objetivos concretos perseguimos con la realización de este proyecto de fin de grado.

El primer objetivo consistirá en realizar un profundo estudio de la tecnología, de las principales características que aporta, de la vertiginosa evolución que ha sufrido en pocos años, de los diferentes paradigmas que se ofertan (incluido Platform as a Service, objeto principal de estudio que nos atañe), y de los principales proveedores de servicios surgidos desde los inicios hasta la candente actualidad.

El segundo objetivo estará enfocado a diseñar e implementar una pequeña aplicación que muestre las posibilidades que brinda el paradigma Platform as a Service. Las claves serán:

- Diseñar una arquitectura escalable usando la nube y aprovechando todos los servicios y herramientas que provee la nube.
- Implementar esta aplicación diseñada en la nube, intentando utilizar todo el poder de la tecnología.
- A continuación realizaremos un estudio comparativo entre esta aplicación lanzada en la nube, y su contrapartida en local, analizando en profundidad las ventajas que aporta, y si fuera necesario, los problemas que se desprenden de su uso.
- Por último realizaremos pruebas de validación y rendimiento, con el objetivo de comprobar que las virtudes descritas a lo largo del presente proyecto pueden verse nítidamente en nuestra aplicación.
- Realizar pruebas y análisis de rendimiento para validar las virtudes de la aplicación en la nube, como la escalabilidad y la eficiencia de los recursos.

Al finalizar este proyecto, esperamos haber conseguido una profunda comprensión de aquellas características y fundamentos más importantes de la tecnología, y más concretamente, del paradigma que nos ocupa, Platform as a Service.

2. CLOUD COMPUTING

En este segundo capítulo nos centraremos en estudiar en profundidad el término Cloud Computing, haciendo especial hincapié en la modalidad de servicio que en este trabajo nos atañe, Plataforma como Servicio. Se describirán tanto el concepto como todos aquellos aspectos relacionados con el mismo imprescindibles para conseguir que la imagen final sobre el tema del lector sea lo más nítida posible.

2.1. ¿Qué es Cloud Computing?

El laboratorio nacional del Departamento de Energía de los Estados Unidos, llamado Laboratorio Nacional Lawrence Berkeley, (Fox, A., 2009), dice lo siguiente:

> *"Cloud Computing se refiere tanto a las aplicaciones entregadas como a los servicios a través de Internet y el hardware y software de sistemas en los centros de datos que brindan esos servicios. Los servicios en sí se conocen desde hace mucho tiempo como software como servicio (SaaS). El hardware y el software del centro de datos es lo que llamaremos una nube. Cuando una nube se pone a disposición del público en general mediante un sistema de pago por uso, la llamamos nube pública; el servicio que se vende es Utility Computing. Utilizamos el término nube privada para referirnos a los centros de datos internos de una empresa u otra organización, que no están disponibles para el público en general. Por lo tanto, Cloud Computing es la suma de SaaS y Utility Computing, pero no incluye las nubes privadas."*

La definición que aporta el National Institute of Standards and Technology (NIST) (Mell, P., 2011) es una de las más completas y ampliamente aceptadas:

> *"Cloud Computing es un modelo para permitir el acceso adecuado y bajo demanda a un conjunto de recursos de cómputo configurables*

> *(p.e. redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser rápidamente provistos y puestos a disposición del cliente con un mínimo esfuerzo de gestión y de interacción con el proveedor del servicio."*

Estas definiciones, no obstante, datan de 2009 y 2011 respectivamente. ¿Es posible que más de 10 años después no hayan perdido vigencia? Como seña y para poder comprobarlo, vamos a centrarnos ahora en definiciones más actuales dadas por otras compañías y organismos de referencia.

Comenzaremos por la definición dada por una de las empresas tecnológicas más importantes del mundo, Microsoft (Finn, A., 2012). Para este gigante, Cloud Computing es:

> *"Dicho de forma sencilla, la informática en la nube es el suministro de servicios informáticos (incluidos servidores, almacenamiento, bases de datos, redes, software, análisis e inteligencia) a través de Internet ("la nube"), cuyo objetivo es ofrecer una innovación más rápida recursos flexibles y economías de escala. Lo habitual es pagar sólo por los servicios en la nube utilizados, de tal forma que ayude a reducir los costos operativos, a ejecutar la infraestructura con más eficacia y a escalar a medida que cambian las necesidades de su negocio."*

Por último, nos parece interesante citar también la definición dada por Google y que dice lo siguiente sobre el término (Google, 2021):

> *"En cloud computing, la inversión de capital destinada a crear y mantener centros de datos se sustituye por un servicio flexible de un proveedor de servicios en la nube que ofrece recursos de TI, como almacenamiento, recursos de computación, redes, tratamiento y*

análisis de datos, desarrollo de aplicaciones, aprendizaje automático e incluso servicios totalmente gestionados."

Podríamos proporcionar muchas más definiciones del término, dadas por otros autores u organismos de referencia, pero todas ellas redundan en las mismas ideas y conceptos. Por tanto, una vez hecho este breve repaso por las definiciones más interesantes y completas del término, observamos que si bien existe cierta evolución en las capacidades que se les vislumbra al término, el objetivo perseguido y los aspectos más relevantes se mantienen vigentes. Parece interesante llegados a este punto elaborar una definición propia que, una vez estudiado el tema en profundidad, permita responder con claridad a la pregunta que se formula en el presente epígrafe: ¿Qué es Cloud Computing?

Podemos definir computación en la nube como un paradigma que permite acceder de forma remota a recursos hardware y software que provee un tercero por medio de una red, normalmente Internet. Este modelo posibilita que el cliente pueda realizar accesos remotos, ubicuos y dinámicos bajo demanda a recursos que gestiona un proveedor, mediante un mecanismo de acceso que debería ser transparente para el usuario, pudiendo reservar o liberar estos recursos según sus necesidades en todo momento.

Para terminar, parece importante remarcar y comprender que Cloud Computing, como se ha ido comentando, es un modelo que nace para cubrir una nueva necesidad en el mercado. Por esta razón en todas las definiciones aparecen tanto conceptos relacionados con la tecnología como relativos al modelo de negocio que se cimenta a su alrededor. Son vertientes indivisibles de un paradigma que lo ha transformado todo.

2.1.1. Características de Cloud Computing

Una vez abordada la conceptualización de la computación en la nube, nos encontramos en disposición de plantear, enumerar y comprender las características

más destacables del modelo. La literatura sobre este tema es, al igual que ocurre con la definición, muy amplia, aunque todas coinciden en unas ideas básicas (Murazzo, M. A., 2010). De la propia definición del término pueden extraerse algunas particularidades, que se sintetizan en los puntos siguientes (ver ilustración 1):

- **Autoservicio bajo demanda:** los recursos de cómputo son accesibles por medios propios, sin necesidad de una acción humana por parte del proveedor del servicio. Esto es especialmente importante para garantizar un acceso rápido y sencillo a estos recursos.
- **Amplio acceso a los servicios:** son accesibles a través de múltiples plataformas como pueden ser un PC, un teléfono móvil o una tablet.
- **Elasticidad y rapidez:** transparencia en cuanto a ubicación de los recursos que se proveen o la forma en que se asignan o reasignan entre los diferentes clientes en función de la demanda.
- **Bajo coste:** el coste de los recursos en la nube suele ser menor a los que una infraestructura física local podría suponer.
- **Servicio medido:** los múltiples servicios que oferta la nube, y de los que hablaremos más adelante, cuentan con diversos mecanismos de medición. Esto permite que su uso pueda ser monitoreado.
- **Seguridad mejorada:** los proveedores suelen ofrecer copias de seguridad automáticas, de modo que se dificulte la pérdida de información. La seguridad y mantenimiento de los recursos recae ahora sobre el proveedor del servicio, siendo como norma general totalmente trasparente para el cliente.

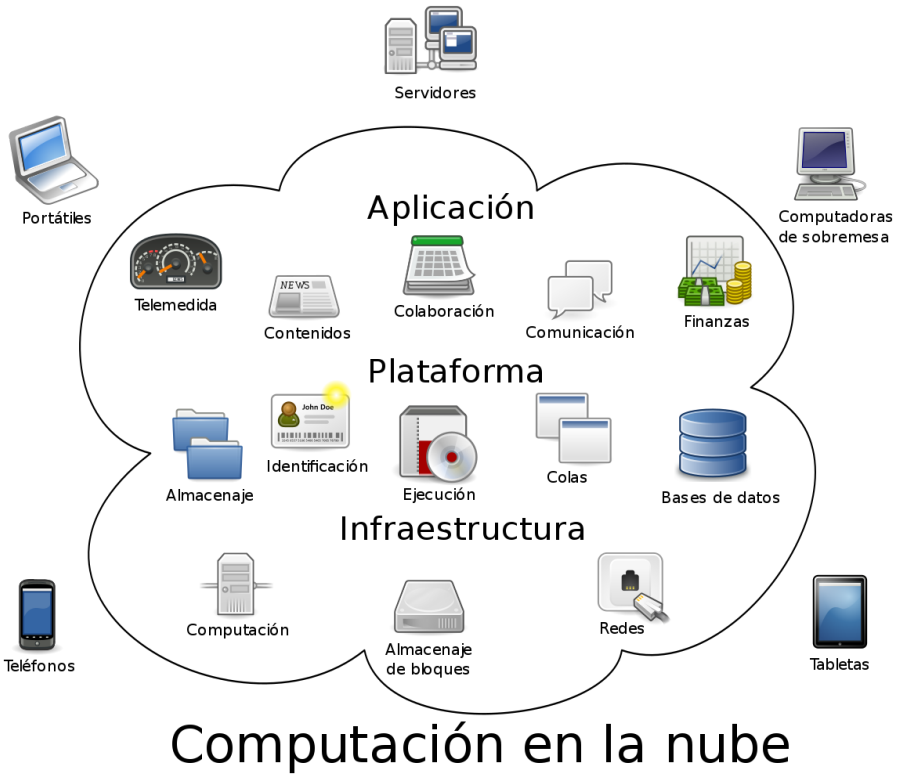

Ilustración 1. Diagrama visual de la computación en la nube.

Fuente: https://es.wikipedia.org/wiki/Computaci%C3%B3n_en_la_nube

2.2. Evolución a lo largo del tiempo

Llegados a este punto, es posible que la narrativa del término pueda llevar a confusión. Por lo visto hasta ahora podría extraerse que Cloud Computing es un paradigma que ha surgido hace pocos años y que su origen es muy reciente. Nada más lejos de la realidad (Sainz, C. C., 2014).

Los primeros indicios del término son algo confusos. Tanto es así que no toda la literatura sobre el tema coincide en el origen o primera aparición. Algunos autores consideran a Joseph Carl Robnett Licklider (Licklider, J. C. R., 1962) el padre de la computación en la nube gracias a su influyente proyecto llamado ARPANET en

1960, cuyo objetivo era conectar personas de todo el mundo. Sin embargo, otros le atribuyen este mérito a John McCarthy (Neto, P., 2011), quien durante el centenario del MIT en 1961 planteó la idea de un futuro en el que tanto poder de cómputo como aplicaciones se comercializarían como servicios, cambiando por completo el paradigma de computación de la época (ver ilustración 2).

A partir de aquí y a lo largo de la década de los sesenta, la idea de alquilar aplicaciones y/o tiempo de computación se hizo cada vez más popular, hasta el punto de que fueron varias las empresas que intentaron proporcionar servicios de computación basadas en esta novedosa idea. Algunas de ellas fueron Tymshare, National CSS y Dial Data (Cruz Valencia, K., 2012). Pero como toda historia que resulta demasiado bonita para ser cierta, las limitaciones técnicas de la época dificultaron mucho su uso y expansión, y poco a poco quedó relegado a un segundo plano respecto a otros avances tecnológicos.

Para encontrar el siguiente hito relevante en la evolución de este concepto debemos trasladarnos a tres décadas más tarde. En 1996 los ejecutivos de la compañía Compaq llamados George Favaloro y Sean O´Sullivan (Daylami, N., 2015) presentaron un plan de negocio donde se usó por primera vez en la historia el término cloud computing.

Avanzamos hacia finales de los años 90 cuando Salesforce, una de las primeras empresas en desarrollar el concepto, comenzó a ofrecer su software como servicio (SaaS), concepto que exploraremos en detalle más adelante (apartado 2.3.3. SaaS: Software as a Service). Salesforce introdujo servicios de aplicaciones empresariales a través de la web.

Posteriormente, en el año 2006, Amazon desarrolla el modelo de Infraestructura como servicio, también conocido por sus siglas IaaS, y lanzó Elastic Compute Cloud (Amazon EC2). En 2007, Google, IBM y varias universidades estadounidenses siguieron su ejemplo. Microsoft lanzó Azure en 2009.

Más tarde, en julio de 2010, la NASA y Rackspace lanzan juntos OpenStack (Sefraoui, O., 2012), proyecto software de código abierto en el que también participan gigantes tecnológicos como AMD, Intel y Dell. Actualmente el proyecto es una de las plataformas de cómputo más importantes del panorama y es administrado por la organización sin ánimo de lucro OpenStack Foundation.

El último hito que comentaremos en este epígrafe ocurre en el año 2012, cuando Google lanza Google Compute Engine. Esta herramienta aprovecha la infraestructura global que utiliza el motor de búsqueda de algunos de los servicios más relevantes de Google (Google, Gmail, YouTube y otros servicios). La principal aportación de Google Compute Engine radica en la posibilidad que ofrece a un usuario de desplegar una máquina virtual, ofreciendo de este modo un modelo de acceso bajo demanda. Podemos ver de un modo más gráfico toda la evolución que hemos tratado y estudiado en este punto en la ilustración 2.

Actualmente, en pleno año 2023, contamos con una gama enorme de servicios en la nube, tanto gratuitos como de pago, y se han integrado en nuestras vidas de modo que ya forman parte de nuestro día a día (Díez Álvaro, J., 2012).

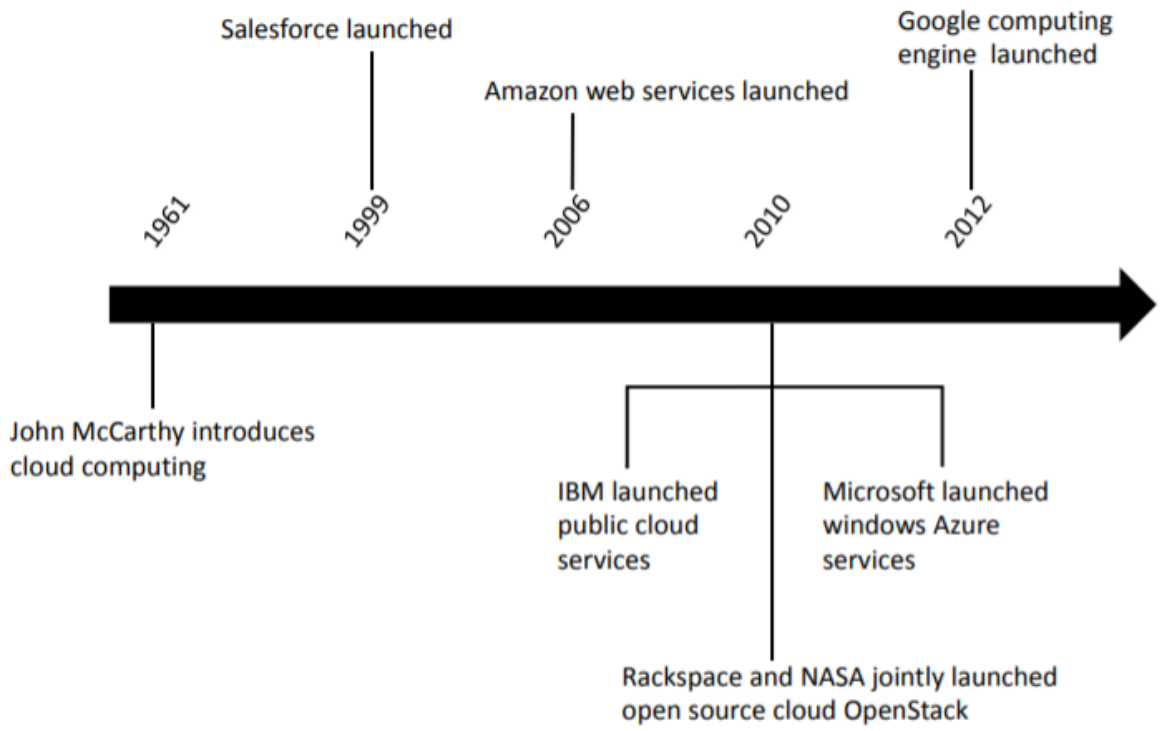

Ilustración 2. Cronología histórica Cloud Computing.

Fuente:
https://www.semanticscholar.org/paper/Cloud-Computing%3A-History-and-Overview-Surbiryala-Rong/

2.3. Paradigmas o modalidades de servicio

De lo visto y estudiado hasta ahora, podemos extraer que el modo en que un proveedor presta sus servicios no es único. No existe un enfoque único, de igual modo que la demanda que un cliente realiza no lo es. Podemos decir que existen por tanto, varios modos o modelos en la nube de prestación de servicios, si bien como veremos a continuación, se basan unos en otros.

Las definiciones clásicas del paradigma engloban los servicios y recursos en tres grandes categorías, entre las que se encuentra el objeto de estudio principal de este

proyecto. No obstante en los últimos años, y según evoluciona el mundo y la tecnología, han surgido varias vertientes más que buscan dar solución a nuevas necesidades y que también es muy interesante estudiar.

2.3.1. IaaS: Infrastructure as a Service

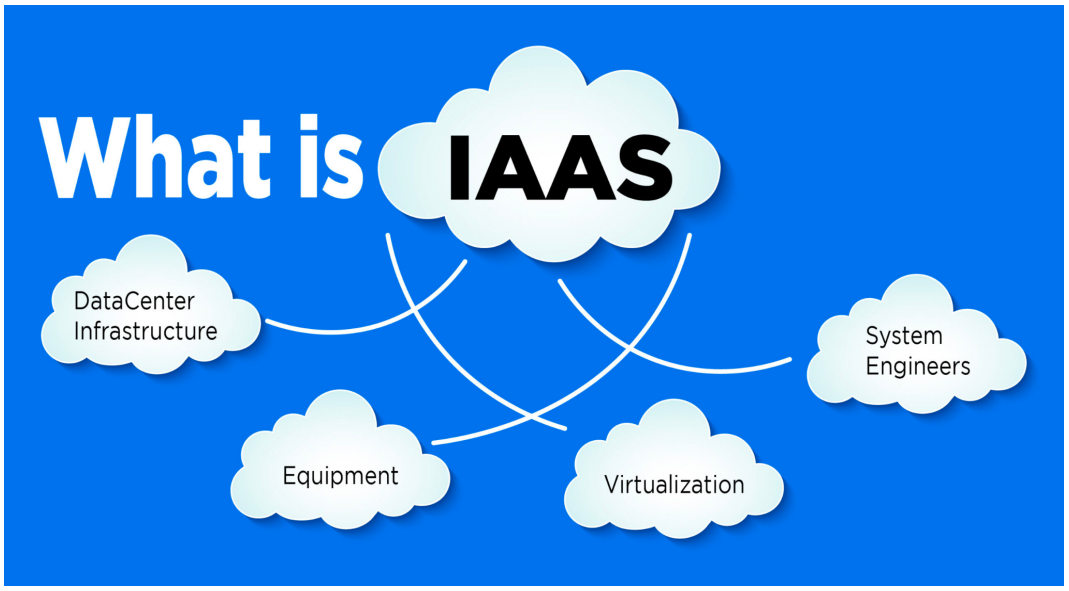

Ilustración 3. ¿Qué es Infrastructure as a service?

Fuente:
https://www.edubridgeindia.com/blog/understanding-infrastructure-as-a-service-in-cloud-computing/

Comenzaremos hablando del modelo infraestructura como servicio (IaaS), definido visualmente en la ilustración 3. Se trata de la categoría más básica o de más bajo nivel de todas y en ella un proveedor alquila infraestructura informática al usuario, tales como servidores, almacenamiento, redes, sistemas operativos, virtualización, etc. y la ofrece como servicio en la red, normalmente Internet (Dawoud, W., 2010).

Se trata del modelo que más similitudes presenta con una simple adquisición de hardware tradicional, con la gran y resaltable diferencia de que el alojamiento, la administración y el mantenimiento de toda esta infraestructura recae en el proveedor del servicio.

Estas empresas proveedoras cuentan normalmente con sus propios centros de datos, donde alojan todo el hardware que ofertan, de modo que su funcionamiento interno sea totalmente transparente para los usuarios. De esta forma los proveedores de servicio se pueden permitir la licencia de garantizar a sus clientes el acceso a un gran volumen de infraestructura en función de sus necesidades, principio básico del modelo.

Pero, ¿es realmente todo lo anterior condición suficiente para justificar su uso? Para responder a esta pregunta, ahondaremos un poco en su funcionamiento y aspectos más internos. Implementar una solución de Infraestructura como servicio presenta múltiples beneficios. Quizás la más llamativa y significativa sea el ahorro económico que supone externalizar toda la infraestructura informática, como ya se ha comentado anteriormente, con el consiguiente ahorro en adquisición, instalación y administración del hardware, que queda relegado al proveedor; pero ésta no es si no la punta del iceberg.

Toda la infraestructura ofertada en el modelo IaaS es escalable, de modo que una vez contratado el servicio, los recursos que ofrece pueden escalarse verticalmente al ritmo que el proyecto evoluciona. Esto proporciona una enorme flexibilidad al usuario, permitiendo adecuar el entorno y dar soporte al volumen de trabajo y a las necesidades de cada momento, de forma totalmente dinámica.

Otro aspecto reseñable que ofrece esta primera categoría de modelos de servicio es la seguridad. Los proveedores instalan toda su infraestructura en inmensos centros de datos que cuentan con estrictas medidas de seguridad, normalmente más elevadas que las que un usuario pueda instalar en su centro local.

¿Quiere esto decir entonces que es la solución perfecta? El modelo tiene algunos problemas y desventajas que parece interesante comentar. Si bien la cesión de la responsabilidad sobre el nivel físico de la infraestructura tiene numerosas ventajas como se ha comentado, también conlleva un problema. El usuario depende totalmente del proveedor en términos de disponibilidad y seguridad. Además, al ser un servicio que se oferta a través de una red, Internet en la mayoría de casos, cualquier problema de conexión repercutirá negativamente en el acceso.

Por último, para clarificar aún más el concepto y a modo de conclusión, comentaremos aquellas situaciones y proyectos para los que el modelo infraestructura como servicio resulta más idóneo. No todos los tipos de empresas o propósitos son adecuados para elegir esta categoría, habiendo otras alternativas que se ajustan mejor a las necesidades propias de cada tipo de proyecto o situación.

- Un primer tipo de proyecto que se ajusta perfectamente a las especificaciones del modelo, es el **desarrollo y posteriores pruebas de una aplicación**, ya que permite escalar y reducir entornos de desarrollo de forma rápida y económica.
- Cuando la **demanda de infraestructura no es estable**, y existen picos significativos a lo largo del tiempo.
- Organizaciones que acaban de empezar sin capital suficiente para invertir en una infraestructura propia, como una **startup**, u **organizaciones que han sufrido un crecimiento acelerado**.
- El **alojamiento y despliegue de proyectos web de gran envergadura** también se beneficia de la flexibilidad del servicio.
- Por otro lado, tareas tales como la **computación de alto rendimiento** o el **análisis de macro data** se benefician enormemente de las capacidades de escalado del modelo.

2.3.2. PaaS: Platform as a Service

Ilustración 4. ¿Qué es Platform as a service?

Fuente: https://bestarion.com/what-is-platform-as-a-service-paas-types-and-use-cases/

El segundo modelo que estudiaremos será plataforma como servicio (PaaS), definido visualmente en la ilustración 4, además objeto de estudio principal del presente trabajo. Se trata de una categoría intermedia entre IaaS y SaaS, en la que un proveedor ofrece una plataforma para desarrollar, ejecutar y administrar desde aplicaciones sencillas hasta complejas aplicaciones empresariales (Lawton, G., 2008), sin que el cliente o usuario deba preocuparse de diseñar y mantener toda la infraestructura y la plataforma que normalmente suele estar asociada al lanzamiento de una aplicación.

Al igual que ocurría con el modelo estudiado en el punto anterior, infraestructura como servicio, este modelo también ofrece toda la infraestructura que el cliente necesita (servidores, almacenamiento, redes, virtualización, etc.). La diferencia entre

ambos radica en que plataforma como servicio incluye, además de todo lo anterior, un paquete de herramientas para empezar a desarrollar aplicaciones de inmediato, mediante aspectos como middleware, autenticación, mensajería, sistemas de administración de bases de datos, las herramientas de desarrollo y los servicios de inteligencia empresarial (BI), etc. Esto permite al usuario centrarse únicamente en la construcción de la aplicación, pues el proveedor se ocupa tanto de las tareas de gestión como de las de mantenimiento de toda la infraestructura y plataforma.

Otra diferencia respecto a IaaS radica en la dificultad inherente a la construcción de aplicaciones y la administración de la plataforma. PaaS gestiona la escalabilidad de forma automática, haciendo uso de los recursos estrictamente necesarios en cada instante de tiempo.

Una vez desarrollado el concepto en cuanto a definición y principales características cabe preguntarse, ¿qué aspectos convierten el paradigma plataforma como servicio en una buena solución para un particular o una empresa?

PaaS ofrece multitud de ventajas a los desarrolladores. Cómo he comentado anteriormente, también ofrece infraestructura como servicio, lo que aporta al desarrollador las mismas ventajas que la modalidad IaaS, por lo que los motivos que esgrimiamos anteriormente en el punto relativo a IaaS son igualmente válidos aquí. El principal de ellos, que evita tener que adquirir y gestionar una infraestructura propia.

No obstante, y como se ha visto a lo largo de toda esta subsección, PaaS cuenta con numerosas características adicionales. La principal es que, como se ha comentado más arriba, reduce significativamente el tiempo de programación. Cuenta con componentes de aplicación perfectamente integrados en la plataforma que permiten al desarrollador centrarse única y exclusivamente en la tarea de dar lógica a su código, acortando los tiempos en gran medida.

Y precisamente de estas herramientas de desarrollo adicionales que suelen venir preprogramadas, se desprenden gran parte de las características que convierten PaaS en la solución idónea de algunos clientes y empresas. Comenzando por la posibilidad de trabajar en múltiples plataformas de forma ágil y sencilla, gracias a las opciones de desarrollo para varias plataformas que suelen incluir este servicio.

De igual modo permite al desarrollador acceder a herramientas tan sofisticadas y prohibitivas económicamente, que de otro modo le sería muy difícil permitirse, o usar herramientas y funcionalidades, a las que no podrían acceder sin la incorporación de personal cualificado en el tema.

Y para concluir con la respuesta a la pregunta que nos atañe, quizás uno de los principales motivos que llevan a un cliente a contratar este servicio es que ofrece todos los elementos necesarios para sustentar el ciclo completo de vida de una aplicación, sin que el desarrollador deba prestar la más mínima atención a ningún aspecto fuera de la lógica de su proyecto. Cualquier proveedor de PaaS suele proporcionar herramientas de compilación, un portal para realizar todas las pruebas necesarias, un entorno para desarrollar, etc.

