UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

TECNOLOGÍAS DE LA INFORMACIÓN

TRABAJO FIN DE GRADO

DocPath CloudPlatform

David García-Moreno Gómez-Caraballo

Diciembre, 2017

DOCPATH CLOUDPLATFORM

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

DEPARTAMENTO DE TECNOLOGÍAS Y SISTEMAS DE INFORMACIÓN

TECNOLOGÍAS DE LA INFORMACIÓN

TRABAJO FIN DE GRADO

DocPath CloudPlatform

Autor: David García-Moreno Gómez-Caraballo

Director: Manuel Ángel Serrano Martín

Director: Arturo Peralta Martín-Palomino

Diciembre, 2017

David-García-Moreno Gómez-Caraballo

Daimiel – Ciudad Real – Spain

E-mail: david.garcia@alu.uclm.es

TRIBUNAL:

Presidente:

Vocal:

Secretario:

FECHA DE DEFENSA:

CALIFICACIÓN:

PRESIDENTE	**VOCAL**	**SECRETARIO**
Fdo.:	Fdo.:	Fdo.:

A mis padres, por poder darme la oportunidad de formarme en esta profesión. Gracias.

Resumen

Actualmente la gestión documental es un ámbito de la informática que está en auge y la empresa DocPath es una de las empresas que trabajan sobre ello, realizando productos para hacer más eficiente la generación y clasificación de los documentos críticos de las empresas

En la actualidad todos sus productos son productos que se tienen que instalar en el ordenador del cliente y hacen uso de los recursos "locales" del mismo. Por eso nació esta oportunidad. Se realiza una plataforma, a partir de la cual, se ofertan los mismos productos que actualmente tiene la empresa en el mercado, pero hacerlo de forma "cloud", es decir, en vez de instalar estos productos en un hardware propio, instalarlos en un cloud remoto y obtener un rendimiento mayor del producto, despreocupándose de la configuración, actualización y mantenimiento del producto.

En un primer momento se tiene que diseñar y realizar una plataforma que trabaje con un servidor cloud para que nos proporcione el servicio de almacenamiento en máquinas virtuales, para poder instalar y trabajar sobre ellas.

La razón fundamental además del ahorro en hardware que tendría el cliente sería poder permitir tener un control sobre el rendimiento de los productos y poder facturar a los clientes por el rendimiento real o uso que le dan al producto. Además, se podrá auto-configurar los productos en función de la necesidad para adaptarse a esos picos de trabajo.

Abstract

Currently document management is a field of information technology that is increasing and the company, DocPath is one of the companies that working on that, making products to make more efficient all the generation and classification of critical business documents.

Now, all its products are products that have to be installed on the client's computer and make use resources of it. That's why this opportunity was born.

The objective is to create a platform, which offered the same products that the company currently has on the market, but in the cloud. Instead of installing these products in their own hardware, install them in a remote cloud to get more performance of the product without having to update or maintain the product.

The first thing is to design and build a platform that works with a server in the cloud and provides the storage service in virtual machines.

The fundamental reason, besides the hardware savings, is to be able to control the performance of the products and be able to invoice by the real use. In addition, you can configure the products depending on the need to adapt to yield ups and downs.

Agradecimientos

Me gustaría agradecer en primer lugar a mis padres, Pedro Antonio y María de los Ángeles, que me han podido dar la oportunidad de formarme en esta profesión y por guiarme en momentos difíciles que he pasado durante estos años. A mi hermano Jesús, por su ofrecerme su apoyo. En general a toda mi familia