¿Significa esto que se ha encontrado el santo grial tecnológico? Como decíamos en la subsección anterior enfocada a IaaS, a pesar de todas las ventajas que las soluciones basadas en este paradigma aportan, nos encontramos con algunos aspectos negativos que es interesante estudiar para ser tenidos en debida consideración.

De nuevo aquí y debido a que incluye infraestructura como servicio, contiene los problemas que se comentaban en su sección. La disponibilidad y por supuesto que la seguridad depende del proveedor del servicio, y el desarrollador no tiene casi ningún control sobre ellas. Esta dependencia puede suponer un gran perjuicio ante la existencia de cualquier problemática.

Otra singularidad susceptible de convertirse en un problema es que puede no ser tarea trivial integrar las aplicaciones creadas con este servicio con las posibles aplicaciones desarrolladas e instaladas de forma local que el cliente posea en su equipo.

Para concluir con esta segunda subsección, de igual modo que hicimos con IaaS y en esta ocasión con más motivo aún debido a la importancia del punto en el proyecto, finalizaremos exponiendo aquellos proyectos y empresas que mejor se amoldan al paradigma PaaS. Como ya dijimos en el punto anterior no todos los proyectos se pliegan de igual forma a todas las modalidades de servicio que Cloud Computing aporta.

- Los **proyectos válidos para IaaS, son igualmente válidos** en esta segunda modalidad. La razón ya se ha aportado a lo largo de la presente subsección.
- Proyectos que cuenten con **varios desarrolladores trabajando en concurrencia en el mismo proyecto,** que comprenden la inmensa mayoría de proyectos empresariales de la actualidad.
- La explosión del **Internet de las cosas** ofrece un interesante lugar que PaaS podría utilizar para crecer. Debido a la flexibilidad , facilidad y rapidez que aportan, los entornos PaaS suponen una opción muy interesante para desarrollar aplicaciones que puedan utilizarse para el Internet de las cosas.

2.3.3. SaaS: Software as a Service

Ilustración 5. ¿Qué es Software as a service?

Fuente: https://www.softzone.es/2018/08/06/software-saas-ventajas-aportar/

El tercer gran ítem que la literatura más clásica del término aborda es software como servicio (SaaS), definido visualmente en la ilustración 5. Se trata de la última categoría de más alto nivel y, como tal, es la que engloba y ofrece una mayor cantidad de servicios (Ma, D., 2007). En ella el proveedor de servicios es el encargado de desarrollar y por consiguiente mantener el software de las aplicaciones en sus propios servidores y los ofrece a los clientes de modo que puedan utilizarlas.

Del mismo modo que ocurría en los modelos anteriores, el cliente no gestiona ni administra la infraestructura, los servidores, sistemas operativos, almacenamiento, seguridad, etc. Añadiendo en este tercer caso todo lo relativo al software de la aplicación. En este modelo por tanto, lo que se oferta es software perteneciente al proveedor, al que los clientes podrán acceder vía Internet en cualquier dispositivo, sin necesidad de realizar ninguna instalación local.

Una amplia mayoría de usuarios de Internet han usado y usan aplicaciones que un proveedor ha puesto a disposición pública generalmente a través de la red. Tenemos muchos ejemplos de ellos, siendo algunos de los más comunes: correo electrónico, calendarios, herramientas ofimáticas, etc. Pero aún más conocidos, y en pleno auge en la actualidad, son las populares aplicaciones de streaming. Ejemplos como Spotify dentro del mundo musical, y Netflix o HBO dentro del mundo audiovisual, son el mejor exponente de cómo prácticamente todos los usuarios de Internet interaccionan y usan el modelo software as a service a diario.

Todos estos ejemplos, no obstante, son servicios que diferentes proveedores ofrecen para uso personal. En el caso de organizaciones y empresas existen igualmente aplicaciones empresariales sumamente sofisticadas, como CRM, ERP o aplicaciones de correo electrónico bastante populares.

Como observamos, en este caso los aspectos que llevan a un cliente a contratar y utilizar aplicaciones que se encuadran dentro de este modelo y que lo convierten una buena solución son fácilmente extraíbles de su propia definición y listado de características.

Adicionalmente, las cualidades que aporta este nuevo paradigma con respecto al anterior, lo dotan de múltiples ventajas muy apreciables para los particulares y empresas que lo usan. Para comenzar, como se ha desglosado a lo largo de la presente subsección, este paradigma posibilita que un usuario pueda acceder a software muy sofisticado sin tener que preocuparse de ningún aspecto en cuanto a su desarrollo, sólo su utilización, a precios razonablemente más bajos de lo que supondría un desarrollo propio.

Además la peculiaridad de SaaS provoca que la puesta en funcionamiento y prestación de servicio para el cliente sea muy rápida. Esto se debe a que el cliente solo debe “conectarse” para comenzar a trabajar, en lugar de tener que instalar software o preocuparse de licencias. Adicionalmente permite el acceso a sus aplicaciones desde cualquier dispositivo y en cualquier lugar con acceso a Internet.

¿Tiene algún aspecto negativo este tercer paradigma? Los inconvenientes y problemas que provoca, son bastante reducidos, aún así parece conveniente tratarlos y estudiarlos.

Los principales problemas asociados a este paradigma están asociados a la tecnología Cloud Computing, y derivan a los modelos de servicio anteriores. Los peligros derivados de la cesión de responsabilidad en cuanto a infraestructura, la cesión de datos a los proveedores o la dependencia de una conexión segura y constante a Internet se postulan como los conflictos más significativos.

Finalmente y para concluir, de igual modo que hicimos con los dos paradigmas anteriores, listaremos aquellas situaciones y proyectos que se antojan idóneos para el modelo SaaS.

- Aplicaciones que implican una **interacción significativa entre usuario y mundo exterior**, suponen un contexto donde SaaS se mueve de manera óptima.
- Proyectos que necesitan de **un software en específico que solo será usado en esa situación**. El ahorro económico que supone frente al desarrollo de un software propio es significativo.
- Aplicaciones cuyo **picos máximos y mínimos de uso varían** irregularmente y con frecuencia en el tiempo.

Vemos ahora una comparativa entre los 3 modelos estudiados en este epígrafe en la ilustración 6.

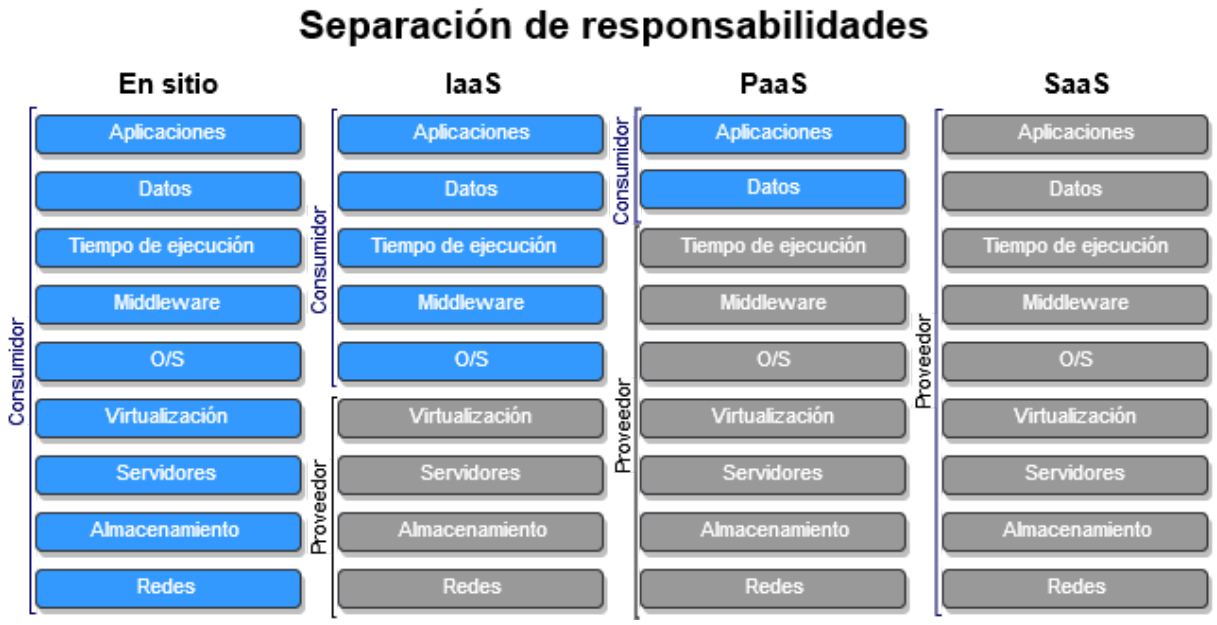

Ilustración 6. Diagrama explicativo sobre los aspectos que gestionan cliente y proveedor tanto en on-site como en cada uno de los modelos de servicio.

Fuente: https://blog.bi-geek.com/cloud-computing-modelos-de-servicio/

2.3.4. Otras modalidades de servicio

Una vez vistas y estudiadas en profundidad las, según la amplia mayoría de bibliografía sobre el tema, tres principales modalidades de prestación de servicio, nos encontramos en disposición de abordar de un modo más resumido los nuevos modelos y paradigmas surgidos en los últimos años.

La lista de nuevos modelos de prestación de servicios es interminable, de la misma forma que la diversidad de problemas a solucionar y necesidades lo es, de modo que aquí nos ceñiremos a explorar algunas de las más relevantes y extendidas, ya sea por sus bondades o su propia naturaleza.

- **iPaaS (Integration Platform as a Service)**

Gartner Inc., empresa líder a nivel mundial en consultoría e investigación en el mercado de las nuevas tecnologías, aporta la siguiente definición (Marian, M., 2012):

> *"Conjunto de servicios en la nube que permite el desarrollo, la ejecución y el gobierno de los flujos de integración que conectan cualquier combinación de procesos, servicios, aplicaciones y datos locales y basados en la nube dentro de una organización individual o en múltiples organizaciones."*

Esta definición del término clarifica bastante su conceptualidad. Dicho de un modo más llano, se trata de una plataforma tecnológica basada en cloud que se utiliza para integrar y conectar las diversas aplicaciones, servicios y datos que utiliza y almacena una organización.

Una solución iPaaS permite a una organización conectar los servicios que utiliza en la nube, con aquellos que almacena de forma local, de forma sencilla. Lo mismo ocurre con los datos y la información que genera.

Esta dualidad, cada vez más extendida ante la complejidad del sistema y el entorno, genera como hemos comentado en los párrafos introductorios, una necesidad. La integración se vuelve cada vez más complicada. El rápido crecimiento y la volubilidad del entorno provoca que los costes para las organizaciones de esta integración sean cada vez más elevados. Un nuevo modelo que realice esta función de modo más rápido y eficiente se hacía cada vez más necesario.

- **FaaS (Functions as a Service)**

El modelo Función como Servicio (FaaS) se encuentra íntimamente relacionado con el concepto de la computación sin servidor (Rajan, A. P., 2020). De modo que antes de entrar a valorar este nuevo modelo la cuestión que se plantea es, ¿qué es la computación sin servidor, más conocida como arquitectura serverless?

El nombre del concepto suele llevar a equívoco. No se trata de tecnología sin servidor, tal cual significan estas palabras. Bajo este paradigma tanto las aplicaciones como los servicios se ejecutan en contenedores temporales y sin estado, que se inician y ejecutan cuando ocurre un evento, y que desaparecen cuando la aplicación deja de ejecutarse. El servidor por tanto se convierte en una parte invisible para el usuario.

Bajo esta idea subyacente nace FaaS, que se nutre de todas estas características y ventajas y las pone a disposición del usuario en este modelo de prestación de servicio.

- **CaaS (Container as a Service)**

Container as a Service es otro modelo que surge ante la necesidad de cubrir nuevas expectativas (Xie, X., 2018). Consiste en una abstracción basada en contenedores, que permiten a un usuario desarrollar y administrar sus propias aplicaciones y servicios a través de ellos.

El concepto como observamos es muy sencillo. No obstante cabe preguntarse qué son exactamente estos contenedores a los que esta definición del modelo hace referencia.

Un contenedor, en el caso que nos ocupa, es una unidad de software que empaqueta el código de una aplicación y posibilita que se ejecute de forma rápida y segura. Esta encapsulación, y aquí radica su razón de ser y una de sus más célebres ventajas, permite crear software fácilmente portable entre distintos entornos informáticos.

En el esquema clásico que vimos en las subsecciones anteriores (IaaS, PaaS y SaaS), Caas podría situarse como un modelo intermedio entre IaaS y PaaS, si bien como hemos visto el concepto de virtualización utilizado en este caso difiere sobremanera respecto a los dos anteriores.

- **STaaS (Storage as a Service)**

Nos encontramos ahora, con uno de los paradigmas más básicos a la vez que útiles, y por tanto extendidos del panorama. El modelo STaaS surge bajo la demanda de organizaciones sin grandes recursos económicos de capacidades de almacenamiento bajo demanda a precios más competitivos. El funcionamiento es muy simple, un proveedor ofrece recursos de almacenamiento, de modo que el cliente accede a los recursos que necesita en cada momento y bajo demanda.

- **SECaaS (Security as a Service)**

Como un sucedáneo del modelo Software as a Service estudiado en el capítulo anterior, surge la modalidad SECaaS. La seguridad como servicio ofrece servicios administrados de seguridad a través de la nube. Como decimos, está basado en el modelo SaaS, siguiendo todas sus especificaciones, si bien en este caso restringe su campo de acción a servicios de seguridad.

Los servicios que oferta son de índole muy diversa, siempre dentro del ámbito de la seguridad y van desde un simple antivirus en Internet, hasta complejas soluciones de seguridad más avanzadas.

2.4. Modelos de despliegue

Esta segunda clasificación que abordaremos se enfoca principalmente en la ubicación y el enfoque de gestión de la infraestructura en la nube de manera generalizada.

Los modelos de despliegue de la nube se encuadran en cuatro grandes bloques estándar (Hernández, N. L., 2014; Stackscale S.L., 2021), aceptados por la amplia mayoría de literatura sobre el tema. Aunque de igual modo a como ocurre con la clasificación anterior de los modelos de servicio, otras opciones se han ido incorporando a medida que la tecnología crecía para cubrir nuevas necesidades de una red de usuarios cada vez más amplia. Las diferencias entre ellas radican en la

propiedad de los recursos y las infraestructuras, y en quién será el encargado de su gestión y mantenimiento. El uso de uno u otro dependerá de la naturaleza del proyecto que se esté llevando a cabo, no siendo ninguno de ellos superior al resto de un modo absoluto. Así pues, en referencia a los modelos de despliegue, podemos hablar de nube pública, nube privada, nube híbrida y nube comunitaria (ver ilustración 7).

Ilustración 7. Diagrama comparativo entre los distintos modelos de despliegue.

Fuente: https://urgente24.com/zona-/nube-hibrida-la-suma-lo-publico-y-lo-privado-sirve-n538537

2.4.1. Nube pública

Comenzaremos hablando del modelo nube pública. En este paradigma un proveedor ofrece una serie de servicios informáticos e infraestructura compartidos con más clientes a través de Internet, ya sea para un público general o un sector acotado.. El

proveedor de servicio será el responsable único de todo el trabajo de administración y mantenimiento del sistema.

Bajo este modelo, los usuarios pueden escalar el uso de la nube bajo demanda, prácticamente sin limitaciones. Si a esto añades que, como se ha comentado, los recursos ofrecidos están disponibles para cualquier persona con conexión a la red, las posibilidades son enormes.

La nube pública permite el acceso de los usuarios a la nube a través de interfaces que utilizan navegadores web. Los usuarios deben pagar sólo por el tiempo que utilizan el servicio, es decir, pago por uso. Esto se puede comparar con el sistema eléctrico que recibimos en nuestros hogares. Pagamos solo por la cantidad que usamos. Aquí se aplica el mismo concepto.

Esta tipología de nube permite además, y esto es muy interesante, almacenar grandes cantidades de información en hardware perteneciente al proveedor de servicio. Obviando las repercusiones a nivel de confidencialidad y la dependencia que genera (aspectos sobre los que ya hemos hablado en la clasificación anterior), la nube pública permite al cliente aumentar y ampliar sus recursos a precios realmente bajos en comparación al modelo tradicional.

Las empresas proveedoras que ofertan este servicio suelen construir sus centros de datos e infraestructura en regiones muy distantes del espacio, y suelen contar con duplicados. De esta forma, consiguen reducir los problemas de disponibilidad de los servicios o posibles fallas en la infraestructura que provoquen un corte del suministro de servicio.

Llegados a este punto, de igual modo que hicimos en los paradigmas de la clasificación anterior, nos preguntamos, ¿qué aspectos y características de la nube pública la convierten en una solución atractiva? Las ventajas para el usuario que la nube pública aporta son incuestionables.

La primera y más llamativa es que se trata de la más sencilla de implementar. El proveedor provee, administra y gestiona toda la infraestructura necesaria, y es responsable de la seguridad tanto de los datos como de las aplicaciones. Por lo tanto los usuarios solo deben preocuparse de usar los servicios que este proveedor les proporciona.

Del punto anterior se extrae que bajo este paradigma no es necesario invertir en infraestructura o personal encargado de administrar y gestionar el sistema. Esto deriva en una significativa reducción de costes y convierte este modelo en una opción muy interesante cuando el presupuesto es reducido.

Para finalizar con la pregunta, observamos que la nube pública es considerablemente flexible en comparación a otros modelos. Para clarificar este aspecto pondremos como ejemplo un cliente que desea almacenar sus datos en un servidor. En un principio necesitará poco espacio, pero quiere aumentarlo progresivamente. La nube pública permite esta casuística debido, como comentamos, a su flexibilidad.

Una vez establecidas las ventajas que aporta el modelo, parece interesante contrastar los problemas que acarrea para los usuarios. El principal de ellos deriva de la compartición de recursos con otros clientes de la que hablábamos en el primer párrafo de la presente subsección. Al compartir servidores y red con otros clientes que están usando los mismos servicios, los datos y la información son más susceptibles de quedar expuestos.

Otras desventajas de menor relevancia, se desprenden de, aspecto que ya hemos comentado varias veces a lo largo de este documento, la dependencia que genera delegar toda responsabilidad en el proveedor de servicios. El cliente no tiene control alguno sobre aspectos como la seguridad o disponibilidad del servicio, lo que supone un riesgo.

Cabe mencionar aquí, que las nubes públicas pueden y suelen complementarse con otras herramientas que ayudan a paliar significativamente los problemas y defectos que comentamos. Servicios de back up o seguridad adicional son algunos ejemplos.

2.4.2. Nube privada

En contraposición a la nube pública, encontramos la nube privada. En este segundo paradigma, sólo una organización, mediante diversas tecnologías tales como la virtualización, tiene acceso a la infraestructura, servicios o aplicaciones de la nube en cuestión. Por decirlo del modo más llano posible, un cliente dispone de forma exclusiva de la nube.

La infraestructura necesaria para soportar este servicio, puede estar ubicada dentro de las propias instalaciones del cliente en un Centro de Proceso de Datos (CPD) o bien en un espacio reservado dentro del centro de datos del proveedor de servicios (Pérez, C. F. V., 2017). A esta infraestructura, así como a la plataforma y servicios que ofrece, acceden solo los miembros de la organización, así como las personas que obtengan permisos de la misma.

Como veremos a continuación en el espacio que estamos reservando en cada subsección de modalidades para conocer las ventajas que posee el modelo, la nube privada proporciona algunas de las ventajas más provechosas de la nube pública, agregando otras diferentes. Esta dualidad convierte esta alternativa en un servicio muy valorado dentro de ciertos sectores.

¿Cuáles son las ventajas que ofrece la nube privada? Como hemos comentado, aporta a los clientes, que suelen ser grandes empresas, algunas de las grandes ventajas que comentamos de la nube pública, ventajas como autoservicio, escalabilidad, elasticidad, etc. Y adicionalmente, debido a sus propias características, aporta algunas más que resulta interesante igualmente estudiar.

La principal y más evidente, es que elimina la problemática de las nubes públicas en cuanto a privacidad y seguridad. La nube privada suele contar con firewalls que protegen la nube del exterior, y mantiene los datos de la organización a salvo de inferencias, de modo que no sean accesibles para personas ajenas a la compañía o para proveedores externos.

Además, el hecho de que sea accesible solo por miembros de la organización también aporta un mayor control sobre los recursos, que estarán configurados a medida del cliente. Lógicamente esta característica aporta numerosos beneficios a una empresa, permitiendo optimizar sus procesos de modos que sería difícil alcanzar sin esta solución de red personalizada.

Para terminar con las ventajas que aporta el modelo, la nube privada aporta una mayor fiabilidad, permitiendo y asegurando que servicios y aplicaciones críticas para el organismo tengan garantizado su funcionamiento las 24 horas del día.

Pero como no es oro todo lo que reluce, el uso de una nube privada implica algunos problemas que es conveniente tratar. Las nubes, ya sean públicas o privadas, necesitan unas tareas de gestión y administración muy específicas para preservar su correcto funcionamiento. Este mantenimiento, en la nube pública es llevado a cabo en su totalidad por el proveedor de servicios, que cuenta con todas las herramientas para optimizar dicha función. ¿Pero qué ocurre en las nubes privadas? Dicho mantenimiento queda a cargo de la empresa, con el inconveniente logístico y necesidad de asignar recursos económicos y humanos que ello implica.

Otro punto en contra respecto al modelo anterior, la nube pública, es la escalabilidad. Aunque aporta cierto grado de escalabilidad, no soporta la comparación respecto a su homólogo. El número de servidores de la nube pública, normalmente, es mucho más elevado que el que una empresa vaya a instalar o contratar de forma privada.

Para finalizar con las desventajas que aporta, podríamos apuntar la inversión inicial que requiere, mucho más elevada que en el caso de la nube pública, que es casi inexistente.

De todo lo anterior se desprende que los tipos de organizaciones y proyectos ideales para una solución de nube privada serán empresas que posean grandes cantidades de tecnología y recursos, tales como: entidades bancarias, administración pública, y empresas donde garantizar la localización de servidores, tener exclusividad de uso de las aplicaciones, etc. sean aspectos básicos.

2.4.3. Nube híbrida

Tras realizar este estudio de los dos modelos principales de despliegue, la cuestión inevitable que surge es cuál predomina sobre la otra. Como vemos, todo depende del propósito. Ambas poseen interesantes ventajas y numerosos inconvenientes. ¿No sería buena idea crear un nuevo modelo que aúne los aspectos positivos de uno y de otro? Precisamente bajo esta premisa surge la idea de nube híbrida.

Se trata de una combinación de nube pública y nube privada (VMware, s.f.). Este tercer modelo combina recursos de computación dedicados y recursos de computación públicos.

El objetivo, como se desprende de lo anterior, es unir lo mejor de ambos mundos para situaciones en que necesitas algunas de las virtudes de un modelo sin perder las ventajas que aporta el otro.

Por ejemplificar lo anterior, la nube híbrida permite aunar el control que aporta la nube privada con la flexibilidad que aporta la nube pública. En esta o similares casuísticas surge el modelo de nube híbrida.

2.4.4. Nube comunitaria

La última modalidad de despliegue de la nube que abordaremos en este escrito es la nube comunitaria (Condori, X. G., 2013). Es quizás el paradigma más desconocido para los poco avezados en el tema, si bien como veremos a continuación, posee características que lo convierten en una solución muy útil para diversas organizaciones y empresas.

Bajo este modelo la infraestructura tecnológica es compartida entre varias organizaciones, que normalmente guardan estrecha relación y similitudes entre ellas. De esta breve definición se desprende una de las ventajas que proporciona respecto a sus análogos compañeros. Esta modalidad pretende abarcar todas aquellas ventajas de las que hablábamos en los 3 modelos anteriores (público, privado e híbrido) a costes mucho más reducidos, al ser como decimos compartidos entre diversas empresas.

Un buen ejemplo de nube comunitaria, y que ayudará a comprender este concepto, es el proyecto "OpenStack" (Rosado, T., 2014). Se trata de una plataforma open source mediante la que varias organizaciones pueden compartir recursos. Además de esto, la plataforma permite a las organizaciones colaborar conjuntamente en el despliegue y gestión de infraestructuras en la nube. Bajo este modelo, un grupo de empresas u organismos pueden unirse para formar una nube comunitaria basada en OpenStack. La gran ventaja radica en que cada uno de ellos aporta sus recursos a la comunidad y obtiene beneficio completo de las capacidades y servicios.

3. PROVEEDORES DE CLOUD COMPUTING

Durante el desarrollo de este capítulo abordaremos el concepto de Cloud Computing desde una nueva perspectiva. A lo largo de todo el proyecto, hemos estudiado desde diferentes perspectivas qué es Cloud Computing, cuáles son sus principales características, principales ventajas que aporta respecto a otros modelos de computación… A estas alturas la pregunta que surge es, ¿quién oferta todos estos servicios y de qué modo? ¿Cuáles son los principales proveedores de servicios en la nube? Finalmente, este detallado análisis nos aportará las armas suficientes para poder elegir con el suficiente conocimiento el proveedor más idóneo para la realización de la aplicación práctica que incluye este documento.

3.1. ¿Qué significa ser proveedor de Cloud Computing?

Como veremos a continuación ser proveedor de Cloud Computing implica una serie de tareas que comprenden un abanico muy amplio de servicios y recursos de computación en la nube a través de una plataforma tecnológica. La magia reside en que estos proveedores ponen a disposición de sus clientes infraestructura, software y servicios totalmente alojados en la nube, ofreciéndoles la posibilidad de administrar y gestionar estos recursos, y aporta una gama de posibilidades, que van desde flexibilidad hasta escalabilidad, según las necesidades del cliente (Mustafa, O., 2023).

En resumidas cuentas, ser proveedor de Cloud Computing, enunciado que titula este epígrafe, consiste en dar acceso a los clientes a servicios, software e infraestructura alojados en la nube, brindándoles la oportunidad de sacar todo el provecho posible a las cuantiosas ventajas que la computación en la nube oferta (Nigro, H., 2022). Todo ello además, y aquí reside el quid de la cuestión, sin tener que gestionar la infraestructura subyacente, aspecto que como todos los que nos dedicamos a este mundo sabemos, en ocasiones puede resultar algo engorroso.

3.2. Principales proveedores en la actualidad

Existen multitud de proveedores de servicio de Cloud Computing en la actualidad, si bien, como ocurre con la mayoría de aspectos, hay una reducida lista de proveedores que lideran el mercado; algunos de ellos, prácticamente desde el nacimiento del concepto.

3.2.1. Amazon Web Service

Ilustración 8. Logotipo Amazon Web Service.

Fuente: https://aws.amazon.com/es/

Es casi obligatorio comenzar este análisis por el proveedor de servicios que se ha convertido en absoluto líder del mercado. Nacido en el año 2006 Amazon Web Service (ver ilustración 8), más conocido por su acrónimo AWS, provee una amplia gama de herramientas y servicios a sus clientes a lo largo y ancho de todo el mundo. Además, me parece importante resaltar aquí, que tener un gigante tecnológico respaldando tu proyecto, como ocurre en este caso con Amazon, es un punto muy importante que vale la pena tener en cuenta. Aunque como veremos a continuación, este es un punto que tienen en común la amplia mayoría de proveedores de servicio más relevantes del mercado prácticamente desde sus inicios.

Varios millones de clientes (algunos de ellos podemos verlos en la ilustración 9) en todo el mundo, abarcando además una gama que va desde empresas consolidadas como Netflix, Samsung o Toshiba a empresas emergentes cuyo objetivo es buscar crecer lo más rápido posible, utilizan AWS lo que nos dice que es vista en el mercado como un opción muy sólida y versátil, además de tener una madurez que otras alternativas de las distintas competencias no pueden ofrecer.

Ilustración 9. Principales clientes de AWS en la actualidad.

Fuente: https://www.frogriot.com/blog/pl/aws-czym-sa-amazon-web-services-qa/

Como muestra de lo anterior un revelador dato: La infraestructura de la nube de AWS está formada por 99 zonas de disponibilidad en 31 regiones geográficas que se distribuyen a lo largo de todo el mundo según datos de su propia web (Amazon Web Service, s.f.).

3.2.2. Microsoft azure

Ilustración 10. Logotipo de Microsoft Azure.

Fuente: https://azure.microsoft.com/es-es/

Y si de gigantes tecnológicos estamos hablando, este segundo ítem no podía estar dirigido sino al servicio ofrecido por la organización Microsoft (ver logotipo en la ilustración 10). Se trata de uno de los mejores competidores del mercado de AWS, y al igual que ocurría con éste, cuenta con una sólida estructura y una amplia variedad de servicios puestos a disposición de su larga lista de clientes. Destaca frente a otros proveedores de servicio por aspectos tan relevantes como una altísima escalabilidad y flexibilidad, alta disponibilidad, amplia variedad de herramientas y servicios, etc. Pero especialmente porque provee integración automática con todas las herramientas de Microsoft (Office 365, SQL Server, etc.), lo que lo convierte en

una alternativa muy a tener en cuenta para empresas que ya utilizan las herramientas de Microsoft en su día a día.

Microsoft Azure cuenta en la actualidad con más de 200 centros de datos físicos, organizados por regiones, igual que ocurre en AWS, según datos de su propia web (Microsoft, s.f.). El crecimiento en este aspecto en los últimos años ha sido sustancial, si bien es un esfuerzo que está dando sus frutos, colocando Azure como uno de los proveedores de servicio más relevantes del mercado como hemos comentado.

Cuenta en su cartera de clientes con algunas de las empresas más importantes del panorama, destacando empresas como Coca-Cola, Intel o BMW (ilustración 11 muestra algunos de ellos).

Ilustración 11. Principales clientes de Microsoft Azure en la actualidad.

Fuente: https://nexsysla.wishpondpages.com/abarca-necesidades-azure/

3.2.3. Google Cloud Platform

Ilustración 12. Logotipo de Google Cloud Platform.

Fuente: https://cloud.google.com/

Y podemos cerrar la trilogía de proveedores de servicio principales y, con una gran diferencia con otras alternativas, más relevantes del mercado, con Google Cloud Platform, cuyo logotipo observamos en la ilustración 12. Tras este proveedor nos encontramos con Google. De nuevo nos encontramos con una inmensa organización que provee un enorme abanico de herramientas y servicios a clientes que se extienden a lo largo de todo el globo.

La principal ventaja que aporta Google Cloud Platform respecto a sus competidores, radica en que permite diseñar, desarrollar y lanzar aplicaciones con una altísima escalabilidad y aportando una gran seguridad. Todo ello gracias a su vasta infraestructura instalada a lo largo del mundo entero.

Su infraestructura está formada por 112 zonas de disponibilidad en sus 37 regiones geográficas, según datos de su propia web (Google, s.f.). Los planes son ampliar

esta red a zonas de centro europa, oceanía y algunas zonas de centroamérica y asia, zonas donde se está demandando una mayor presencia.

3.2.4. IBM Cloud

Ilustración 13. Logotipo de IBM Cloud.

Fuente: https://www.ibm.com/es-es/cloud

Pasamos ahora a hablar sobre otros proveedores de servicio que destacan en la actualidad. Surge la siguiente pregunta, ¿hay algún proveedor que se soporte la comparación con el *big 3* de proveedores anteriores? La relevancia en el mercado de Amazon Web Service, Microsoft Azure y Google Cloud Platform no admite discusión hoy día con ningún otro, si bien hay algunos proveedores de servicio que están ganando cada vez más cuota de mercado y es muy interesante incluir en una lista. Y dentro de este grupo tiene una posición de honor IBM Cloud (ver ilustración 13).

Administrada y lanzada por la empresa IBM, proporciona servicios y herramientas con altos niveles de cumplimiento y seguridad, gracias a los centros de datos que tienen desplegados por todo el mundo (Platzi, 2018).

Cabe destacar aquí que oferta una de las nubes públicas más seguras del panorama, aspecto muy valorado por una gran cantidad de clientes.

3.2.5. Alibaba Cloud

Ilustración 14. Logotipo de Alibaba Cloud.

Fuente: https://eu.alibabacloud.com/en

La amplia mayoría de bibliografía que se puede consultar sobre este proveedor destaca que proporciona un amplio abanico de servicios en la nube, sustentado por una infraestructura que opera en 168 países, divididos en este caso en 18 regiones (MarkoInsights, K. M., 2019).

La amplia mayoría de sus clientes se geolocalizan en Asía, teniendo una cuota de mercado en América y Europa relativamente baja. Podríamos hablar por tanto que se trata de una opción más situacional, si bien se trata de una alternativa nada desdeñable para negocios que planean expandirse por el continente asiático.

3.3. Comparativa de prestaciones

Una vez vistos y estudiados los principales proveedores de servicio que se ofertan en el mercado en la actualidad, el siguiente paso lógico podría resumirse en las siguientes preguntas: ¿Cuál de ellos elegir para mi proyecto y/o empresa?, ¿cuál de ellos es mejor? (ver ilustración 15).

A priori son preguntas complicadas, y no existe una respuesta clara y objetiva. Como casi todo en la vida, la mejor respuesta depende de nuestras necesidades. No obstante parece procedente hacer un pequeño análisis comparando sus principales características. Nos centraremos en los tres principales proveedores: Amazon Web Service, Microsoft Azure y Google Cloud Platform.

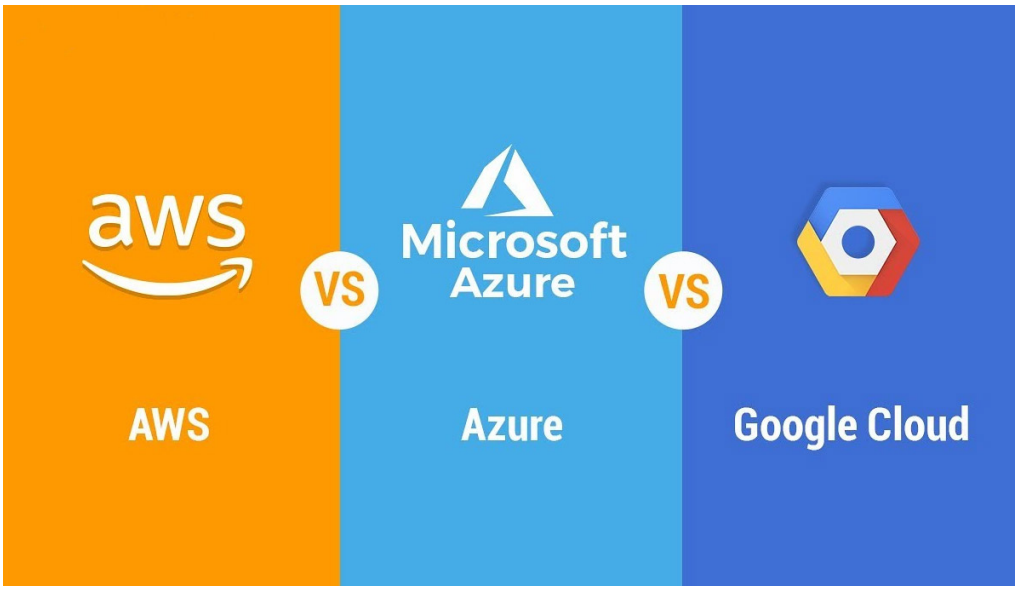

Ilustración 15. Diagrama comparativo entre AWS, Microsoft Azure y Google Cloud Platform.

Fuente: https://www.moveworkforward.com/blog/aws-vs-azure-vs-google-cloud-which-cloud-services-is-better-for-enterprises

Para iniciar esta subsección de comparación entre los tres grandes dominadores del mercado, necesitamos definir cuáles serán las reglas y los parámetros de comparación. A lo largo de este punto intentaremos responder a algunas preguntas cuyas respuestas podrían poner luz en esta comparativa de gigantes. ¿En qué año se lanza cada una de ellas?, ¿qué cuota de mercado aproximada tienen y cuál podría ser su cuota de mercado estimada?, ¿cómo queda la comparación de precios entre estos tres proveedores de servicio?, ¿qué ventajas y desventajas aportan unos respecto a otros?

- **¿En qué año se lanza cada una de ellas?**

Por orden cronológico Amazon Web Service se lanza en el año 2006, Microsoft Azure, inicialmente llamada solo Azure, se lanza en el año 2010, y Google Cloud Platform se lanza en el año 2011. La diferencia en cuanto a experiencia acumulada no es enorme, pero sin duda es aspecto que puede ser tenido en cuenta para grandes corporaciones.

- **¿Qué cuota de mercado aproximada tienen y cuál podría ser su cuota de mercado estimada?**

Hasta hace pocos años el indiscutible líder del mercado ha sido AWS, llegando a guarismos muy superiores. No obstante, el crecimiento de sus competidores ha provocado que según estudios de Synergy Research Group, AWS ha experimentado en estos últimos años un crecimiento más reducido respecto a sus años iniciales, mientras que tanto Microsoft Azure como Google Cloud PLatform han crecido a un ritmo vertiginoso con el correspondiente aumento de cuota de mercado.

Según los datos anteriores a principios del año 2021 AWS ostentaba un 33% de cuota de mercado mundial de servicios en la nube, mientras que Azure y GCP obtienen un 20% y un 10% respectivamente, como muestra la ilustración 16.

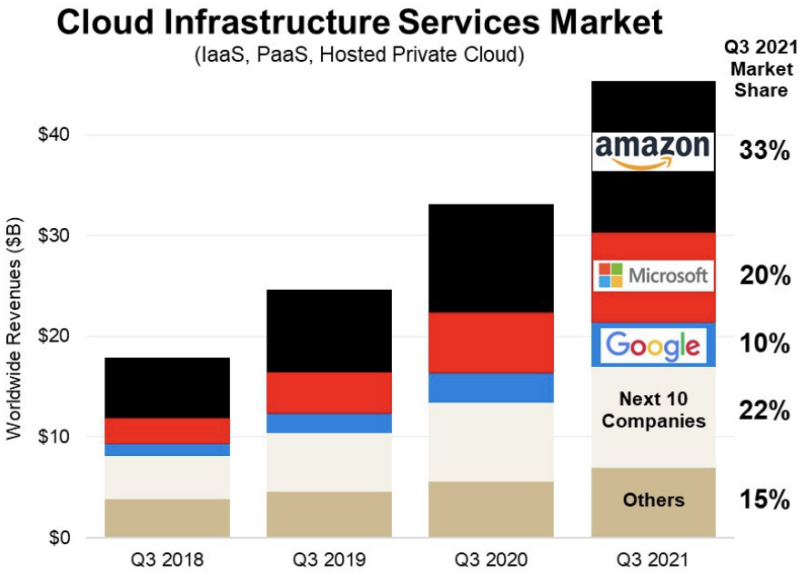

Ilustración 16. Comparativa cuota de mercado de los principales proveedores de servicio en los años 2018-2021.

Fuente: https://www.prnewswire.com/news-releases/amazon-microsoft--google-grab-the-big-numbers--but-rest-of-cloud-market-still-grows-by-27-301411702.html

- **¿Cómo queda la comparación de precios entre estos tres proveedores de servicio?**

El precio es sin duda uno de los aspectos más importante en esta comparativa. Cabe destacar aquí que los tres proveedores ofrecen una estrategia de precios flexibles y bastante competitiva. La lucha que estamos viviendo por alzarse dominador del mercado provoca que todas intenten brindar la mejor oferta a sus clientes para no quedarse atrás.

Los tres cuentan con un nivel gratuito que en muchos casos y para proyectos no demasiado grandes es más que suficiente. A partir de aquí podemos afirmar que para proyectos de gran envergadura no hay grandes diferencias y este ítem no

debería ser un objeto de elección destacado, si bien Google Cloud Platform lleva bastante tiempo proponiendo ofertas muy agresivas económicamente con la firme intención de ganar clientes a sus competidores.

- **¿Qué ventajas y desventajas aportan unos respecto a otros?**

Comenzando por Amazon Web Service quizás lo primero que llama la atención de este proveedor con respecto a sus competidores es su experiencia. Gartner (Costello, K., 2021) describe AWS como:

> *"El proveedor más maduro para la empresa, con las capacidades más profundas para gobernar una gran cantidad de usuarios y recursos."*

Respecto a Microsoft Azure su característica más destacada respecto a sus competidores radica en propio origen. Un buen porcentaje tanto de grandes empresas como otras más pequeñas y clientes individuales usan tanto Windows como otras herramientas de Microsoft. No deja de tener cierto sentido para este buen porcentaje de casos llegar a la conclusión de que la mejor alternativa por un simple cuestión de facilidad de integración es Microsoft Azure.

En cuanto a Google Cloud Platform destaca sobremanera en su oferta en contenedores con su servicio Kubernetes, en el que quizás tiene más experiencia que sus competidores y le da un plus en este aspecto, además de aportar un valor significativo en cuanto a seguridad. Por el lado negativo su abanico de herramientas y servicios es más reducido que los de AWS y Microsoft Azure lo que les quita un poco de cuota de mercado.

Por último, y enfocándonos en el objeto principal de estudio de este trabajo, si bien todos ofrecen una amplia gama de servicios de Platform as a service, la elección predilecta en la actualidad podría ser Microsoft Azure, que cuenta con una oferta un poco más completa que sus competidores. AWS cuenta con grandes servicios PaaS como Elastic Beanstalk, del que hablaremos más adelante, y por parte de Google

Cloud Platform podríamos destacar Cloud Run. No obstante consideramos que en este apartado Microsoft Azure se lleva la palma gracias a su variedad de servicios entre los que destaca sobremanera Web App Service, plataforma en la nube para alojar sitios web.

3.4. ¿Qué proveedor voy a utilizar en mi aplicación?

Elaborar esta tercera sección completamente enfocada en el estudio de los diferentes proveedores de servicio existentes en el mercado nos otorga un importante bagaje de conocimientos para responder a esta pregunta y seleccionar el proveedor de servicio que posteriormente utilizaremos en la siguiente sección para elaborar nuestra aplicación.

Los grandes candidatos, como no podía ser de otra manera, han sido Amazon Web Service, Microsoft Azure y Google Cloud Platform, si bien este último quedó descartado en primeras instancias por ser el menos conocido para mí.

De entre los dos candidatos restantes mi elección final ha sido Amazon Web Service. Su plataforma y oferta me han parecido más interesante y tras unas primera interacciones con ambos proveedores, la capa gratuita de AWS y mi mayor conocimiento de este proveedor han terminado por decantar la balanza.

4. APLICACIÓN

Llegamos al fin al punto en que describiremos, analizaremos y, por último, desarrollaremos una aplicación a modo de ejemplo, con la firme intención de probar y poder llegar a una serie de conclusiones sobre todo lo visto y estudiado a lo largo del presente trabajo.

Como se ha ido comentando en los puntos anteriores, en la actualidad, la informática y las tecnologías de la nube han revolucionado el modo en que las aplicaciones se desarrollan, implementan, e incluso se utilizan. Una de las principales ventajas de la computación en la nube, y una de las que me parece más interesante estudiar aquí, es la capacidad de desplegar aplicaciones de manera rápida y eficiente, sin la necesidad de mantener costosas y, en muchos casos, complicadas de gestionar, infraestructuras físicas. En este contexto, durante el presente apartado de este trabajo de fin de grado nos enfocaremos en el despliegue de una aplicación en la nube utilizando el modelo de Plataforma como Servicio (PaaS).

La aplicación que diseñaremos, implementaremos y finalmente desplegaremos en la nube utilizando la metodología Platform as a Service, estará centrada en el mundo del deporte, más concretamente en la NBA, liga de baloncesto profesional americana, de la cual soy un ferviente fan prácticamente desde que tengo recuerdos. La aplicación permitirá al usuario realizar búsquedas y acceder a estadísticas relativas a dicha liga.

El acceso a estadísticas, tanto actuales como históricas, puede ser de gran utilidad para aficionados, periodistas deportivos, analistas y cualquier persona interesada en seguir de cerca el mundo del baloncesto profesional. Como hemos ido desgranando en puntos anteriores, el desarrollo y despliegue de la aplicación en la nube nos permitirá exprimir todas las ventajas que ofrece la nube, tales como escalabilidad, disponibilidad y flexibilidad, todo ello sin tener que preocuparnos por la gestión de la

infraestructura subyacente. Los aspectos positivos son sin duda unívocos, y creo firmemente que quedarán plasmados con suficiencia durante el desarrollo de mi aplicación, pero, ¿será todo color de rosa?, ¿o nos encontraremos con dificultades asociadas a la nube durante el proceso?

Comenzaremos hablando de las tecnologías usadas para desarrollar la aplicación y continuaremos realizando la planificación temporal y la planificación de costes del proyecto y finalizamos con un breve análisis, a modo de introducción, de los siguientes subapartados en que se desgranará este epígrafe centrado en el diseño, implementación y despliegue de nuestra aplicación.

4.1. Tecnologías empleadas

En este primer apartado detallaremos tanto las herramientas que hemos utilizado durante el diseño, desarrollo e implementación de nuestra aplicación, como los lenguajes usados en la programación de la misma. Haremos especial hincapié en los motivos que nos han llevado a realizar estas elecciones.

4.1.1. Herramientas

- **Visual Studio Code**

Es un entorno de desarrollo integrado (IDE) de código abierto y multiplataforma desarrollado por Microsoft (ver logotipo en ilustración 17). Los motivos que nos han llevado a seleccionar esta herramienta de desarrollo por encima de otras opciones disponibles han sido muy variados.

Ilustración 17. Logotipo de Visual Studio Code.

Fuente: https://code.visualstudio.com/

Podríamos comenzar hablando sobre su versatilidad, el gran rendimiento que ofrece o la amplia gama de extensiones y plugins que pone a disposición del desarrollador. Sin duda todas estas son razones más que suficientes para elegir este entorno para desarrollar una aplicación. Pero la lista de aspectos positivos que me hacen decantarme por esta opción no termina aquí.

Visual Studio Code proporciona una interfaz de usuario bastante intuitiva y totalmente personalizable, aspecto que añadido a la amplia gama de lenguajes con los que presenta compatibilidad, lo convierte en una excelente opción.

- **Visual Paradigm**

Visual Paradigm es una herramienta de modelado y diseño de software utilizado para diseñar, modelar, visualizar y finalmente documentar todo el ciclo de vida del desarrollo de una aplicación. Los diagramas y modelos visuales que proporciona aportan una gran utilidad al proyecto, y mucho valor a las personas implicadas en él.

¿Por qué nos hemos decantado por esta opción por encima de otras herramientas? Visual Paradigm admite un abanico muy amplio de técnicas de modelado y

diagramas, y hemos considerado las ventajas que aporta unido al mayor conocimiento previo que tenía de esta herramienta por encima de otras similares, la colocan en un escalón difícilmente alcanzable. La Universidad de Jaén, por si todo esto no fuera suficiente, proporciona una versión de pago de la misma (ver logotipo en ilustración 18).

Ilustración 18. Logotipo de Visual Paradigm.

Fuente: https://forums.visual-paradigm.com/

- **GitHub**

GitHub es una plataforma que proporciona a los desarrolladores la capacidad de colaborar en proyectos de software, gestionar el control de versiones y alojar repositorios de código Su conocido logotipo lo podemos observar en la ilustración 19. Su uso está ampliamente estandarizado en la comunidad.

Ilustración 19. Logotipo de GitHub.

Fuente: https://github.com/logos

Es difícil entender hoy día el desarrollo de software sin una herramienta de control de versiones basada en Git, y de entre las opciones disponibles GitHub es una opción muy sólida que además cuenta con una interfaz muy intuitiva y permite realizar multitud de operaciones relativas a nuestros repositorios de forma sencilla.

Centrándonos en el caso que nos ocupa, la aplicación que queremos desarrollar, GitHub nos será de mucha utilidad para desplegar la aplicación, y tendrá un papel importante el modelo que despliegue que utilizaremos.

4.1.2. Lenguajes de programación y principales bibliotecas

- **HTML5**

HTML5 es la última versión del lenguaje de marcado HTML (HyperText Markup Language) utilizado para la creación y estructuración de páginas web.

- **CSS**

CSS es un lenguaje de hojas de estilo ampliamente utilizado por los desarrolladores para definir y posteriormente crear la presentación de un documento estructurado escrito en un lenguaje de marcado. Permite definir una amplia gama de

características como colores, fuentes, márgenes, etc. Se trata de una herramienta fundamental para el diseño web, y aporta un serie de ventajas, entre las que se listan aspectos como flexibilidad y control, y resultan fundamentales para desarrollar una aplicación.

- **Python**

Python destaca por ser un lenguaje de programación de alto nivel, interpretado y multipropósito. Entre sus principales características encontramos una sintaxis clara y legible, aspecto que consideramos fundamental para escribir y posteriormente comprender el código. Se trata de una opción muy popular entre desarrolladores de diferentes niveles de experiencia, y el lenguaje con el que yo como desarrollador de la aplicación de este trabajo de fin de grado me siento más cómodo y poseo más experiencia.

No obstante, más allá de estas preferencias personales a las que hacemos referencia, Python cuenta con una amplia variedad de módulos y bibliotecas que facilitan la vida al desarrollador en múltiples aspectos, cuestión que consideramos importante en este proyecto.

- **Flask**

Flask es un framework de desarrollo basado en Python que permite crear aplicaciones web de manera rápida y sencilla. Aporta una serie de ventajas y características que no conviene obviar, y que lo convierten en un gran competidor como elección. La herramienta Flask destaca por su flexibilidad y capacidad para adaptarse a diferentes proyectos, permitiendo que el desarrollador construya aplicaciones adaptándose a las necesidades específicas del proyecto.

4.2. Planificación temporal

En este punto definiremos una planificación temporal inicial de nuestro trabajo de fin de grado. Para conseguir los objetivos propuestos inicialmente (1.2. Objetivo del

proyecto) se hace necesario realizar una planificación de tiempo que determine la duración de cada una de las tareas en que se descompone el desarrollo del trabajo (ver tabla 1). Es importante señalar que el carácter de esta planificación es puramente orientativo, tratándose de una estimación que se realiza antes de comenzar a desarrollar el proyecto. El calendario final de horas podría llegar a variar bastante respecto a esta estimación inicial.

Nº tarea	**Tarea**	**Duración estimada**
1	Investigar sobre Cloud Computing y sus características	2 semanas
2	Investigar sobre el paradigma Platform as a Service	2 semanas
3	Familiarizarse con los principales proveedores de servicio del mercado	1 semana
4	Diseñar y planificar la aplicación	4 semanas
5	Desarrollar la aplicación	6 semanas
6	Desplegar la aplicación	1 semana
7	Validar la aplicación	2 semanas
8	Fase de retroalimentación de la aplicación	3 semanas
9	Elaboración de la memoria final	2 semanas

Tabla 1. Planificación temporal del trabajo de fin de grado.

No obstante y con el objetivo de clarificar esta planificación inicial que acabamos de definir utilizaremos un diagrama de Gantt (ver ilustración 20). Un diagrama de Gantt es una herramienta gráfica (Hinojosa, M. A., 2003) utilizada ampliamente en una gran variedad de proyectos de muy diferente índole para representar la planificación temporal.

Ilustración 20. Diagrama de Gantt.

Es importante señalar en este punto y en vista del diagrama de Gantt que observamos en la ilustración 20, que para la implementación de la aplicación seguiremos un modelo de desarrollo de software incremental, como veremos más adelante en el apartado 4.5. Implementación de la aplicación. Debido a esto las tareas relativas al software que vamos a crear (diseñar y planificar la aplicación, desarrollar la aplicación, desplegar la aplicación, validar la aplicación y fase de retroalimentación de la aplicación) no seguirán un orden estricto, e irán intercalando cíclicamente por cada sprint o incremento de la aplicación que se decida implementar.

4.3. Estimación de costes de la aplicación

En este punto realizaremos un estudio de los costes a nivel económico de nuestro proyecto. Tomaremos como base el punto anterior (4.2. Planificación temporal) donde suponíamos un tiempo de 16 semanas y yo soy el único desarrollador.

En 2023 el salario de un desarrollador web se sitúa de media en 31.600 € brutos por año (Jobted, 2023). Suponiendo un trabajo semanal de 40 horas, la hora sale a

15,19 €/hora. Dado que para el desarrollo de la aplicación se han planificado 16 semanas y no se podrá dedicar más de 5 horas semanales, el coste debido a personal es de 1215,20 €.

Además del anterior, otro aspecto que tendremos en cuenta para la estimación de costes será suponer una vida útil al equipo con el que he desarrollado el trabajo y la aplicación de 5 años.

En la tabla 2 observamos la estimación de costes final definida.

- Para calcular el coste del desarrollador web se calcula el número de horas que trabajará en el proyecto y se calcula por el valor de su hora, que como se ha indicado son 1.215,20 €.
- Para calcular el gasto en el equipo del desarrollador, un MacBook Pro (16 pulgadas, 2019), valorado en 1.855 € (Amazon, 2023) y suponiendo una vida útil de 5 años, debemos calcular el valor por mes del equipo y posteriormente calcular el coste total, teniendo en cuenta el número de semanas estimado que se usará para la realización de este proyecto: 1855 € / 260 semanas = 7,13 €/mes. El proyecto durará 23 semanas: 7,13 x 23.
- Respecto al software utilizando Github y Visual Studio Code son gratuitos, y Visual Paradigm tiene una versión para estudiantes descargable mediante la web de la Universidad de Jaén de la que he hecho uso.

Concepto	Coste
Desarrollador web	1.215,20 €
Equipo del desarrollador: MacBook Pro (16 pulgadas, 2019)	164 €
Visual Studio Code (descarga gratuita)	0 €
Visual Paradigm (versión gratuita para estudiantes de la Universidad de Jaén)	0 €
GitHub	0 €
Coste final	**1.279,20 €**

Tabla 2. Costes asociados al proyecto.

4.4. Análisis de la aplicación

4.4.1. Historias de usuario

Una historia de usuario, en el contexto de desarrollo de software, puede definirse como una representación de un requisito funcional propio del software que se quiere desarrollar. Su principal cometido es capturar y posteriormente describir de forma clara y concisa estos requisitos identificados para el sistema desde la perspectiva del usuario final.

Para describir las historias de usuario intentaremos seguir el siguiente formato:

Como [actor] quiero [caso de uso] para [descripción/beneficio]

Una vez descritas es importante establecer un orden de prioridad que posteriormente nos permita seleccionar aquellas historias de usuario que iremos desarrollando durante la fase de implementación de la aplicación. Para ello usaremos el método MoSCoW (Chiyana, S., 2020).

MoSCoW es una técnica de priorización de tareas y funcionalidades en proyectos con limitaciones de tiempo, que es precisamente en el contexto en que nos encontramos en un trabajo de fin de grado.

La palabra MoSCoW, ideada en la década de los 90 por Dai Clegg, es un acrónimo de las cuatro categorías que describe:

- *Must have:* incluirá aquellas historias de usuario que son imprescindibles para el desarrollo del proyecto, y se considera que no deben faltar en la versión final del producto.
- *Should have:* en esta categoría se encuadrarán las historias de usuario que, si bien no son imprescindibles para el producto, aportan un valor muy alto, y creemos que deberían estar en la versión final del producto.
- *Could have*: historias de usuario sin las que el proyecto tendrá igualmente una funcionalidad completa, pero consideramos que su inclusión aporta un valor añadido al producto.
- *Want to have but Won't have this time*: tareas que no son consideradas una prioridad en este momento, y aunque podrían ser reconsideradas en el futuro, creemos que no serán incluidas en la versión final.

Por último y antes de pasar a detallar y describir las historias de usuario parece importante especificar que estas historias de usuario deben seguir los principios INVEST (Scrum manager bok, 2021). Este método es usado para asegurar la calidad y legibilidad de las historias de usuario y para cumplirlo, debe cumplir las características en las que se desgrana su nombre: *independent, negotiable, valuable, estimable, small* y *testable* (independiente, negociable, valiosa, estimable, pequeña y comprobable).

Nº historia de usuario	**Actor**	**Casos de uso**	**Descripción**	**MoSCoW**
	como	*quiero*	*para*	
1	usuario logueado / usuario no logueado	acceder a una pantalla inicial	obtener una explicación de la funcionalidad de la aplicación y tener acceso a las distintos usos que ofrece	Must have
2	usuario logueado / usuario no logueado	saber la media de puntos anotados por un jugador en una temporada	visualizar los resultados que coinciden con dicha búsqueda	Must have
3	usuario logueado / usuario no logueado	saber la media de asistencias das por un jugador en una temporada	visualizar los resultados que coinciden con dicha búsqueda	Must have
4	usuario logueado / usuario no logueado	saber la media de rebotes cogidos por un jugador en una temporada	visualizar los resultados que coinciden con dicha búsqueda	Must have
5	usuario logueado / usuario no logueado	saber las victorias de un equipo en una temporada	visualizar los resultados que coinciden con dicha búsqueda	Should have
6	usuarios logueado / usuario no logueado	saber el jugador con mejor porcentaje de tiros libres de la temporada	visualizar los resultados que coinciden con dicha búsqueda	Should have
7	usuario no logueado	registrarme en la aplicación	hacer uso de las funcionalidades de usuario logueado	Could have
8	usuario logueado	tener acceso a mis opciones más frecuentes	acceder más rápidamente a aquellas opciones de las que hago uso habitualmente	Could have
9	usuario logueado	poder descargar la respuesta de la api	obtener la respuesta en un archivo en local	Could have
10	usuario logueado	poder introducir nuevos datos	incrementar los datos existentes en el sistema	Want to have but Won't have this time

Tabla 3. Historias de usuario definidas inicialmente para desarrollar la aplicación BasketApp.

4.5. Inicialización de la aplicación en la nube de AWS

El proveedor de servicios elegido para desplegar nuestra aplicación en la nube ha sido Amazon Web Services, más conocida por su acrónimo AWS (tal y como desgranamos y detallamos en el punto 3.4 del presente proyecto, ¿qué proveedor voy a utilizar en mi aplicación?).

La aplicación usará el lenguaje de programación Python, siendo Flask el framework elegido para lanzar el microservicio que proveerá de datos la aplicación. Django era mi otra alternativa para realizar dicha tarea pero finalmente, y debido a que Flask está especialmente diseñado para aplicaciones algo más pequeñas con funcionalidades más simples, y a mi mayor conocimiento de este framework, Flask ha sido mi elección.

En la tarea que ahora nos ocupa, despliegue de mi aplicación en la nube, la inevitable pregunta que nos aborda y que, aunque ya hemos respondido durante la parte más teórica del proyecto no está de más volver a incidir es: ¿Qué aporta PaaS frente a un modelo más tradicional?

Basta un simple paseo por AWS, por ejemplificar con el proveedor que usaremos, para darse cuenta del vasto conjunto de funcionalidades y facilidades que provee con un, al menos en la comparación con el modelo tradicional, bajo nivel de tiempo y esfuerzo.

Para nuestro caso, nosotros implementaremos integración continua en la nube de AWS, de modo que todo quede lo más automatizado posible, y podamos observar de primera mano el enorme potencial de utilizar el paradigma Platform as a Service, objeto principal de estudio que aquí nos ocupa. El diagrama de aplicaciones que usaremos para este despliegue podría resumirse en la ilustración 21.

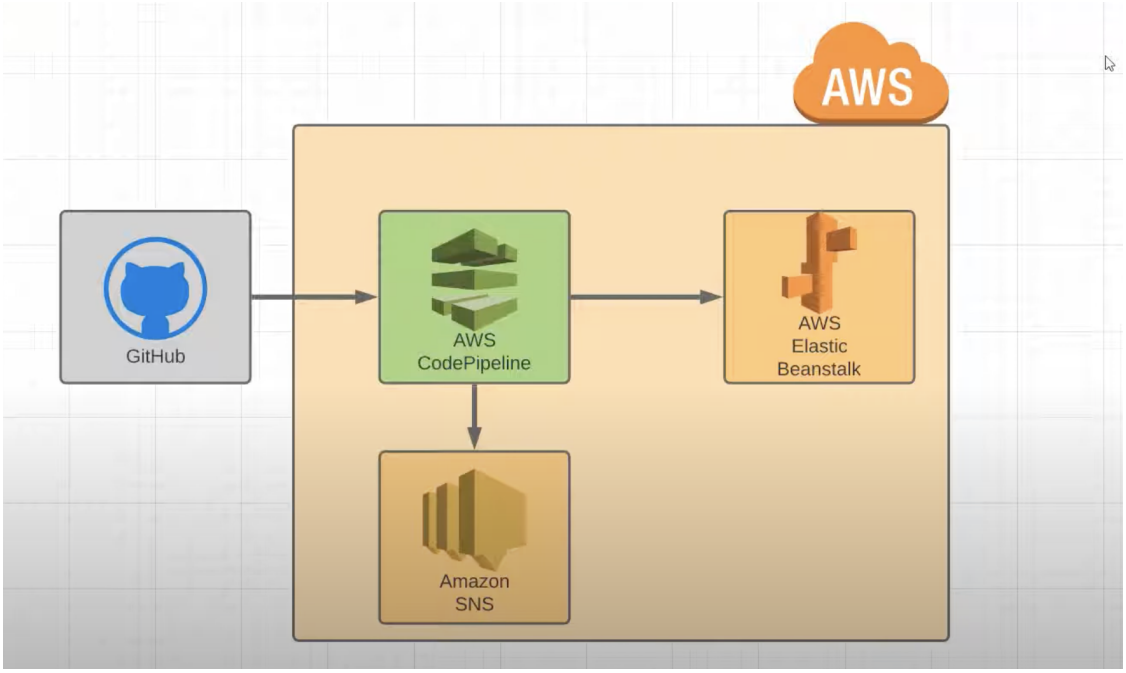

Ilustración 21. Diagrama de aplicaciones que se usarán para el despliegue en AWS.

Como observamos, y a modo de introducción para clarificar el proceso, el flujo de trabajo será el siguiente. Tenemos una aplicación que subiremos a un repositorio de GitHub. Este repositorio lo conectaremos con AWS CodePipeline[1], donde de modo muy sencillo e intuitivo tendremos dos funcionalidades muy encapsuladas. Un primer punto de generación y actualización de código automatizada, que veremos más detalladamente en un momento, y un segundo acto de despliegue de dicho código.

En esta funcionalidad de despliegue de código tendremos una conexión con AWS Elastic Beanstalk, que nos permitirá desplegar aplicaciones sin tener que preocuparnos por infraestructuras o escalabilidad de las mismas. Se trata de una aplicación muy interesante que me ha parecido muy útil incluir en mi desarrollo. Finalmente, para recibir notificaciones del estado y actualización del pipeline, usaremos Amazon SNS.

[1] https://docs.aws.amazon.com/codepipeline/latest/APIReference/Welcome.html

Una vez visto todo lo anterior pasamos a desplegar una primera versión muy simple de aplicación web. El primer paso, por tanto, una vez dentro de nuestra cuenta de AWS, será ir a AWS CodePipeline. Nos aparecerá la pantalla que observamos en la ilustración 22.

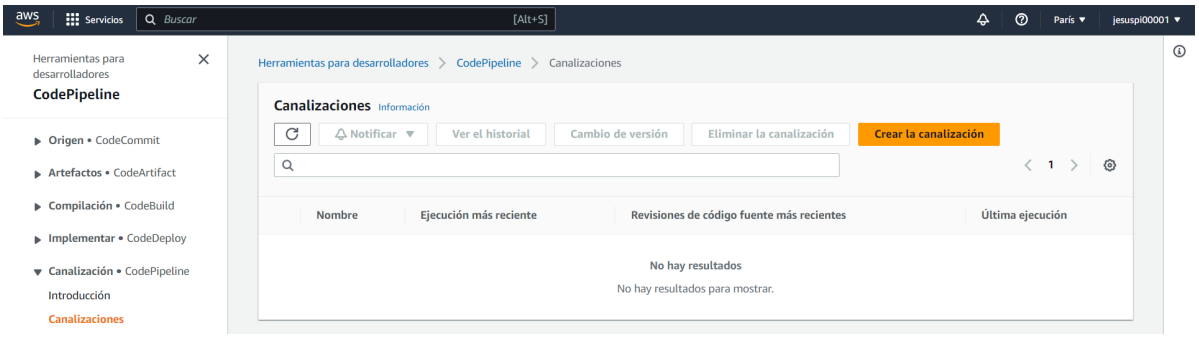

Ilustración 22. Pantalla inicial CodePipeline.

Una vez aquí clicamos sobre Crear la canalización y se nos abrirá la pantalla de la ilustración 23.

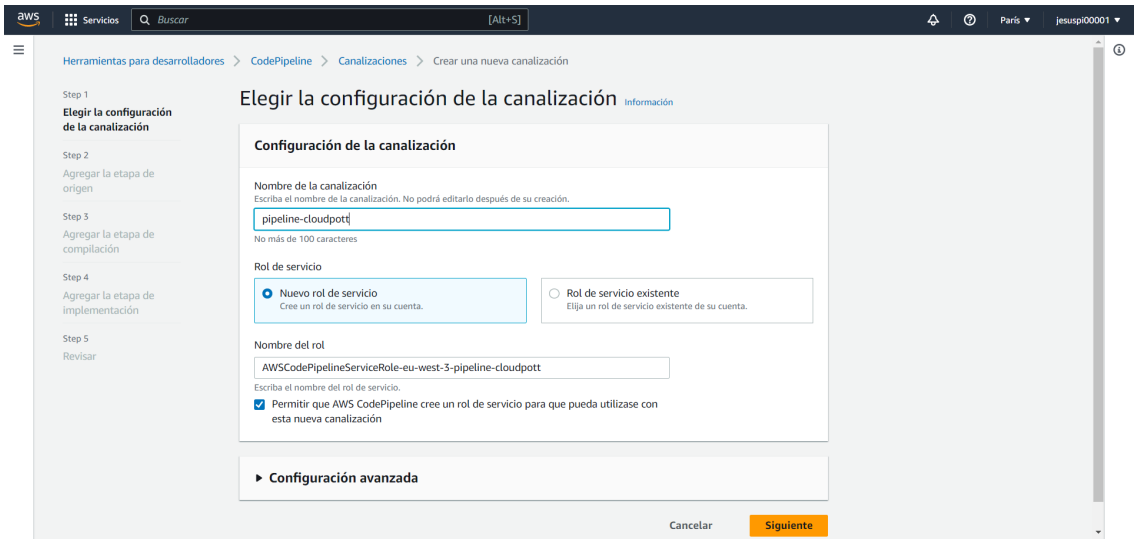

Ilustración 23. Pantalla configuración de la canalización del pipeline.

En esta nueva pantalla pondremos un nombre a nuestro pipeline y clicaremos sobre Siguiente. Se nos abrirá la pantalla de la ilustración 24.

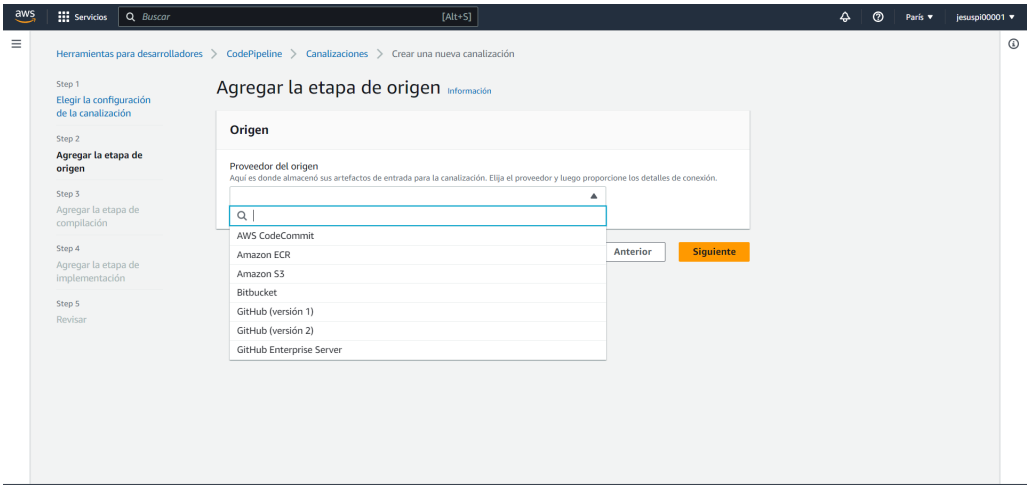

Ilustración 24. Pantalla selección proveedor de código para nuestra canalización.

Aquí debemos seleccionar nuestro proveedor de código. Seleccionaremos GitHub Versión 2. A continuación, clicamos sobre Connect to GitHub y se nos abrirá la pantalla que vemos en la ilustración 25.

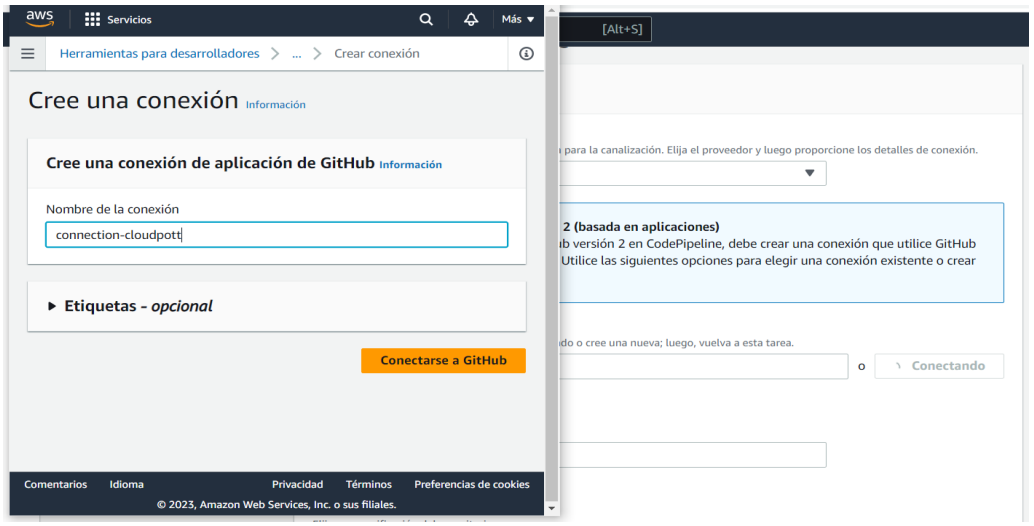

Ilustración 25. Pantalla elección nombre de la conexión entre GitHub y la canalización de CodePipeline.

Ponemos un nombre a la conexión y clicamos sobre Conectarse a GitHub. Se nos abrirá la siguiente pantalla donde elegiremos nuestro repositorio creado anteriormente y clicaremos sobre Install.

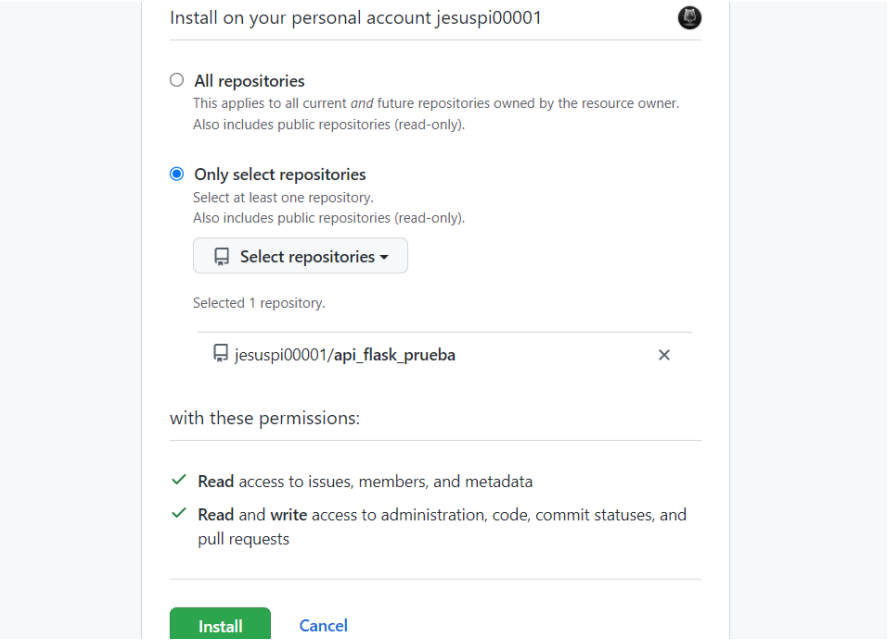

Ilustración 26. Pantalla selección repositorio de GitHub que queremos conectar.

A continuación nos aparecerá la pantalla de la ilustración 27.

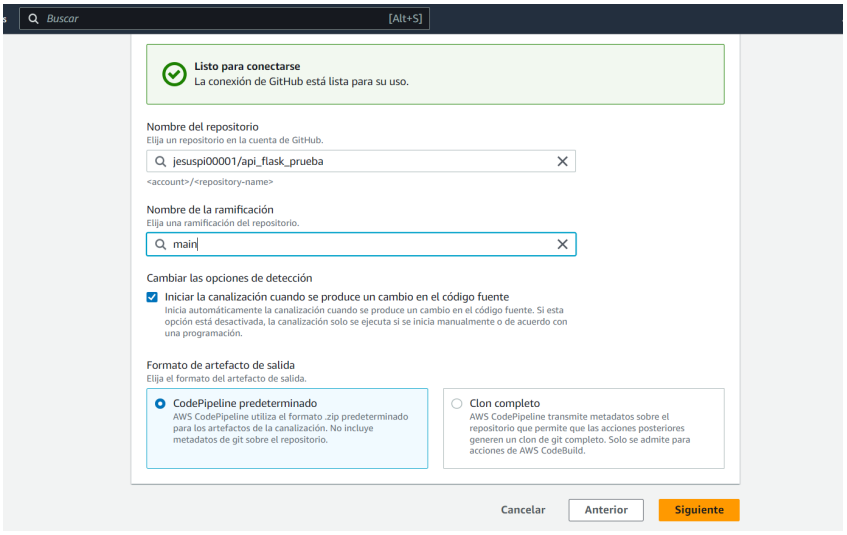

Ilustración 27. Pantalla elección de la conexión que queremos conectar y que rama será la de despliegue.

Vemos que la conexión ha sido creada con éxito. Pero como observamos en la ilustración 27, una conexión puede tener varios repositorios, así que ahora debemos

seleccionar qué repositorio queremos conectar, y qué rama queremos que se despliegue en nuestro entorno de producción. Aquí será muy importante seleccionar la opción Iniciar la canalización cuando se produce un cambio en el código fuente, con el fin de conseguir la integración continua que comentábamos anteriormente.

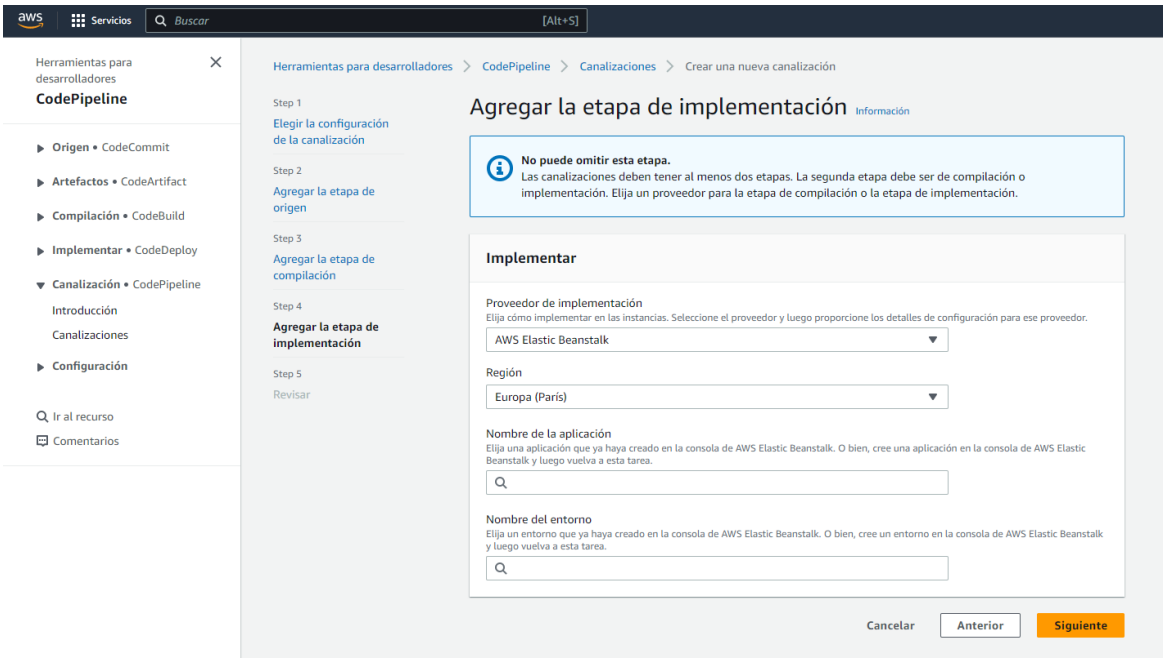

Ilustración 28. Pantalla donde elegiremos Beanstalk como opción de despliegue.

El siguiente paso, como se observa en la pantalla de ilustración 28, será la fase de despliegue. Tenemos varias opciones, nosotros usaremos Beanstalk. Debemos seleccionar una región, que en nuestro caso sería París por cercanía, y que seleccionemos una aplicación y un entorno creados en Beanstalk. Así que ese será nuestro siguiente paso.

Buscamos AWS Elastic Beanstalk y nos dirigimos a la siguiente pantalla, donde clicaremos sobre Create application para comenzar a crear nuestra aplicación y entorno.

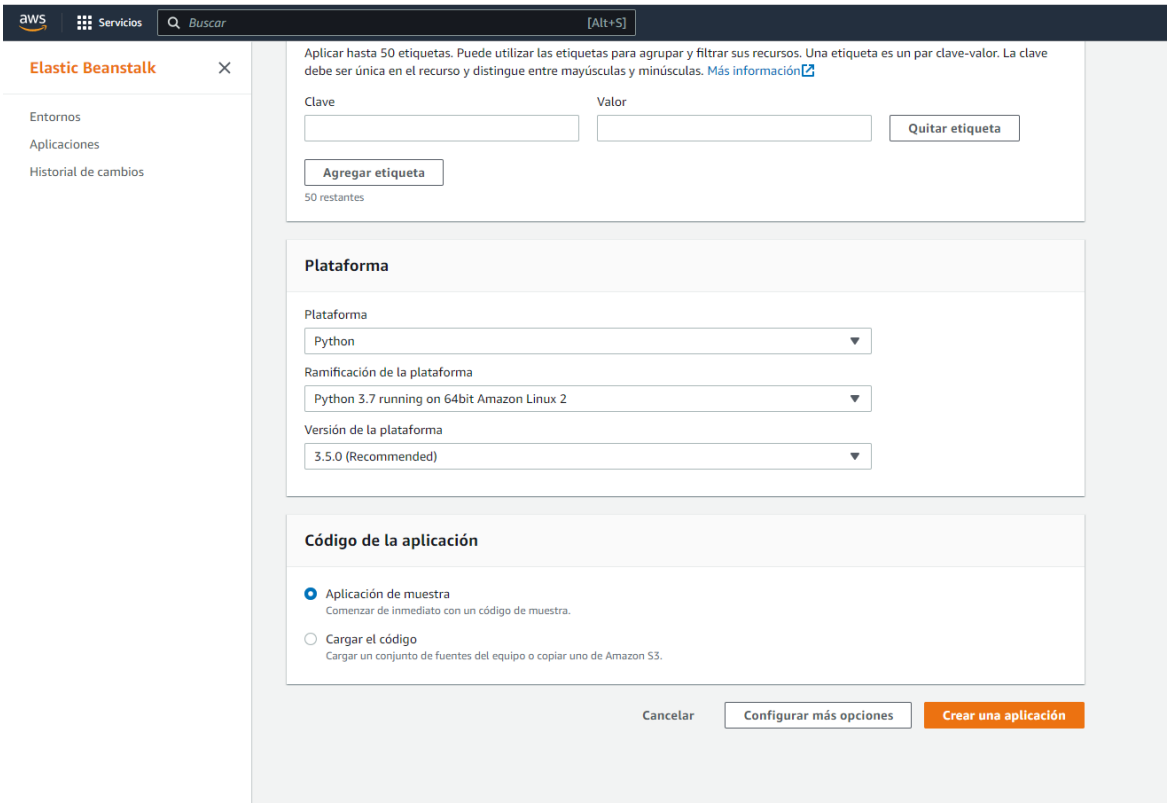

Ilustración 29. Pantalla selección nombre del entorno de Beanstalk y lenguaje de programación.

Aquí pondremos un nombre a nuestra aplicación y elegiremos Python como plataforma (ver ilustración 29). Por defecto observamos que nuestra aplicación correrá por debajo en una instancia de Linux, aunque eso será transparente para nosotros. Finalmente para crear nuestra aplicación en esta primera versión para comprobar que nuestro entorno quedó correctamente configurado, clicamos sobre Crear una aplicación. Esto podría demorarse unos minutos.

Una vez finalizado este proceso, nos aparecerá la siguiente pantalla (ilustración 29). Aquí podemos observar el estado de nuestra aplicación, así como los últimos eventos ocurridos.

Ahora podemos volver a CodePipeline y, ahora sí, podemos seleccionar la aplicación y el entorno creados en el paso anterior, como observamos en la siguiente imagen. A continuación clicamos sobre Siguiente y sobre Crear pipeline.

Ahora tenemos nuestro pipeline creado, con los dos pasos comentados anteriormente, toma de código y despliegue de código, como observamos en la imagen que observamos en la ilustración 30.

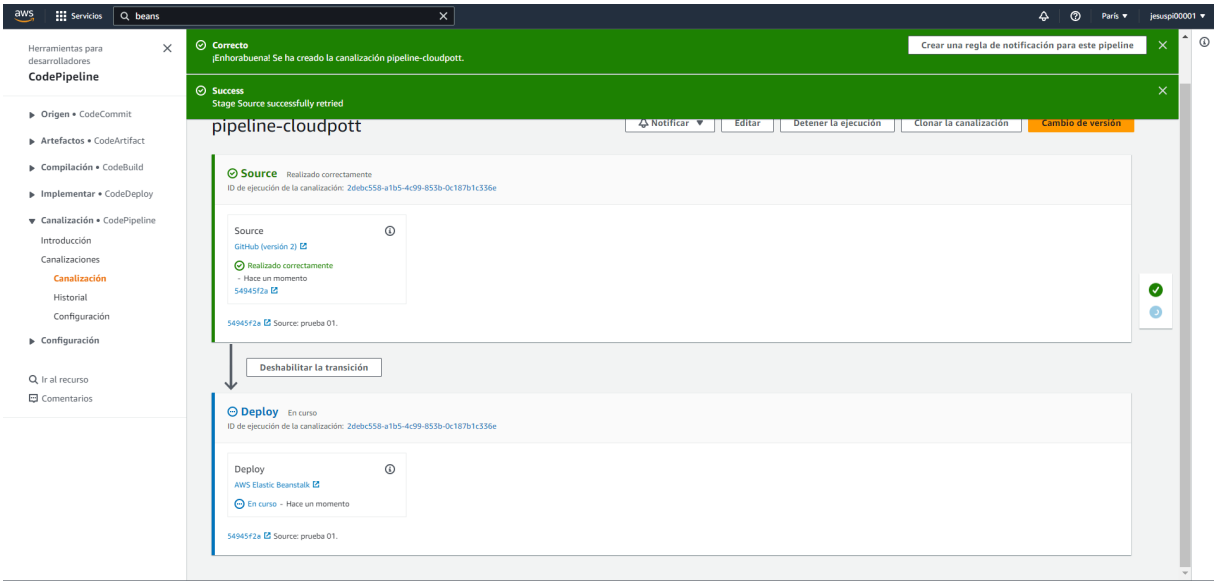

Ilustración 30. Pantalla donde observamos el estado de las herramientas que hemos usado.

La fuerte ventaja de esto será que es un proceso automático cada vez que despleguemos en la rama main seleccionada. De modo que podremos ir publicando nuevas versiones conforme nuestra aplicación vaya creciendo y adquiriendo nuevas funcionalidades.

El último paso de nuestro flujo de trabajo, como se comentó con anterioridad, será configurar Amazon SNS para que Amazon nos notifique cada nueva ejecución del pipeline, y si está ejecución ha sido exitosa o si ha habido algún problema.

Para ello, en esta misma pantalla, clicamos sobre Notificar, en el desplegable, sobre Crear la regla de notificación. Nos aparecerá la pantalla de ilustración 31.

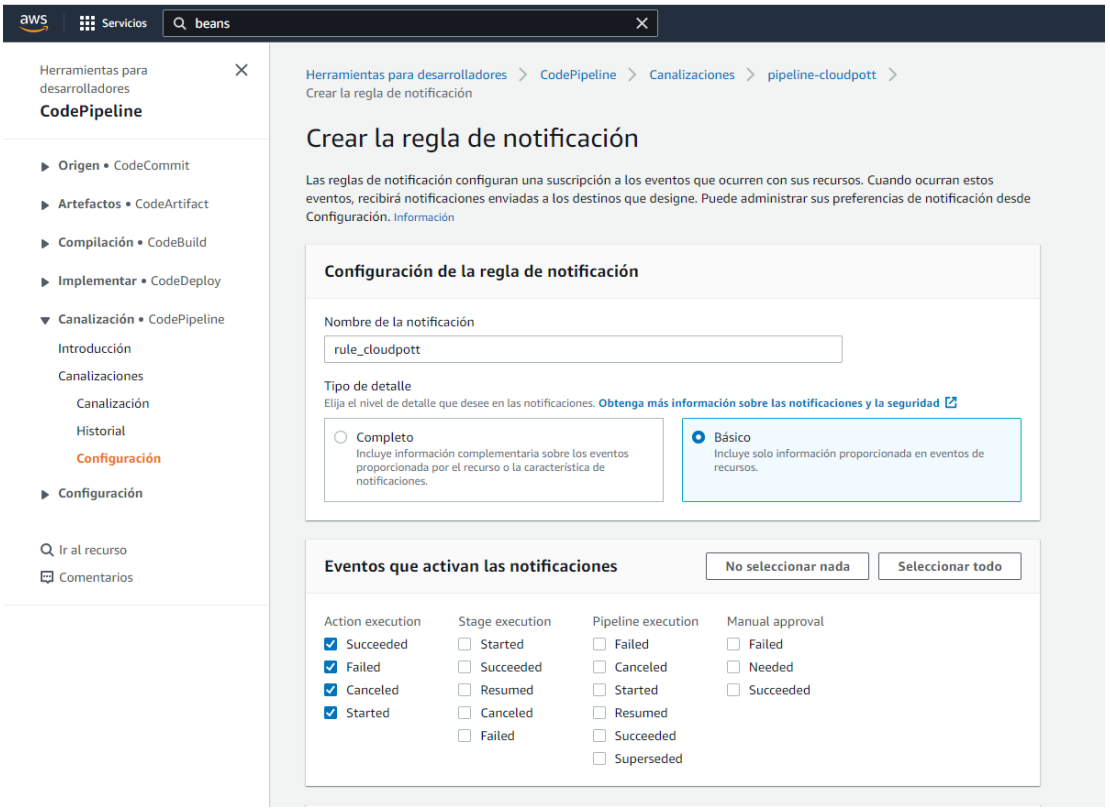

Ilustración 31. Pantalla configuración SNS.

Ponemos un nombre a nuestra regla de notificación, elegimos Básico y seleccionamos los eventos que queremos que activen nuestras notificaciones.

A continuación debemos configurar esta notificación, para seleccionar el modo en que queremos ser notificados. Se mostrará la pantalla de la ilustración 32.

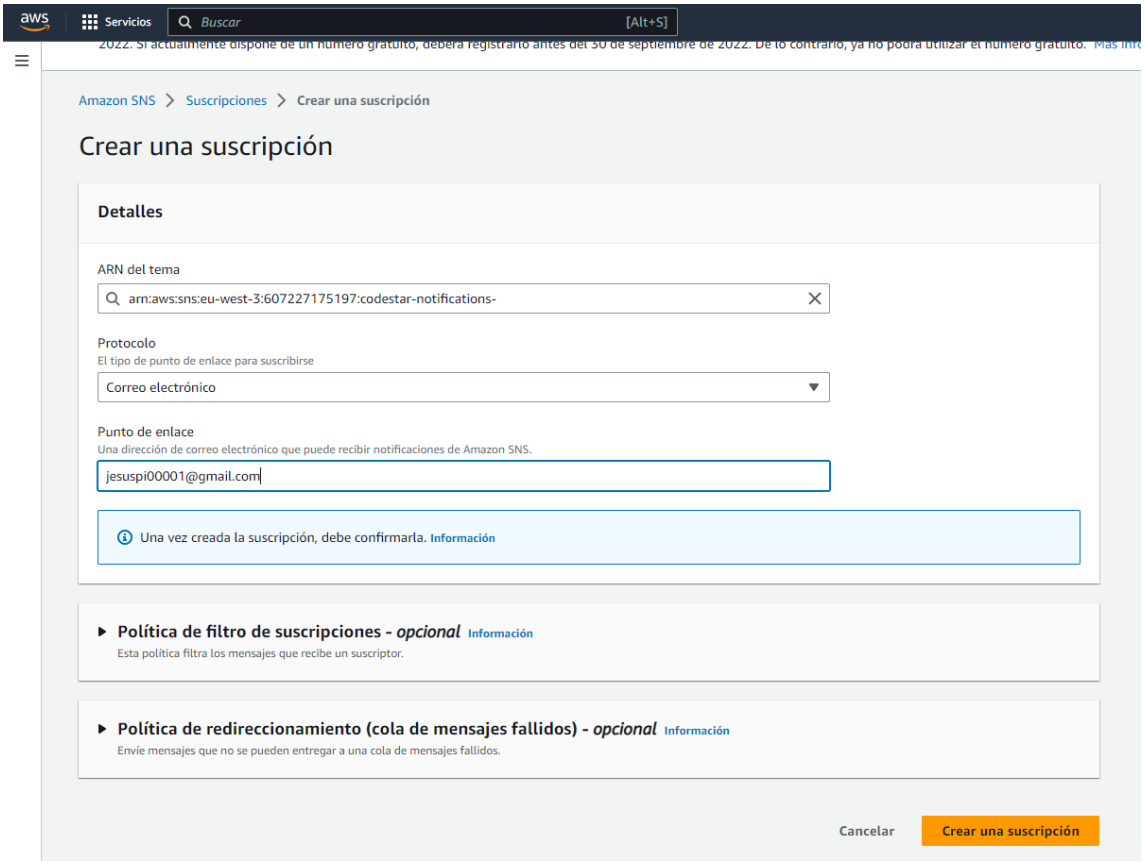

Ilustración 32. Pantalla creación notificación a correo electrónico de nuestra elección.

La configuramos a nuestro gusto, y clicamos sobre Crear una suscripción.

Finalmente tenemos desplegada y configurada en AWS nuestra aplicación. La potencia de este desarrollo, también radica en la facilidad con la que puedes desplegar las distintas versiones desplegadas de tu aplicación desde la pestaña versiones desde el propio pipeline.

El enlace a la aplicación será el siguiente:

http://apitfgjesus-env.eba-7epaqmjf.eu-west-3.elasticbeanstalk.com/

El enlace al repositorio de GitHub será el siguiente:

https://github.com/jesuspi00001/api-tfg-jesus

4.6. Implementación de la aplicación.

En este nuevo apartado abordaremos el enfoque y la metodología seleccionada para desarrollar nuestra aplicación. En busca de un enfoque eficiente y flexible, hemos optado por utilizar una metodología de desarrollo de software incremental. Las preguntas que surgen son: ¿en que se basa este modelo de desarrollo?, ¿por qué es la mejor solución para el desarrollo de nuestra aplicación?

La metodología de desarrollo de software incremental (ver ilustración 33) basa su idea en la división del proyecto en incrementos o iteraciones, donde cada una de estas iteraciones agrega una funcionalidad adicional a la aplicación. Este modelo permite un desarrollo más ágil y adaptable que las metodologías tradicionales, y nos permitirá una primera versión funcional de la aplicación en etapas muy tempranas del desarrollo.

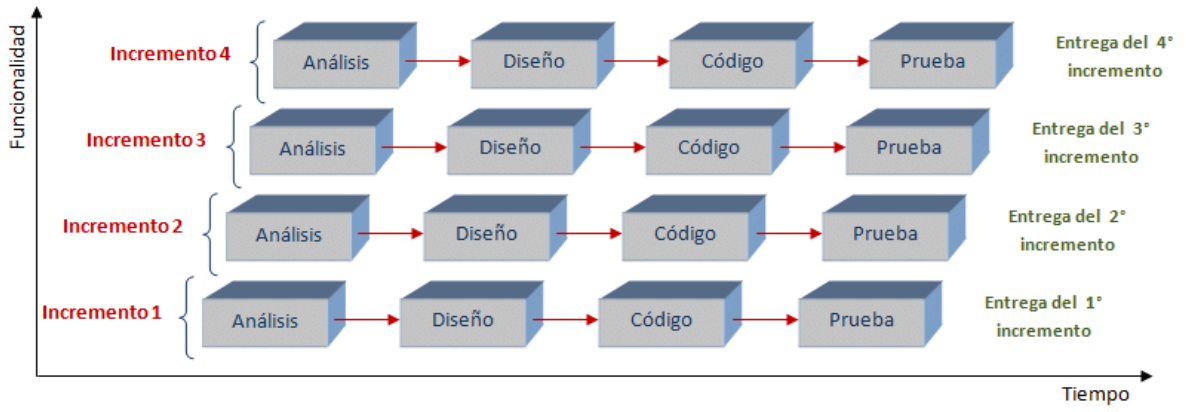

Ilustración 33. Fases de la metodología de desarrollo de software incremental.

Fuente: https://proyectosagiles.org/desarrollo-iterativo-incremental/

Es importante recalcar que cada iteración entrega una o un conjunto de funcionalidades completas y perfectamente probadas, lo que permitirá la retroalimentación temprana de los posibles usuarios/clientes y nos ofrecerá la

posibilidad de adaptar su diseño y desarrollo a las necesidades específicas del usuario, además de permitirnos obtener un feedback muy interesantes desde etapas muy tempranas del desarrollo, y agregar nuevas funcionalidades en siguientes incrementos teniendo en cuenta las necesidades y requisitos del usuario. Esto, como es lógico, conlleva un importante ahorro en tiempo e inversión por posibles cambios futuros, una vez el proyecto se encuentra en fases más avanzadas.

Además, el enfoque incremental es especialmente interesante en el contexto de este proyecto debido a que se alinea muy bien con el entorno de la nube y el despliegue en PaaS. La flexibilidad y escalabilidad que ofrecen las herramientas y servicios que ponen a nuestra disposición los proveedores de servicio, como es el caso de Amazon Web Service, nos dará la posibilidad de desplegar y probar iterativamente la aplicación en la nube.

Pasamos ahora a documentar cada una de las iteraciones del proyecto, que a partir de este momento comenzaremos a llamar sprint. Como hemos mencionado anteriormente, en cada sprint intentaremos incluir aquellas historias de usuario que consideremos que aporten más valor al producto, teniendo en cuenta y dando valor al feedback que iremos recibiendo al finalizar cada uno de estos sprint por parte de los usuarios.

4.6.1. Primer sprint

4.6.1.1. Historias de usuario seleccionadas

Para este primer sprint de nuestra aplicación, hemos decidido desarrollar las historias de usuario 1 y 2 (ver tabla 4). Se especificarán aquí además, las condiciones o criterios que deben cumplir cada historia de usuario en las pruebas de validación posteriores, para dar cómo válida esa historia de usuario concreta. A partir de este momento nos referiremos a ellos como criterios de validación. Cómo recordatorio y con los criterios de validación que deben cumplir añadidos, las historia de usuario 1 y 2 son las siguientes:

Nº historia de usuario	Actor	Casos de uso	Descripción	MoSCoW	Criterios de validación
	como	*quiero*	*para*		
1	usuario logueado / usuario no logueado	acceder a una pantalla inicial	obtener una explicación de la funcionalidad de la aplicación y tener acceso a las distintos usos que ofrece	Must have	- Al entrar a la aplicación se visualizará esta pantalla inicial.
2	usuario logueado / usuario no logueado	saber la media de puntos anotados por un jugador en una temporada	visualizar los resultados que coinciden con dicha búsqueda	Must have	- Devuelve el valor pedido cuando se introducen los parámetros correctos. - Devuelve mensaje de error en caso contrario.

Tabla 4. Historias de usuario que se van a implementar en el primer sprint.

¿Por qué se ha decidido incluir estas dos historias de usuario para esta primera versión de la aplicación? Consideramos que la aplicación tendrá una funcionalidad mínima con valor que poner a disposición de los usuarios en esta primera entrega. Hay que tener en cuenta que estamos en una etapa muy temprana del desarrollo, por lo que la aplicación debe ser básica y funcional al mismo tiempo, de modo que podamos recibir este feedback del que hablábamos anteriormente por parte de nuestros potenciales clientes.

4.6.1.2. Diseño

- **Diseño de interfaz**

En este punto comenzaremos realizando un diseño de la aplicación en este primer incremento o sprint. Se trata de una de las partes más importantes del desarrollo de software, más si cabe tratándose de una aplicación web. Hay que ser consciente de que la calidad de la interfaz puede ser la diferencia entre el éxito o fracaso de un

proyecto de este tipo, anteponiendose en esta temprana etapa a las funcionalidades que subyacen bajo esta capa.

Una buena interfaz de usuario ha de ser visualmente atractiva, con un diseño limpio y organizado que permita una fácil navegación. Se usarán controles simples y claros, se presentarán los resultados de manera organizada y se considerará la adaptabilidad a diferentes dispositivos. El objetivo es proporcionar una experiencia de usuario satisfactoria y agradable al interactuar con la aplicación.

Para desarrollar este primer mockup de la aplicación con las diferentes vistas de que consta este primer sprint, usaremos la aplicación Canva[2]. Hemos elegido esta herramienta debido a su sencillez, que no requiere descarga y que es completamente gratuita. Además es una herramienta que ya había usado en algunas asignaturas de la carrera y me resultaba más familiar que otras opciones disponibles en el mercado.

Los prototipos diseñados son los siguientes:

[2] https://www.canva.com/

Storyboard vista Inicio

Ilustración 34. Storyboard vista de Inicio de la aplicación.

Incluye un botón en un panel superior que nos dirigirá a la vista de búsquedas, un espacio bajo este panel donde habrá una o varias fotos de la NBA y por último una sección con fondo blanco donde se describe BasketApp, así cómo las funcionalidades que tenga la versión en que nos encontremos.

Storyboard vista Búsqueda media puntos de un jugador en una temporada

Ilustración 35. Storyboard vista Búsqueda media puntos de un jugador en una temporada.

Incluye el mismo panel superior que la vista de Inicio, de modo que podamos navegar entre ambas vistas sin ninguna dificultad. Justo debajo nos encontramos con el formulario que el usuario rellenará para realizar la búsqueda. Se podrá introducir el nombre de un jugador, y una temporada que esté comprendida entre 1946 (año en que se fundó la liga) y la actualidad. Una vez lanzada la petición aparecerá una nueva sección inferior con el resultado de la búsqueda.

- **Diagrama de secuencia**

Comenzaremos definiendo qué es un diagrama de secuencia, de modo que el lector sea capaz sin ningún ápice de dudas de entender su importancia dentro de este epígrafe de diseño del primer sprint de la aplicación. Cuando hablamos de diagrama de secuencia nos referimos a una representación visual que permite ver la secuencia de interacciones entre objetos o componentes en un sistema a lo largo del tiempo. Es una herramienta utilizada en el análisis y diseño de software para modelar el flujo de mensajes entre los distintos elementos del sistema. Un diagrama de secuencia se compone de los participantes, las líneas de vida de estos objetos a lo largo del tiempo, y flechas para modelar la interacción entre ellos. Destacar que su importancia radica en que este tipo de diagrama permite comprender de forma clara y concisa la manera en que los objetos interactúan entre sí y la forma en que fluye la información en la aplicación.

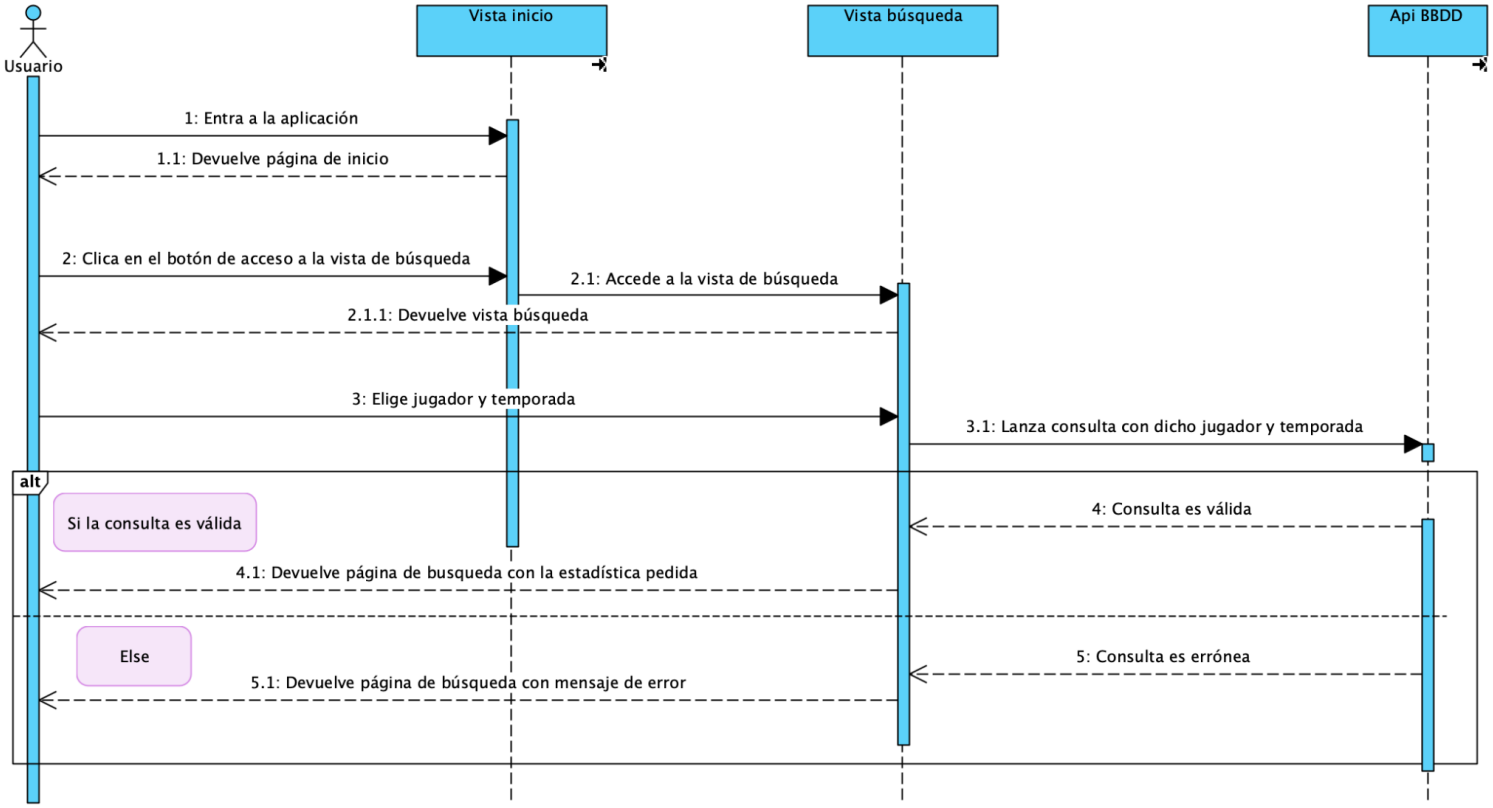

Ilustración 36. Diagrama de secuencia de la primera versión de la aplicación.

En este diagrama (importante ver ilustración 36) observamos la interacción entre el usuario y el sistema y el modo en que puede acceder a la aplicación, realizar la búsqueda de puntos anotados por un jugador en una temporada, y la posible respuesta de la aplicación en función de si los datos introducidos son correctos o no. Se corresponde como hemos comentado anteriormente con las historias de usuario 1 y 2.

En la historia de usuario principal del sprint, conocer la media de puntos anotados por un jugador a lo largo de una temporada, el usuario introducirá en los labels proporcionados en la vista búsqueda para ello, el nombre de un jugador que participe o haya participado en la liga, y una temporada en que formara parte de un equipo. Esta petición se lanzará contra la api, que en caso de que los parámetros sean correctos devolverá un json con la información que buscamos. Posteriormente

esta estadística podrá visualizarse en nuestra vista de búsqueda. En caso de parámetros erróneos se visualizará la pantalla búsqueda con este mensaje de error.

4.6.1.3. Implementación

Teniendo ya una versión inicial con un simple Hello world! desplegada en AWS, como comentamos en el apartado 4.2 de este proyecto, podemos comenzar a desarrollar la aplicación con las funcionalidades e historias de uso especificadas para este primer sprint. Intentaremos seguir el boceto diseñado en el apartado anterior, así como los diferentes diagramas también diseñados.

El modo de trabajar será el siguiente. Desarrollaremos la aplicación en local, en nuestro repositorio anteriormente conectado a AWS. Tendremos una rama principal, que es la que se encuentra desplegada en nuestro entorno, que hará las veces de rama de producción. De esta rama sacaremos ramas de desarrollo para los diferentes sprints, y una vez probadas fusionaremos con la rama de producción que se encuentra desplegada y automáticamente desplegará nuestra nueva versión.

Comenzaremos hablando a grandes rasgos de la tecnologías utilizadas para esta implementación (se desarrollarán más en profundidad en un epígrafe posterior).

Para el front-end de nuestra aplicación emplearemos el framework Bootstrap[3], que proporciona un conjunto de herramientas y componentes predefinidos para el diseño e interfaz de usuario. Esto facilitará la creación de una interfaz visualmente atractiva y responsive, aspectos que considero fundamentales en una aplicación web de este tipo. Debido a que el objeto de este proyecto es mostrar el proceso de desarrollo de una aplicación en PaaS, y no la aplicación en sí misma, hemos considerado que hacer uso de esta potente herramienta para crear una interfaz de usuario simple y atractiva al mismo tiempo, es la mejor opción.

[3] https://getbootstrap.com/

Para el back-end de la aplicación se usará Python como lenguaje de programación y Flask como framework de despliegue de la aplicación. La razón como hemos visto en el punto dedicado a hablar de las herramientas utilizadas anteriormente, es un simple cuestión de mi mayor conocimiento y experiencia en el uso del framework. Las consultas se lanzarán a la API de balldontlie[4], que proveen una API totalmente gratuita, que no requiere registro ni clave para su uso, y que ofrece estadísticas completamente actualizadas de la NBA con un histórico de temporadas que comprende desde 1946 hasta la actualidad.

Debo comentar aquí que la idea inicial era usar la API de SportsDatabase (https://sportsdatabase.com/NBA/query.html). Debido a los fines académicos de su uso, desde el soporte de la base de datos, nos mandaron un *user* y un *token* gratuitos para hacer uso de la API. Tras varios intentos infructuosos de conectar vía API decidimos probar otras alternativas, y balldontlie ofrece toda la funcionalidad que necesitamos.

El siguiente paso es conectar vía API con la base de datos. Hicimos un primer intento de conexión utilizando la biblioteca Requests debido a que me encuentro cómodo con su uso porque me es bastante familiar en mi entorno laboral. La conexión fue exitosa, así que solo faltaba lanzar las llamadas pertinentes con los datos que el usuario introduzca por teclado. El código es el que vemos en la ilustración 37.

[4] https://app.balldontlie.io/

```
url_player = f"https://www.balldontlie.io/api/v1/players?search={jugador}"

player = requests.get(url_player)
points=-1

if player.status_code == 200 and jugador!="":
    data_player = player.json()
    if data_player['data']:
        player_id = data_player['data'][0]['id']

        url_puntos = f"https://www.balldontlie.io/api/v1/season_averages?player_ids[]={player_id}&season={temporada}"

        puntos = requests.get(url_puntos)

        if puntos.status_code == 200:
            data_puntos = puntos.json()
            if data_puntos['data']:
                points = data_puntos['data'][0]['pts']
```

Ilustración 37. Conexión vía API con balldontlie usando la biblioteca Request.

La aplicación lanzada en local y en mi entorno de python, funcionaba perfectamente y cumplía con las especificaciones de este primer sprint. Así que sólo faltaba mergear esta rama de desarrollo con la rama principal de producción conectada con mi entorno de AWS para desplegar la aplicación y pasar al siguiente ítem de desarrollo software.

Pero, ¿qué ocurrió cuando desplegamos esta primera versión funcional de la aplicación en el entorno de AWS? La aplicación dejó de funcionar. Elastic Beanstalk, aplicación que usamos para desplegar nuestra aplicación y entorno de AWS, proveé diferentes opciones para solucionar errores. Los mensajes que aparecen en la pestaña Eventos no ofrecen demasiada información, como se observa en la ilustración 38.

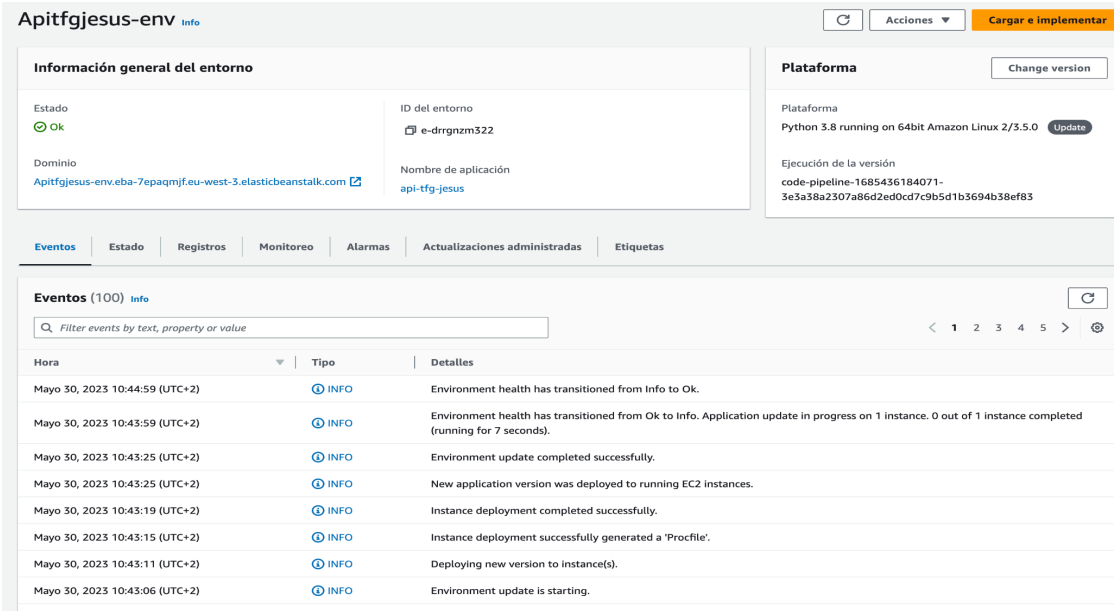

Ilustración 38. Pantalla información del entorno de AWS.

Para obtener información más detallada podemos descargar los logs del entorno desde la pestaña Registros. Aquí puedes seleccionar entre descargar los últimos 100 registros, o un histórico completo, como observamos en la ilustración 39.

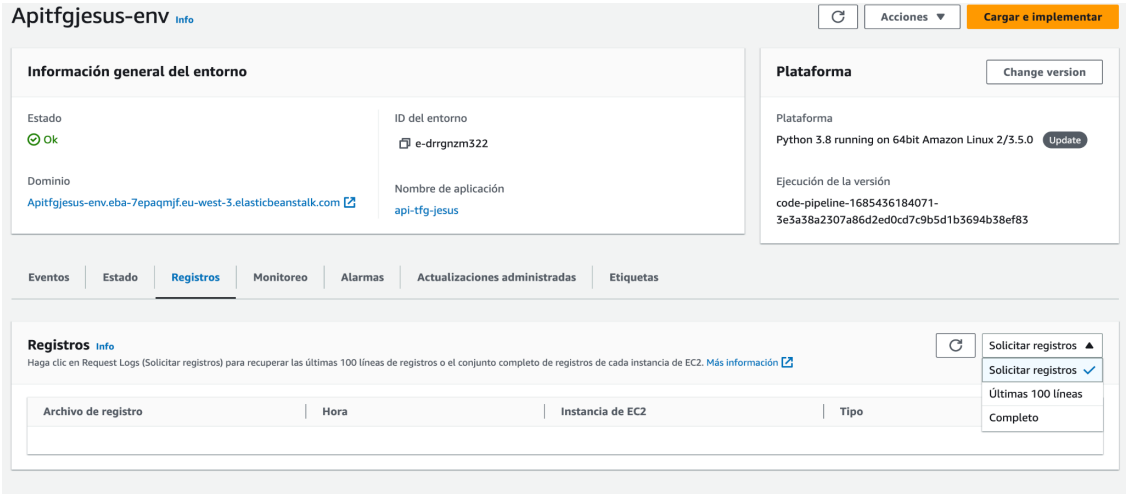

Ilustración 39. Pantalla de registros del entorno de AWS.

Tras varios intentos fallidos, concluí que se debía a un fallo en los paquetes de Python instalados en el entorno, o a problemas derivados de la versión del paquete

requests, objeto principal del conflicto. Ante la dificultad de resolver esto, optamos por utilizar otra de las bibliotecas HTTP disponibles para Python. Cambié la lógica de la conexión con la API y las llamadas, de modo que el código quedó como observamos en la ilustración 40.

```
base_url = "www.balldontlie.io"
endpoint = "/api/v1/players"

context = ssl._create_unverified_context()

conn = http.client.HTTPSConnection(base_url, context=context)
headers = {'Content-type': 'application/json'}

params = f"search={jugador}"
url_player = f"{endpoint}?{params}"
url_player = url_player.replace(' ', '%20')
try:
    conn.request("GET", url_player, headers=headers)
except:
    return ["Alguno de los parámetros introducidos no es correcto. Escribe un nombre de ju

response = conn.getresponse()

if jugador != "" and temporada != "":

    data = json.loads(response.read().decode())
    if data["data"]:
        player_id = data["data"][0]["id"]

        jugador = data["data"][0]["first_name"] + " " + data["data"][0]["last_name"]
        equipo = data["data"][0]["team"]["full_name"]

        endpoint = f"/api/v1/season_averages?season={temporada}&player_ids[]={player_id}"
        conn.request("GET", endpoint, headers=headers)
        response = conn.getresponse()
        try:
            data = json.loads(response.read().decode())
        except:
            return ["Alguno de los parámetros introducidos no es correcto. Escribe un nomb
```

Ilustración 40. Conexión vía API con balldontlie usando la biblioteca http.client.

En esta segunda versión del primer sprint el proceso de obtención de los datos consiste en realizar una primera llamada para obtener el ID del jugador buscado. Una vez obtenido el ID, se realiza una segunda llamada, pasando el nombre del jugador y la temporada como parámetros, para obtener la media de puntos de ese jugador en esa temporada específica.

En local seguía funcionando correctamente, y al desplegarlo en nuestro entorno de AWS, a diferencia de la vez anterior, no nos encontramos con ningún problema y todo funcionaba a la perfección.

4.6.1.4. Pruebas de validación

Pasamos ahora, en este nuevo apartado de Pruebas de validación, a mostrar los resultados obtenidos tras el desarrollo de esta primera y temprana versión de nuestra aplicación. Con este paso pretendemos probar las funcionalidades añadidas, y descubrir posibles fallas en su funcionamiento, de modo que llegue a los usuarios perfectamente probado y libre de errores.

Iremos historia de usuario a historia de usuario, asegurándonos que los criterios de validación que establecimos para cada una de ellas quede resuelto. De este modo y solo de este modo, daremos por validado el punto.

- **Historia de usuario nº 1: acceder a una pantalla inicial.** Criterio de validación:
    - Al entrar a la aplicación se visualizará esta pantalla inicial.

Respecto a la visualización de la pantalla de Inicio, y la navegación entre pantallas, no se observa ningún problema. Además el diseño de interfaz es totalmente responsive, tal y como se pretendía, y se visualiza correctamente en cualquier tamaño de pantalla o dispositivo, como observamos en las ilustraciones 41 y 42.

Ilustración 41. Vista de Inicio de la aplicación versión web en la versión 1.

Ilustración 42. Vista de Inicio de la aplicación versión móvil en la versión 1.

- **Historia de usuario nº 2: saber la media de puntos anotados por un jugador en una temporada.** Criterios de validación:

- Devuelve el valor pedido cuando se introducen los parámetros correctos.
- Devuelve mensaje de error en caso contrario.

Comenzamos esta segunda historia de usuario accediendo a la vista de Búsqueda de media de puntos anotados por un jugador en una temporada (ilustración 43).

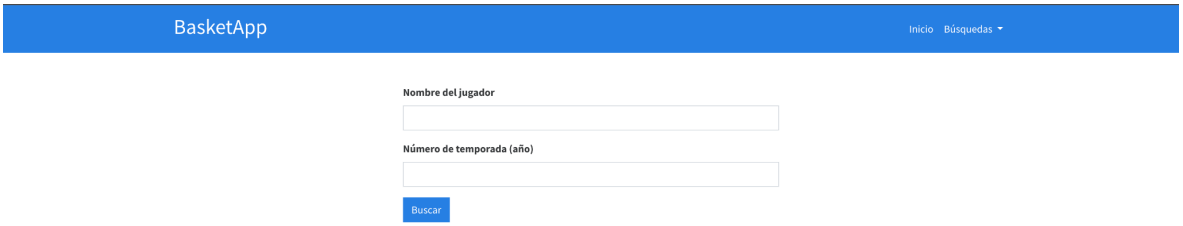

Ilustración 43. Vista de Búsqueda media puntos por temporada de la aplicación en la versión 1.

Pasamos ahora a probar la historia de usuario o funcionalidad principal de este primer sprint, conocer la media de puntos anotados por un jugador a lo largo de una temporada.

Las pruebas consistieron en introducir diferentes valores, tanto para jugador como para temporada, con el fin de observar las diferentes respuestas de la aplicación. Los diferentes posibles casos son los siguientes, añadiendo una captura adjunta justo debajo de cada uno de ellos:

- Introducir un nombre de jugador que ha jugado en la liga y una temporada en la que formó parte de un equipo. El resultado es satisfactorio, devolviendo la respuesta correcta al instante.

Ilustración 44. Vista de Búsqueda media de puntos por temporada con parámetros correctos.

- Introducir nombre de jugador, válido o no, pero dejar vacío el campo temporada o viceversa. El resultado es de nuevo correcto, pues devuelve el mensaje de error que programamos.

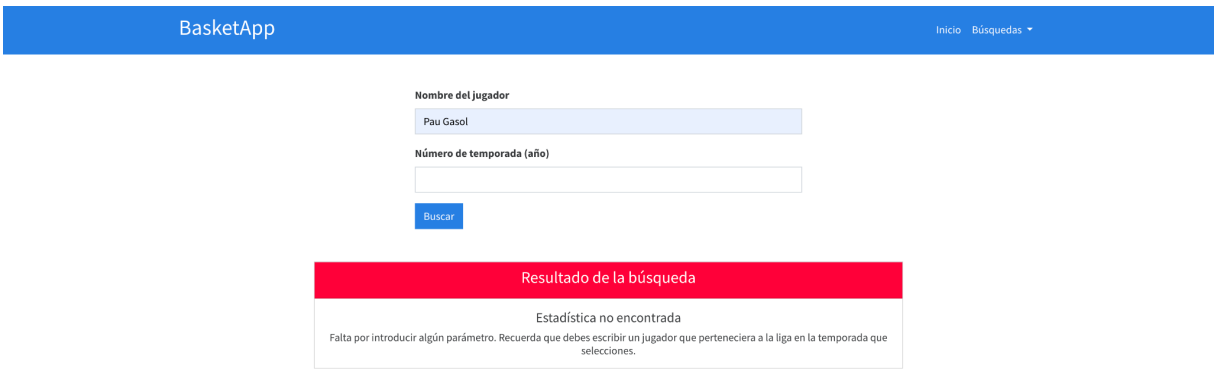

Ilustración 45. Vista de Búsqueda media de puntos por temporada dejando campo vacío.

- Introducir un nombre de jugador válido pero una temporada errónea. El mensaje obtenido es el correcto para este escenario.

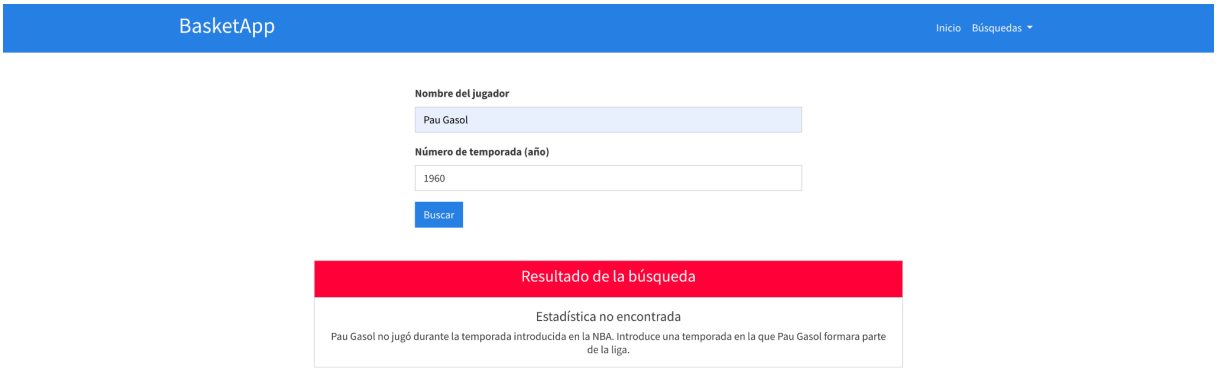

Ilustración 46. Vista de Búsqueda media de puntos por temporada con temporada errónea.

- Introducir un nombre inventado. De nuevo la aplicación funciona como debería.

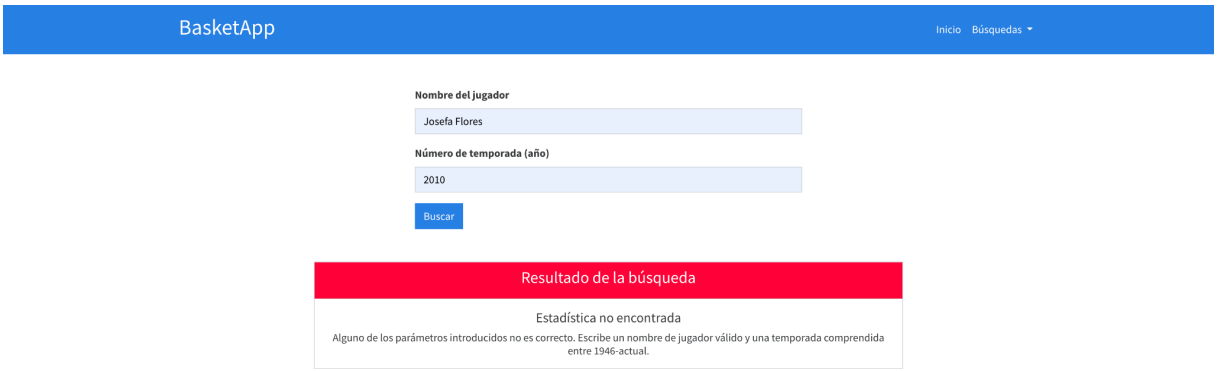

Ilustración 47. Vista de Búsqueda media de puntos por temporada con jugador incorrecto.

- Introducir letras en la temporada. El comportamiento es correcto, y los tiempos de ejecución son más que aceptables.

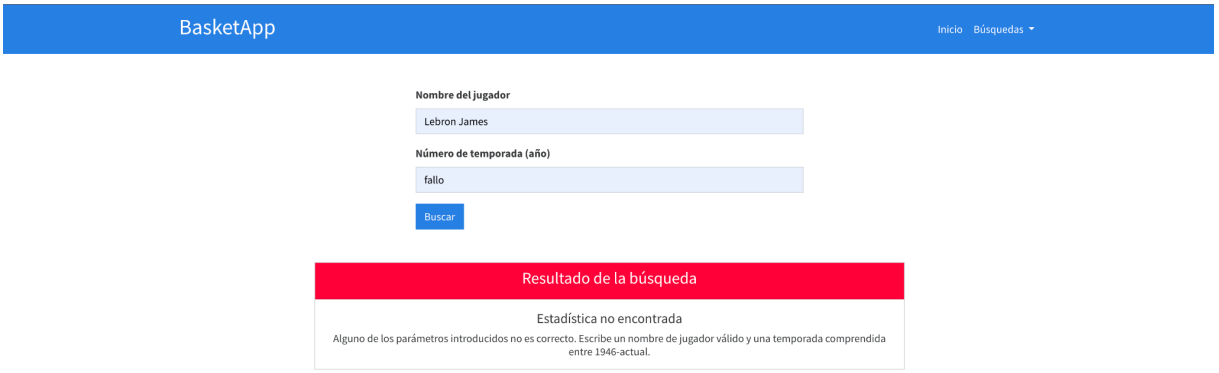

Ilustración 48. Vista de Búsqueda media de puntos por temporada con temporada incorrecta.

Tras repasar cada historia de usuario implementada en este primer sprint, y una vez validados los criterios de validación identificados para cada una de ellas, podemos dar por validado esta primera versión de la aplicación.

4.6.1.5. Feedback

Para éste último apartado del primer sprint, he construido un pequeño cuestionario con 3 preguntas sobre la aplicación, de modo que los usuarios que prueben la versión que se encuentra desplegada en nuestro entorno de producción nos proporcionen este feedback del que hablábamos anteriormente. Se trata de un punto muy importante dentro de la metodología de desarrollo que estamos llevando a cabo.

El cuestionario construido es el siguiente:

1. ¿Cuál es su opinión respecto a la aplicación en general? ¿La experiencia durante su uso fue agradable? ¿Considera que la navegación ha sido intuitiva y sencilla?
2. ¿Ha encontrado útil la funcionalidad de búsqueda de la media de puntos anotados por un jugador de la NBA en una temporada específica? ¿Considera que las respuestas recibidas son satisfactorias?

3. ¿Se le ocurre alguna funcionalidad adicional que le gustaría ver en la aplicación en siguientes versiones? ¿Algún aspecto en concreto de las funcionalidades ya existentes que considere que podría mejorarse y de este modo conseguir una experiencia de uso más satisfactoria?

Esta primera versión de la aplicación no tuvo una mala acogida, siendo 12 los usuarios que tuvieron la oportunidad de probarla y posteriormente rellenar este pequeño cuestionario. Los resultados obtenidos para las dos primeras preguntas, referentes a la utilidad y calidad de la aplicación en este punto, pueden verse reflejados en las ilustraciones 49 y 50.

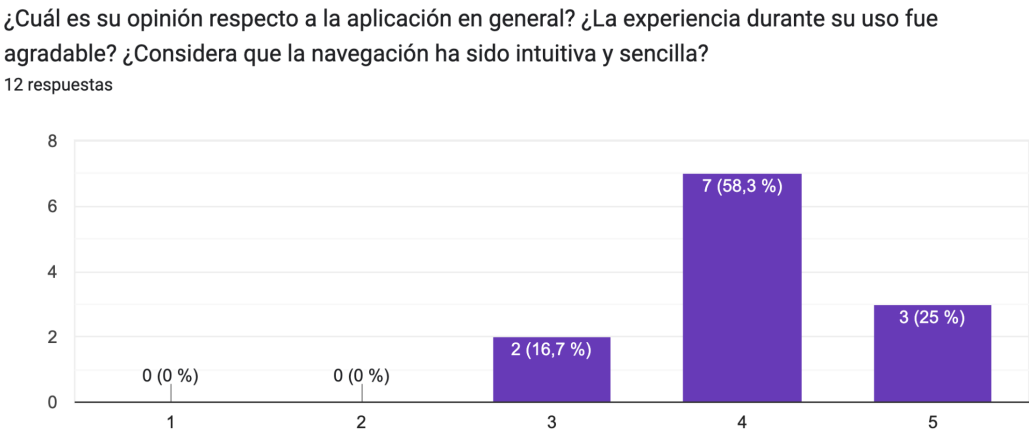

Ilustración 49. Gráfico de barras respuestas a la primera pregunta del cuestionario en el primer sprint.

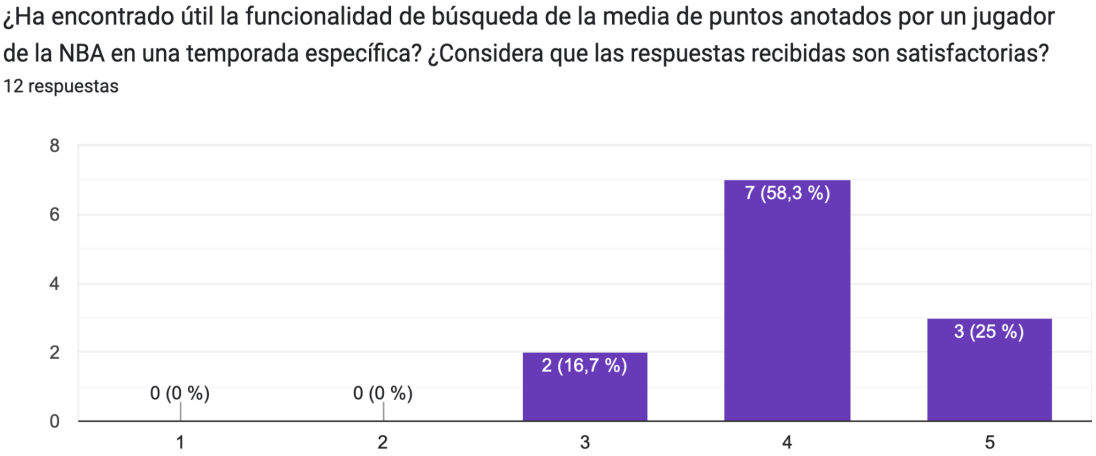

Ilustración 50. Gráfico de barras respuestas a la segunda pregunta del cuestionario en el primer sprint.

No obstante, la pregunta realmente interesante para este apartado, y cuestión que más valor a aportar a este paso del desarrollo del software es la tercera.

Las respuestas dadas por estos primeros usuarios han sido de lo más variopintas, aportando una visión en muchos casos muy interesante. Han sido las siguientes:

- *El título de la aplicación "BasketApp" permite que uno pinche en él, pero no lleva a ningún sitio. Quizá, en vez de la opción "inicio", podría usarse ese "botón" para ir a la página de inicio. En cuanto a la búsqueda, sería interesante que no dejara enviar la solicitud si falta algún parámetro obligatorio. También sería interesante que, una vez introducido el jugador, nos indicara qué temporadas hay disponibles para él. En vez de usar un campo de texto para el año de la temporada, sería interesante que hubiese un desplegable con los años válidos para ese jugador. Finalmente, no me queda claro si cuando introduzco el año 2012, por ejemplo, se está tomando en cuenta el año natural 2012, o la temporada 2011/2012 o la temporada 2012/2013.*
- *Añadir en la función de búsqueda, un apartado menos específico, para ver la media de puntos de un jugador en distintas temporadas.*
- *Añadir una parte con ejemplos de jugadores famosos para poder ver sus estadísticas.*
- *Me gustaría tener una lista de ejemplos con los máximos anotadores de la última temporada para poder tener los nombres visualmente cerca.*
- *Realizaría un selector tanto de jugador como de año. Para que resulte más fácil la selección.*
- *Me gustaría que la aplicación contenga información sobre la fuente de datos utilizada. Además encontraría interesante disponer de más tipos de búsquedas.*
- *Más intuitiva a la hora de acceder y ver los usos que tiene la app.*
- *Me parecería interesante que la búsqueda devolviera, no solo el dato de la media de puntos, sino algún gráfico relevante, como un evolutivo de los puntos por mes de esa temporada. También veo algo confuso título de BasketApp del panel superior izquierdo, es clicable pero no tiene ninguna*

funcionalidad. La intuición me dice que debería llevarte a la página inicial de la aplicación, por lo que el botón Inicio ya no tendría sentido.

- *Podría incluirse en siguientes versiones un sistema de registro y autenticación? Sería interesante tener un historial de búsquedas, de modo que a través de mi usuario pueda visualizar un histórico de la búsquedas que he realizado. Otra idea que me surge es añadir una vista con estadísticas NBA relevantes, pudiendo elegir el usuario entre ver histórico completo, o temporada actual, por ejemplo, e incluyendo gráficos, que visualmente creo que podría quedar una funcionalidad muy chula.*
- *Me gusta, pero se podría añadir algo de info a la búsqueda, no me gusta el baloncesto y no conozco jugadores.*
- *Que la búsqueda permita incluir varios jugadores y mostrar un gráfico comparando sus temporadas, quizás un gráfico de líneas o de barras podría estar bien.*

De los resultados de este cuestionario una vez vista y probada la aplicación por una muestra de usuarios significativa, extraemos que si bien esta primera versión de la aplicación ha gustado en términos generales, contiene aspectos de usabilidad que quizás habían pasado por alto que se deben mejorar. Además se han propuesto nuevas funcionalidades que no se habían tenido en cuenta en un principio y nos han resultado muy interesantes.

El siguiente paso por tanto es redefinir nuestras historias de usuario y decidir cuáles se van a implementar en el segundo sprint de desarrollo.

4.6.2. Segundo sprint.

4.6.2.1. Historias de usuario seleccionadas

Atendiendo a la retroalimentación aportada por los usuarios en el apartado feedback del primer sprint, hemos decidido definir algunas historias de usuario extra. La retroalimentación dada por los usuarios nos lleva a darnos cuenta de algunas

funcionalidades y mejoras que hemos considerado suficientemente relevantes para ser incluidas en nuestro backlog de tareas.

Nº historia de usuario	**Actor**	**Casos de uso**	**Descripción**	**MoSCoW**
	como	*quiero*	*para*	
11	usuario logueado / usuario no logueado	acceder a una vista con información sobre la base de datos de donde se está extrayendo la información	conocer la base de datos que reporta las estadísticas que estamos obteniendo	Could have
12	usuario logueado / usuario no logueado	acceder rápidamente a la página de inicio pulsando sobre el logotipo de la aplicación	obtener un comportamiento similar al de la mayoría de aplicaciones	should have
13	usuario logueado / usuario no logueado	que la pagina no me permita iniciar la búsqueda hasta haber completado todos los filtros obligatorios	no equivocarme y no obtener mensajes de error	should have
14	usuario logueado / usuario no logueado	que las vistas de búsqueda incluya un título identificativo	saber en qué tipo de búsqueda me encuentro	should have

Tabla 5. Historias de usuario identificadas y añadidas tras la retroalimentación del primer incremento.

Una vez identificadas y definidas las nuevas historias de usuario, el siguiente paso es seleccionar aquellas historias que incluiremos en el segundo sprint.

Se ha atendido a diferentes razones para la elección, entre las que destaca la relación valor aportado/tiempo de implementación de cada una de ellas. Finalmente las elegidas han sido las historias de usuario 3, 4, 5, 11, 12, 13 y 14. Incluimos de

nuevo, al igual que hicimos en el primer sprint de este desarrollo, los criterios de validación que usaremos para validar cada una de estas funcionalidades.

Nº historia de usuario	**Actor**	**Casos de uso**	**Descripción**	**MoSCoW**	**Criterios de validación**
	como	*quiero*	*para*		
3	usuario logueado / usuario no logueado	saber la media de asistencias dadas por un jugador en una temporada	visualizar los resultados que coinciden con dicha búsqueda	Must have	- Devuelve el valor pedido cuando se introducen los parámetros correctos. - Devuelve mensaje de error en caso contrario.
4	usuario logueado / usuario no logueado	saber la media de rebotes cogidos por un jugador en una temporada	visualizar los resultados que coinciden con dicha búsqueda	Must have	- Devuelve el valor pedido cuando se introducen los parámetros correctos. - Devuelve mensaje de error en caso contrario.
5	usuario logueado / usuario no logueado	saber las victorias de un equipo en una temporada	visualizar los resultados que coinciden con dicha búsqueda	Should have	- Devuelve el valor pedido cuando se introducen los parámetros correctos. - Devuelve mensaje de error en caso contrario.
11	usuario logueado / usuario no logueado	acceder a una vista con información sobre la base de datos de donde se está extrayendo la información	conocer la base de datos que reporta las estadísticas que estamos obteniendo	Could have	- Al entrar a la vista ‘Información de la BBDD’ visualizamos de modo correcto toda la info sobre la API utilizada para obtener las estadísticas.
12	usuario logueado / usuario no logueado	acceder rápidamente a la página de inicio pulsando sobre el logotipo de la aplicación	obtener un comportamiento similar al de la mayoría de aplicaciones	should have	- El botón BasketApp enlaza con la vista ‘Inicio’ correctamente desde cualquier punto de la aplicación.
13	usuario logueado / usuario	que la pagina no me permita iniciar la búsqueda hasta	no equivocarme y no obtener mensajes de	should have	- El botón Buscar aparece deshabilitado en las 4 vistas de búsqueda.

	no logueado	haber completado todos los filtros obligatorios	error		- Aparece como enabled cuando rellenamos los dos campos y nos deja lanzar la búsqueda.
14	usuario logueado / usuario no logueado	que las vistas de búsqueda incluya un título identificativo	saber en qué tipo de búsqueda me encuentro	should have	- Las 4 vistas de búsqueda incluyen un título identificativo.

Tabla 6. Historias de usuario seleccionadas para el segundo sprint.

¿Por qué hemos decidido implementar estas 7 historias de usuario cuando además en el primer incremento sólo implementamos 2? Siguiendo la metodología de desarrollo ágil que hemos decidido utilizar para desarrollar la aplicación, consideramos que era fundamental hacer un primer incremento con funcionalidad mínima que los usuarios pudieran ver y probar lo antes posible, tal y como ya comentamos en la documentación del primer incremento. Una vez los usuarios han visto y probado la aplicación, hemos comprobado que en términos generales ha tenido buena acogida y se ha superado el miedo inicial a que la aplicación no gustase y no tuviese sentido invertir tiempo en desarrollarla, se ha decidido incluir un mayor número de funcionalidades e historias de usuario para este segundo sprint.

4.6.2.2. Diseño

- **Diseño de interfaz**

Los prototipos diseñados para este segundo incremento pueden verse en las ilustraciones 51, 52, 53 y 54.

Storyboard vista Información sobre la BBDD

Ilustración 51. Storyboard vista de Información sobre la BBDD.

Incluye un botón en el panel superior de todas las vistas que nos dirige a esta nueva pantalla. Una vez dentro de la vista vemos un título, una descripción de la API que estamos utilizando, un botón que nos lleva a información más detallada de la API en su web oficial, y algunas cajas con características reseñables de la misma.

Storyboard vista Búsqueda media asistencias de un jugador en una temporada

Ilustración 52. Storyboard vista Búsqueda media asistencias de un jugador en una temporada.

Vista análoga a la vista en la ilustración 31, búsqueda de media de puntos por temporada de un jugador. Incluye el botón de Información BBDD del storyboard anterior, un título identificativo para saber la búsqueda en que nos encontramos, y vemos que el botón búsqueda lo hemos convertido en un desplegable con las diferentes búsquedas.

Storyboard vista Búsqueda media rebotes de un jugador en una temporada

Ilustración 53. Storyboard vista Búsqueda media rebotes de un jugador en una temporada.

Vista análoga a la anterior, con la salvedad de que esta vista devuelve media de rebotes en vez de asistencias.

Storyboard vista Búsqueda victorias por temporada equipo

Ilustración 54. Storyboard Búsqueda victorias equipo en x temporada.

Incluye un formulario donde podremos introducir un equipo de la NBA y una temporada comprendida entre 1946 y la actualidad. Una vez lanzada la búsqueda obtendremos el resultado en un panel inferior.

- **Diagrama de secuencia**

En los diagramas de secuencia de las funcionalidades de este segundo incremento observamos de nuevo la interacción entre usuario, sistema y base de datos para devolver las estadísticas de los diferentes tipos de búsquedas, ver ilustración 55

para Búsqueda de media de asistencias en una temporada que será análoga a Búsqueda de media de rebotes y Búsqueda victorias por equipo, razón por la cual sólo haremos el diagrama de la primera.

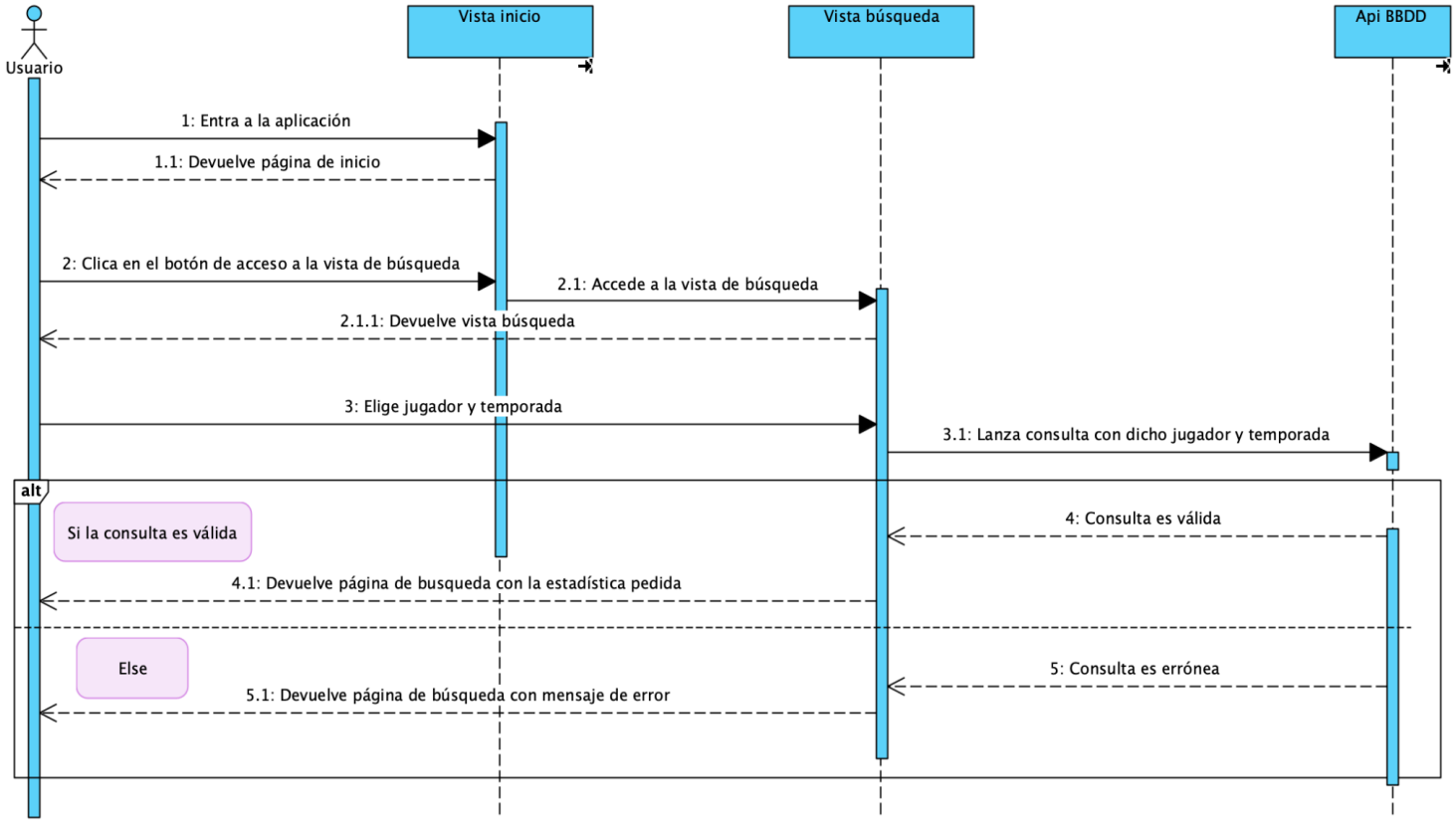

Ilustración 55. Diagrama de secuencia de la historia de usuario Búsqueda de media de asistencias en una temporada de un jugador.

A continuación en el diagrama de la ilustración 56 observamos la interacción del usuario con la vista de Información sobre la BBDD y la propia web de Balldontlie.

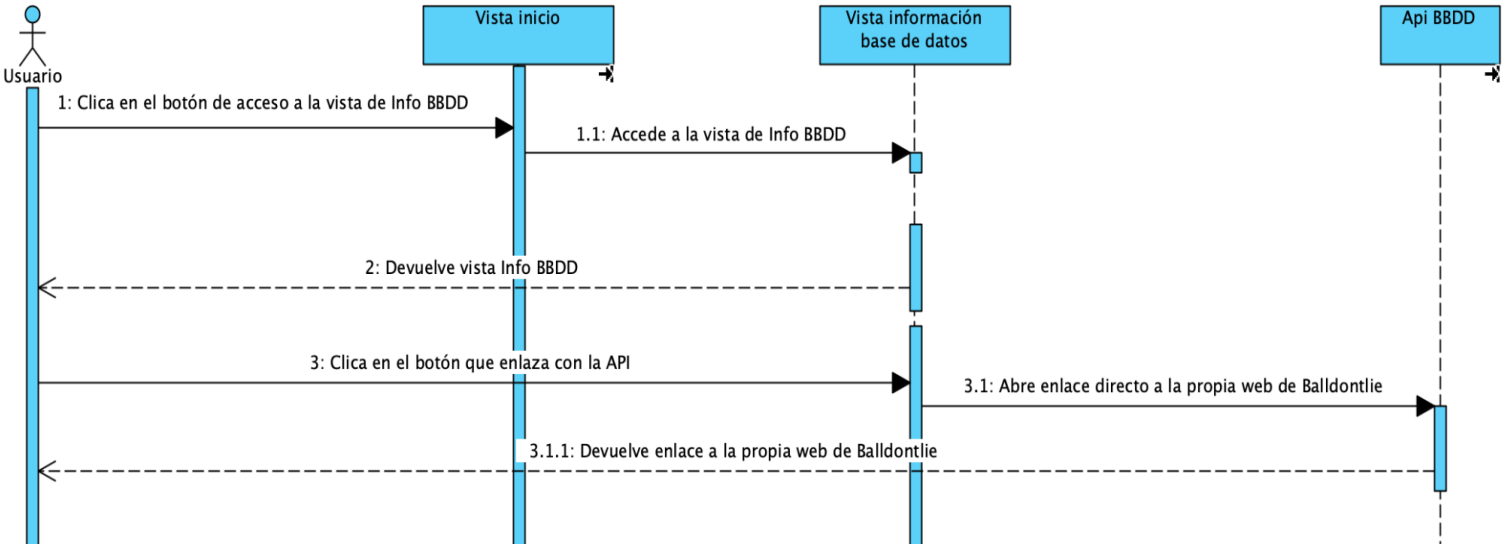

Ilustración 56. Diagrama de secuencia de la historia de usuario Información sobre la BBDD.

4.6.2.3. Implementación

La implementación de las historias de usuario definidas para este sprint, comenzó por incluir 3 nuevos tipos de búsqueda a la aplicación. En el primer sprint de nuestro software sólo podíamos obtener la estadística de media de puntos obtenida por un jugador en una temporada.

Comenzamos por crear una nueva rama en nuestro repositorio que parta de la versión estable de nuestro entorno de producción, donde incluiremos todas las nuevas funcionalidades y, una vez probadas y validadas, procederemos a fusionar con nuestra rama principal, y gracias al modelo Cloud Computing que hemos implementado, se desplegará automáticamente en nuestro entorno de producción. Será, por usar un lenguaje lo más similar posible a un proyecto empresarial más profesionalizado, nuestra subida a producción.

Implementamos las nuevas búsquedas utilizando llamadas a la API similares a la búsqueda de puntos que ya teníamos implementada, así que esta parte nos resultó

relativamente sencilla. Los endpoint que debíamos atacar eran similares a la anterior. Cabe destacar aquí que, en lo que respecta al front-end, hemos decidido cambiar el botón que teníamos para nuestra vista de búsqueda de media de puntos por un desplegable que aglutina los 4 tipos de búsquedas que ahora tendrá la aplicación.

A continuación desarrollamos la vista explicativa de la API que estamos usando como base de datos para la aplicación e incluimos un nuevo botón en el panel superior de nuestras vistas para dirigirnos a esta nueva vista. Como aspecto reseñable de esta parte incluimos un botón que nos enlaza con la web de Balldontlie.

Una vez desarrollado este segundo incremento e implementadas cada una de las historias de usuario que definimos para él, procedimos a fusionar la rama con la de producción como comentamos anteriormente. En esta ocasión y a diferencia de lo sucedido en el primer incremento no tuvimos ninguna dificultad con nuestro entorno de AWS. La experiencia conseguida durante la implementación del primer sprint nos ayudó mucho en esta ocasión para evitar problema derivados de la subida

4.6.2.4. Pruebas de validación

De nuevo y de forma análoga a lo hecho para la documentación del primer sprint, detallaremos las pruebas realizadas a la aplicación historia de usuario a historia de usuario, intentando en la medida de lo posible centrarnos en validar los criterios de validación definidos para cada una de ellas.

- **Historia de usuario nº 11: acceder a una vista con información sobre la base de datos de donde se está extrayendo la información.** Criterio de validación:
    - Al entrar a la vista Información de la BBDD visualizamos de modo correcto toda la información sobre la API utilizada para obtener las estadísticas.

Las pruebas que realizaremos para validar este criterio de validación será acceder a esta nueva vista y comprobar que devuelve la información que debería, y es accesible desde cualquier vista (ver ilustración 57).

Ilustración 57. Vista de Información sobre la BBDD.

- **Historia de usuario nº 12: acceder rápidamente a la página de inicio pulsando sobre el logotipo de la aplicación.** Criterio de validación:
    - El botón BasketApp enlaza con la vista Inicio correctamente desde cualquier punto de la aplicación.

Las pruebas realizadas para esta historia de usuario consisten en comprobar que la vista Inicio es accesible desde cualquier vista clicando sobre el botón BasketApp.

Comprobamos que la funcionalidad ahora es mucho más intuitiva y similar a aplicaciones parecidas.

- **Historia de usuario nº 13: que la página no me permita iniciar la búsqueda hasta haber completado todos los filtros obligatorios.** Criterio de validación:
    - El botón Buscar aparece deshabilitado en las 4 vistas de búsqueda.
    - Aparece como enabled cuando rellenamos los dos campos y nos deja lanzar la búsqueda.

Esta historia de usuario compete a las 4 vistas de búsqueda. Ahora el botón Buscar debería aparecer deshabilitado al entrar en las búsquedas, y cambiar a enabled cuando los dos campos obligatorios estén rellenos. Momento en el que se podrá lanzar la búsqueda. Observamos esto en la ilustración 58 donde vemos la vista con el botón deshabilitado por estar vacíos los campos, y en la ilustración 59 con el botón funcional una vez rellenados los dos campos.

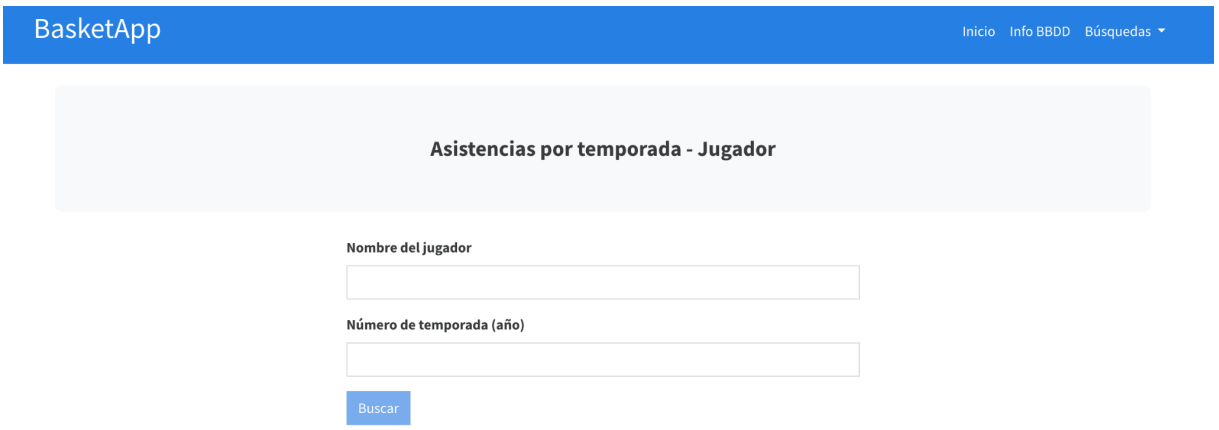

Ilustración 58. Vista de búsqueda con botón disabled.

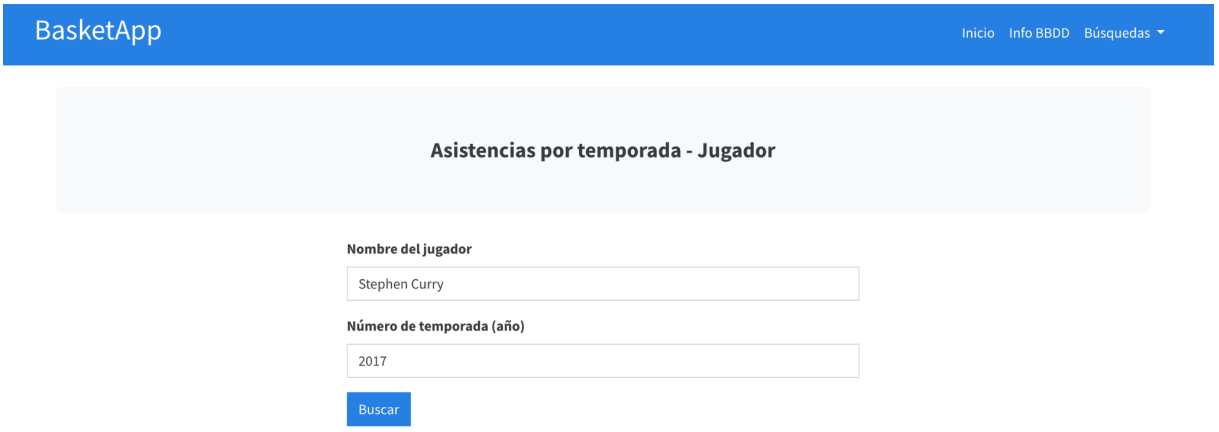

Ilustración 59. Vista de búsqueda con botón enabled.

Damos por tanto la historia de usuario como validada.

- **Historia de usuario nº 14: que las vistas de búsqueda incluyan un título significativo.** Criterio de validación:
    - Las 4 vistas de búsquedas incluyen un título significativo.

Las pruebas que realizaremos para validar esta 14 historia de usuario serán muy sencillas. Accederemos a las vistas de búsqueda y comprobaremos que incluyen un título. Nos ha parecido un aspecto muy importante en este punto del desarrollo pues antes de incluir esta funcionalidad la navegación entre vistas era algo confusa (ver ilustración 60).

Ilustración 60. Vista de Búsqueda con título.

- **Historia de usuario nº 3: saber la media de asistencias dadas por un jugador en una temporada.** Criterio de validación:
  - Devuelve el valor pedido cuando se introducen los parámetros correctos.
  - Devuelve mensaje de error en caso contrario.

Las pruebas en este caso consistirán en repetir lo hecho para la historia de usuario 2: saber la media de puntos anotados por un jugador en una temporada, dado que los criterios de validación definidos han sido los mismos. Por este motivo pondremos sólo una captura cuando la búsqueda se realice correctamente (ver ilustración 61) y otra con el campo jugador incorrecto (ilustración 62).

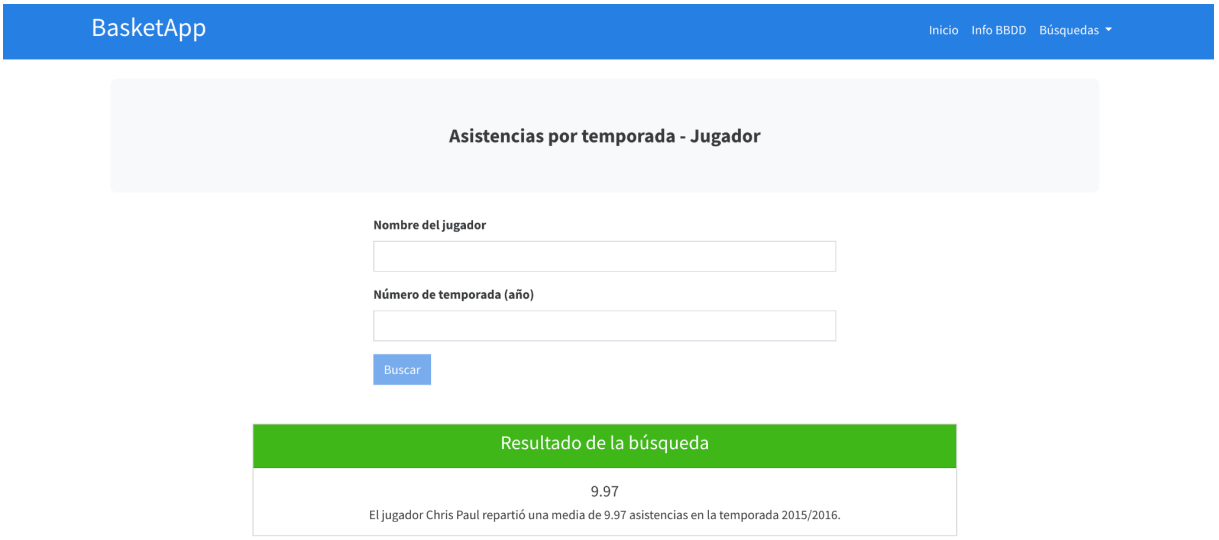

Ilustración 61. Vista de Búsqueda media de asistencias por temporada correcta.

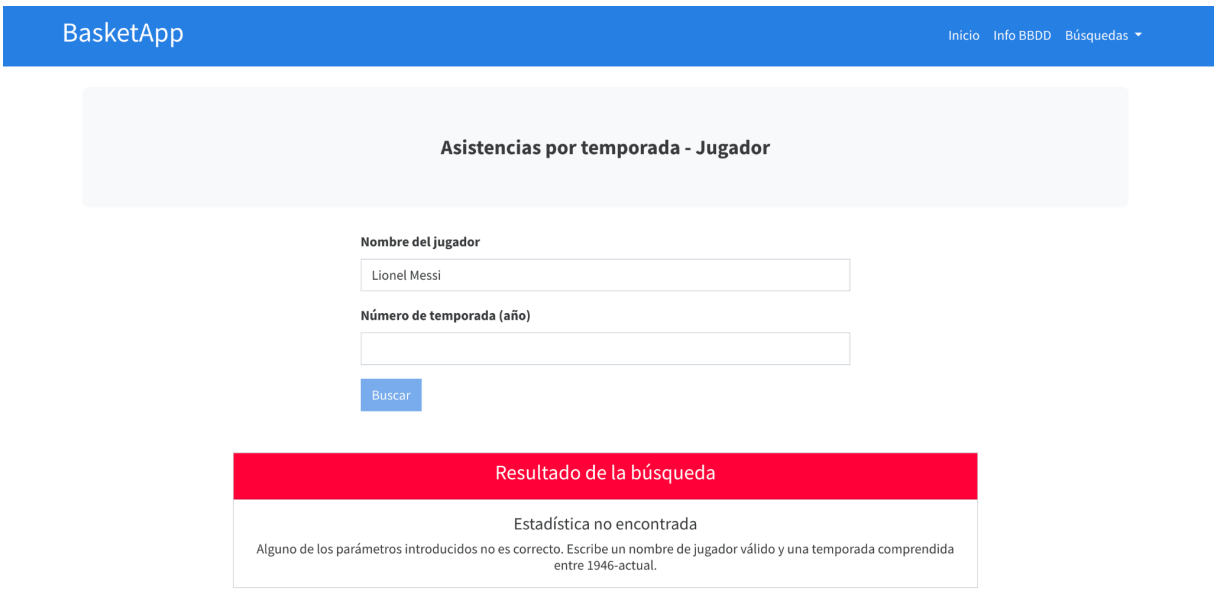

Ilustración 62. Vista de Búsqueda media de asistencias por temporada con nombre de jugador inexistente.

- **Historia de usuario nº 4: saber la media de rebotes capturados por un jugador en una temporada.** Criterio de validación:
    - Devuelve el valor pedido cuando se introducen los parámetros correctos.
    - Devuelve mensaje de error en caso contrario.

De nuevo repetimos las mismas pruebas realizadas por las historias de usuario 2 y 3. Vemos la búsqueda realizada correctamente en la ilustración 63 y la respuesta cuando el jugador no jugó en la temporada introducida en la ilustración 64.

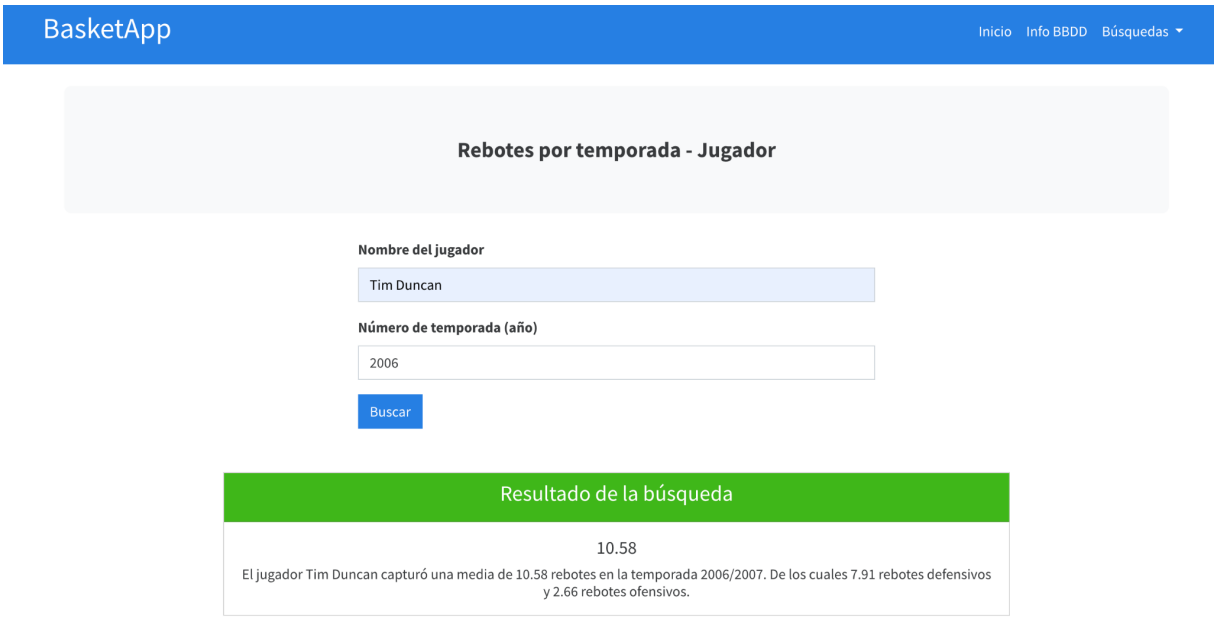

Ilustración 63. Vista de Búsqueda media de rebotes por temporada correcta.

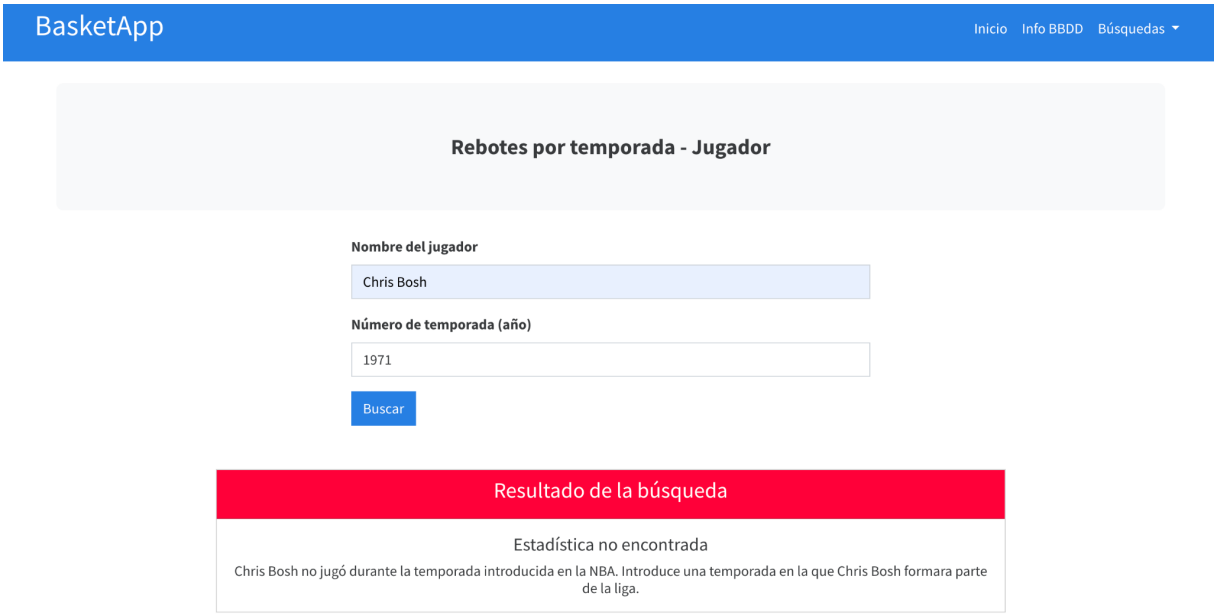

Ilustración 64. Vista de Búsqueda media de rebotes por temporada con temporada no jugada por el jugador.

- **Historia de usuario nº 5: saber las victorias de un equipo en una temporada.** Criterio de validación:
  - Devuelve el valor pedido cuando se introducen los parámetros correctos.
  - Devuelve mensaje de error en caso contrario.

De nuevo pruebas idénticas a las realizadas en las historias de usuario 2, 3, y 4. Ilustración 64 para búsqueda correcta e ilustración 65 para búsqueda errónea.

Ilustración 65. Vista de Búsqueda victorias equipo por temporada correcta.

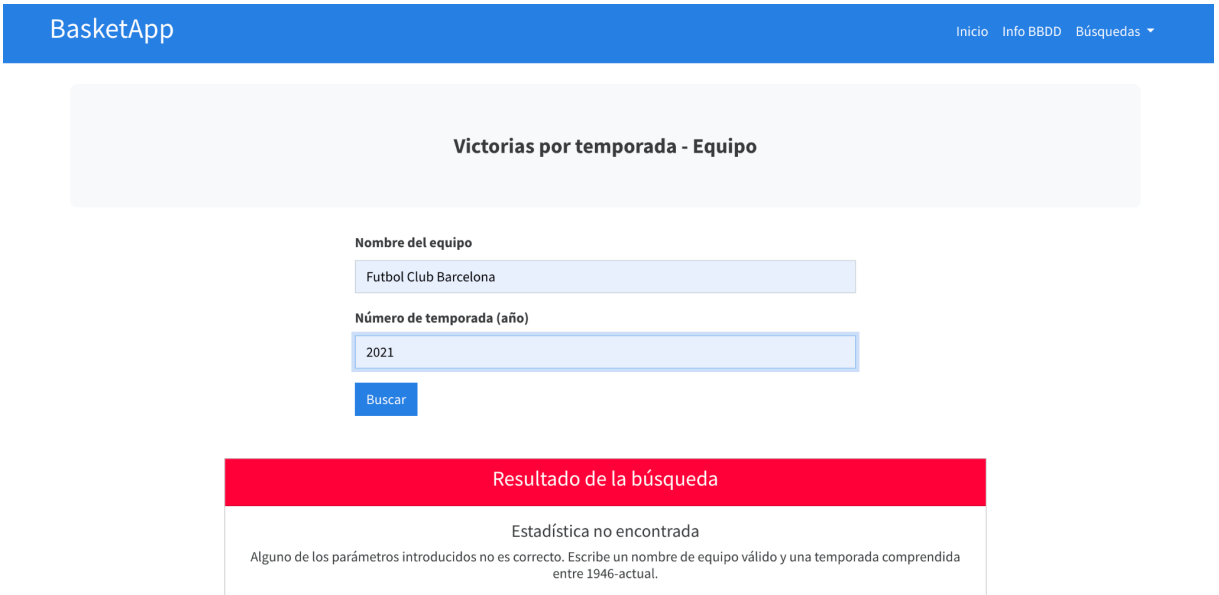

Ilustración 66. Vista de Búsqueda victorias equipo por temporada errónea por equipo incorrecto.

Una vez comprobados los criterios de validación de todas y cada una de las historias de usuario que habíamos, primero definido y posteriormente implementado, podemos dar por validado este segundo sprint de la aplicación.

4.6.2.5. Feedback

Para este último apartado de este segundo sprint, que como hemos visto en la documentación del primer sprint tiene una importancia capital en nuestra metodología de desarrollo ágil, he vuelto a construir un pequeño cuestionario que incluye las 3 preguntas que ya hicimos en el primer sprint, y una cuarta pregunta que nos dará una imagen visual del grado de buen hacer que está teniendo nuestra metodología de desarrollo. El cuestionario construido es el siguiente:

1. ¿Cuál es su opinión respecto a la aplicación en general? ¿La experiencia durante su uso fue agradable? ¿Considera que la navegación ha sido intuitiva y sencilla?
2. ¿Ha encontrado útiles las nuevas funcionalidades de búsqueda añadidas? ¿Considera que las respuestas recibidas son satisfactorias?
3. ¿Se le ocurre alguna funcionalidad adicional que le gustaría ver en la aplicación en siguientes versiones? ¿Algún aspecto en concreto de las funcionalidades ya existentes que considere que podría mejorarse y de este modo conseguir una experiencia de uso más satisfactoria?
4. ¿Considera que las mejoras, cambios y nuevas funcionalidades introducidas en esta segunda versión de la aplicación han aportado utilidad?

El número de usuarios que han probado este segundo sprint y han rellenado nuestro pequeño cuestionario han sido 8. Las ilustraciones 67, 68 y 69 muestran los resultados obtenidos para las preguntas 1, 2 y 4 respectivamente.

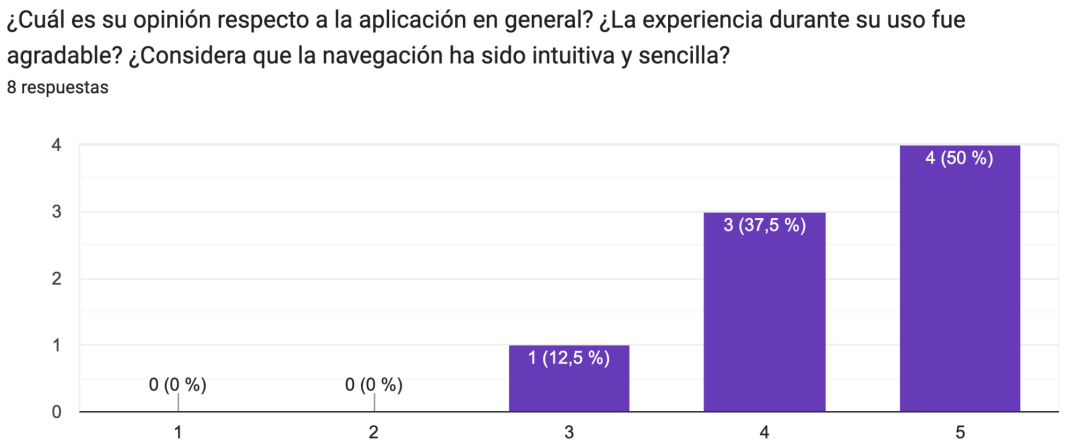

Ilustración 67. Gráfico de barras respuestas de la primera pregunta del cuestionario en el segundo sprint.

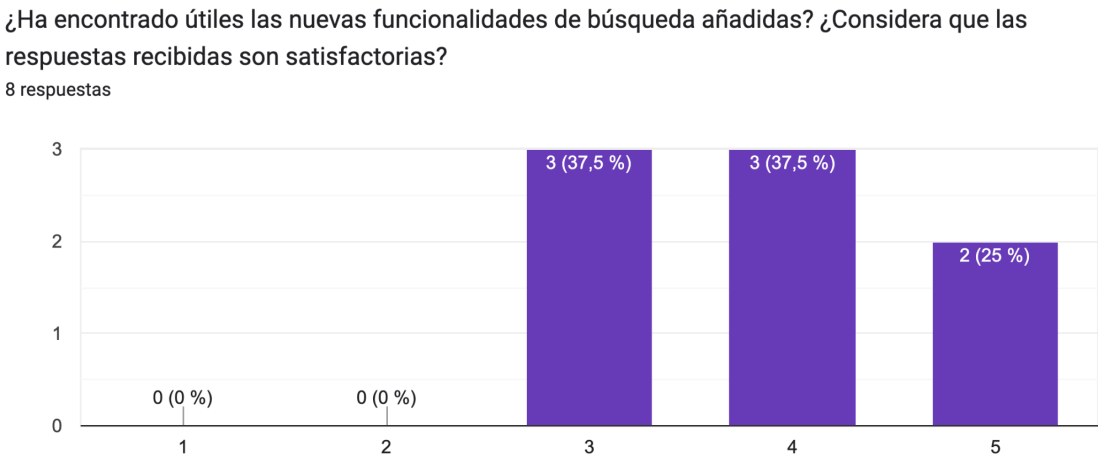

Ilustración 68. Gráfico de barras respuestas de la segunda pregunta del cuestionario en el segundo sprint.

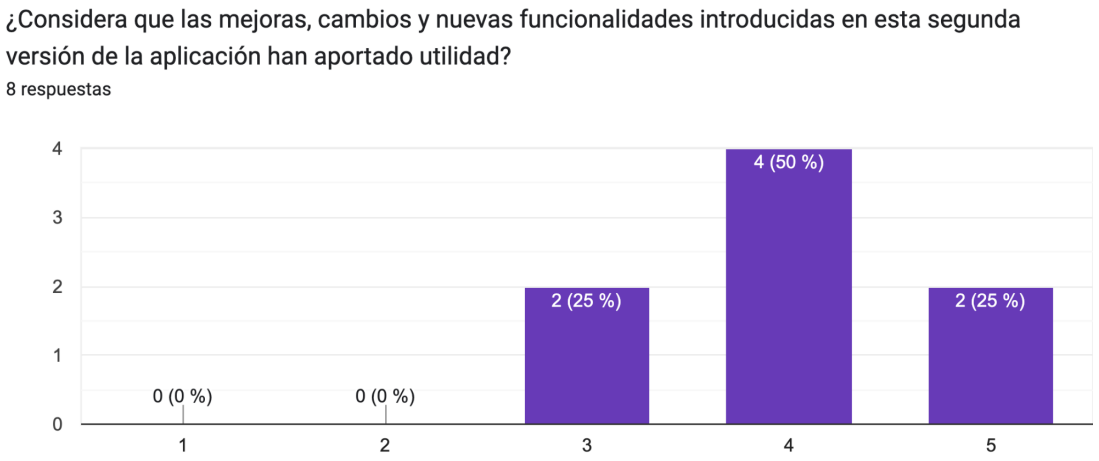

Ilustración 69. Gráfico de barras respuestas de la cuarta pregunta del cuestionario en el segundo sprint.

Estos gráficos demuestran que la acogida de la segunda versión de nuestra aplicación, en términos generales, ha tenido una buena acogida y ha gustado a los usuarios. Si bien como ocurría en el primer sprint, de nuevo la respuesta más interesante para obtener valor de la retroalimentación dada por los usuarios se encuentra en la tercera pregunta.

Las respuestas dadas han sido las siguientes:

- *Que incluya modo claro/oscuro.*
- *¿Podría incluirse una vista con gráficos estadísticos comparativos entre jugadores? Sería interesante que incluyeran una opción para elegir a los jugadores que quieres comparar.*
- *Añadir registro en la aplicación para poder tener alguna vista más personalizada, que incluya por ejemplo los últimos resultados de las búsquedas que he realizado*

- *Molaría que cuando búsquedas estadísticas de un jugador o equipo y sea correcta, aparezca aparte de la estadística pedida una imagen del jugador o equipo.*
- *Sería interesante añadir algo similar a un asistente virtual que pudiera ayudar al usuario con sugerencias. Yo por ejemplo no sé nada de baloncesto y nba y me ha costado hacer búsquedas.*
- *Me gusta como se han enfocado los distintos tipos de búsquedas disponibles, y me parecería muy interesante incluir en la siguiente versión que no solo muestre la estadística individual de un jugador, sino que también exista la posibilidad de establecer comparaciones con otros jugadores y equipos. Respecto al diseño de la web, también me parecería interesante incluir en siguientes versiones la opción de elegir la paleta de colores con la que veremos la web.*
- *incluir una vista con información sobre la infraestructura sobre la que se está desplegando la aplicación, la vista con la bbdd me ha parecido interesante.*
- *Se podría añadir un motor de búsqueda algo más complejo y sofisticado, con búsquedas más complejas y con más filtros.*

De los resultados de las primeras preguntas y de las respuestas dadas a esa pregunta relativa a ideas sobre nuevas funcionalidades, extraemos que la aplicación está gustando y resulta útil a la mayoría de usuarios que la han probado. Algunas de las nuevas funcionalidades propuestas por los usuarios resaltan interesantes, y e nuevo, de igual modo que nos ocurrió en el primer sprint de desarrollo del software, podrían ser incluidas en nuestro backlog de tareas como nuevas historias de usuario, y podrían llegar a ser implementadas en un tercer o posterior sprint de la aplicación.

El siguiente paso por tanto sería redefinir nuestras historias de usuario para incluir estas nuevas funcionalidades.

5. CONCLUSIONES

Con la perspectiva de la realización de este proyecto de fin de grado podemos comenzar haciendo un pequeño resumen de lo estudiado durante el documento, así como de los aspectos más relevantes del trabajo práctico realizado para la consecución de los objetivos que nos habíamos marcado inicialmente.

Para comenzar hemos estudiado tanto la tecnología Cloud Computing como las características más relevantes que aporta y los principales paradigmas que han surgido desde sus comienzos hasta la actualidad. Considero que se ha obtenido una imagen mental muy clara de los fundamentos teóricos de la tecnología, lo que nos ha permitido tener un amplio conocimiento de las aportaciones que ya ha tenido y de las enormes posibilidades que aún subyacen bajo los diferentes paradigmas.

El segundo gran objetivo que nos propusimos en un principio fue diseñar, desarrollar, implementar y desplegar una pequeña aplicación que utilizase las amplias capacidades y algunos de los servicios proporcionados por algún proveedor de servicio bajo el paradigma Platform as a Service. Este desarrollo nos ha permitido, no solo conocer, sino también comprobar y validar las ventajas del enfoque que habíamos estudiado en la parte teórica del proyecto. Algunas de ellas han sido: olvidarnos de la gestión y administración de la infraestructura, alta disponibilidad y escalabilidad automática.

Este proyecto ha supuesto un gran desafío y una fantástica oportunidad para aplicar todos aquellos conocimientos que he ido adquiriendo durante mi etapa universitaria, y considero que ha sido un broche de oro perfecto para cerrar una etapa y comenzar a vislumbrar mi camino. Ha implicado mejorar mi capacidad de investigación, desarrollar mis habilidades para diseñar, implementar y desplegar una aplicación, así como para valorar y evaluar su funcionamiento. Todo ello, como no podía ser de otra manera, me hace terminar este escrito con un gran sabor de boca y con la convicción y serenidad que aporta el trabajo bien hecho. Listar aquí aquellas

asignaturas que me han ayudado a realizar este proyector es una tarea complicada. Han sido unos años de mi vida apasionantes e inolvidables, y mis propios conocimientos y capacidades se han enriquecido hasta puntos que, cuando comencé esta andadura, no podía ni imaginar.

Asignaturas como Fundamentos de arquitectura de computadores o Fundamentos de programación en mi primer año universitario, Estructuras de datos y Diseño de algoritmos en segundo, Interacción persona-ordenador y Tecnologías basadas en la web en mi tercer año académico, y Gestión de proyectos software en mi último año de carrera, han sido asignaturas que me han ayudado mucho a la realización no solo de este proyecto de fin de grado, sino también a mi inclusión en un mundo laboral y empresarial del que formo parte desde hace ya algo más de dos años. Por todo ello solo puedo decir gracias. A pesar de los sinsabores propios de cualquier etapa de la vida, más si cabe cuando accedes a un mundo como el de la programación y la informática, considero que estos años han sido fundamentales en mi vida y mi desarrollo humano, y visto con la perspectiva del tiempo, estoy convencido que estaré eternamente agradecido a la Universidad de Jaén, la Escuela Politécnica Superior de Jaén y todos los profesores que me han acompañado en esta etapa eternamente.

Desde mi perspectiva y enfocándonos ahora en el tema principal de este trabajo de fin de grado, Cloud Computing en su paradigma Platform as a Service, utilizar esta tecnología y más concretamente este paradigma ha demostrado ser beneficioso por varias razones. En primer lugar, he comprobado que se reducen significativamente el tiempo y los recursos necesarios para implementar y desplegar la aplicación en comparación con los enfoques tradicionales. No es un tema baladí, pues en el desarrollo de un software estos aspectos lucen fundamentales y los beneficios que aporta la nube me han resultado fascinantes.Poder centrarse en el desarrollo de la aplicación sin tener la necesidad de preocuparse por la infraestructura aporta una ventaja sustancial que he aprendido a valorar mucho durante el desarrollo de este proyecto.

No obstante, parece también relevante hablar sobre los problemas que me he encontrado durante su uso. Dependiendo del proveedor utilizado, nos podemos encontrar con algunas limitaciones de personalización y flexibilidad de la infraestructura subyacente que parece importante remarcar. Además del propio marco teórico de la tecnología se desprende que la dependencia de terceros que se genera podría llegar a plantear problemas de seguridad y privacidad para algunos tipos de proyectos y aplicaciones.

En resumen, este trabajo de fin de grado me ha supuesto una valiosa experiencia permitiendo un profundo aprendizaje del marco teórico del paradigma, y podría llegar a decir que incluso sumergirme en el fascinante mundo del Cloud Computing y del paradigma Platform as a Service. Considero que mi comprensión de esta novedosa y emergente tecnología ha crecido mucho durante la realización de este escrito y sienta unas bases muy interesantes para futuras investigaciones y proyectos en el campo de la computación en la nube.

Centrándonos en la aplicación que he desarrollado, si bien se encuentra en una etapa temprana de desarrollo, hemos visto que tiene grandes posibilidades, y teniendo en cuenta la retroalimentación dada por los usuarios tras el segundo incremento, y el backlog de funcionalidades que ya teníamos y por una cuestión de tiempo no pudimos implementar, considero que continuar mi trabajo podría ser una opción muy a tener en cuenta para futuros estudiantes que transiten mi mismo camino.

Bibliografía

Amazon Web Service, s.f. AWS (s.f.) ¿Qué es AWS?. https://aws.amazon.com/es/what-is-aws/

Amazon, 2023 Amazon (2023) https://www.amazon.es/Nuevo-Apple-MacBook-pulgadas-almacenamiento/dp/B081G9YQ73

Condori, X. G., 2013 Condori, X. G. (2013). Community cloud computing. Revista de Información Teconología Y Sociedad, 70-72.

Costello, K., 2021 Costello, K., (2021). Gartner predice el futuro de la computación en la nube. https://www.gartner.es/es/articulos/gartner-predice-el-futuro-de-las-infraestructuras-de-cloud-y-edge-computing

Cruz Valencia, K., 2012 Cruz Valencia, K. (2012). Historia del cloud computing. Revista de Información, Tecnología y Sociedad, 51.

Dawoud, W., 2010 Dawoud, W., Takouna, I., & Meinel, C. (2010, March). Infrastructure as a service security: Challenges and solutions. In 2010 the 7th International Conference on Informatics and Systems (INFOS) (pp. 1-8). IEEE.

Daylami, N., 2015 Daylami, N. (2015). The origin and construct of cloud computing. International Journal of the Academic Business World, 9(2), 39-45.

Díez Álvaro, J., 2012 Díez Álvaro, J. (2012). Los entornos en nube (Cloud Computing): modalidades, sistemas Cloud en la actualidad, normativa aplicable, controles a considerar y guía de implantación (Bachelor 's thesis).

Finn, A., 2012 Finn, A., Vredevoort, H., Lownds, P., & Flynn, D. (2012). Microsoft private cloud computing. John Wiley & Sons.

Fox, A., 2009 Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., ... & Stoica, I. (2009). Above the clouds: A berkeley view of cloud computing. Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS, 28(13), 2009.

Google, 2021 Google. (1 de enero de 2021). ¿Qué es cloud computing? https://cloud.google.com/what-is-cloud-computing?hl=es

Google, s.f. Google (s.f.). Descripción general de Google Cloud. https://cloud.google.com/about/locations?hl=es

Hernández, N. L., 2014 Hernández, N. L., & Florez-Fuentes, A. S. (2014). Computación en la nube. Mundo Fesc, 4(8), 46-51.

Hinojosa, M. A., 2003 Hinojosa, M. A. (2003). Diagrama de gantt. Producción, procesos y operaciones, 48.

Jobted, 2023 Jobted, (2023). Salarios en España. https://www.jobted.es/salario

Lawton, G., 2008 Lawton, G. (2008). Developing software online with platform-as-a-service technology. Computer, 41(6), 13-15.

Licklider, J. C. R., 1962 Licklider, J. C. R., & Clark, W. E. (1962, May). On-line man-computer communication. In Proceedings of the May 1-3, 1962, spring joint computer conference (pp. 113-128).

Ma, D., 2007 Ma, D. (2007, July). The business model of" software-as-a-service". In Ieee international conference on services computing (scc 2007) (pp. 701-702). IEEE.

Marian, M., 2012 Marian, M. (2012). iPaaS: Different ways of thinking. Procedia Economics and Finance, 3, 1093-1098.

MarkoInsights, K. M., 2019 MarkoInsights, K. M. (2019). Una introducción a Alibaba Cloud para empresas occidentales. https://www.computerweekly.com/es/consejo/Una-introduccion-a-Alibaba-Cloud-para-empresas-occidentales

Mell, P., 2011 Mell, P., & Grance, T. (2011). The NIST definition of cloud computing.

Microsoft, s.f. Microsoft (s.f.) ¿Cómo funciona Azure? https://learn.microsoft.com/es-es/azure/cloud-adoption-framework/get-started/what-is-azure

Murazzo, M. A., 2010 — Murazzo, M. A., Millán, I. F., Rodríguez, N. R., Segura, D., & Villafañe, D. A. (2010). Desarrollo de aplicaciones para Cloud Computing. In XVI Congreso Argentino de Ciencias de la Computación.

Mustafa, O., 2023 — Mustafa, O. (2023). Overview of Amazon Web Services. In A Complete Guide to DevOps with AWS: Deploy, Build, and Scale Services with AWS Tools and Techniques (pp. 1-35). Berkeley, CA: Apress.

Neto, P., 2011 — Neto, P. (2011). Demystifying cloud computing. In Proceeding of doctoral symposium on informatics engineering (Vol. 24, pp. 16-21).

Nigro, H., 2022 — Nigro, H. (2022). Cloud computing: retos y oportunidades. Revista Ingeniería, Matemáticas y Ciencias de la Información, 9(18), 11-16.

Pérez, C. F. V., 2017 — Pérez, C. F. V., Cleves, J. E. P., & Pallares, L. (2017). Computación en la nube: Un nuevo paradigma en las tecnologías de la información y la comunicación. Redes de Ingeniería, 138-146.

Platzi, 2018 — Platzi (2018). ¿Qué es IBM Cloud y cómo funciona? https://platzi.com/blog/que-es-ibm-cloud-y-como-funciona/

Rajan, A. P., 2020 — Rajan, A. P. (2020). A review on serverless architectures-function as a service (FaaS) in cloud computing. TELKOMNIKA (Telecommunication Computing Electronics and Control), 18(1), 530-537.

Rosado, T., 2014 — Rosado, T., & Bernardino, J. (2014, July). An overview of openstack architecture. In Proceedings of the 18th International Database Engineering & Applications Symposium (pp. 366-367).

Sainz, C. C., 2014 — Sainz, C. C., Alonso, J., Tuesta, D., & de Lis, S. F. (2014). El desarrollo de la industria del cloud computing: impactos y transformaciones en marcha.

Sefraoui, O., 2012 — Sefraoui, O., Aissaoui, M., & Eleuldj, M. (2012). OpenStack: toward an open-source solution for cloud computing. International Journal of Computer Applications, 55(3), 38-42.

Stackscale S.L., 2021 — Stackscale S.L. (2021). Tipos de cloud computing. https://www.stackscale.com/es/blog/tipos-cloud-privado-publico-hibrido/

VMware, s.f. VMware (s.f.). ¿Qué es la nube híbrida? https://www.vmware.com/es/topics/glossary/content/hybrid-cloud.html

Xie, X., 2018 Xie, X., Yuan, T., Zhou, X., & Cheng, X. (2018). Research on trust model in container-based cloud service. Computers, Materials and Continua, 56(2), 273-283.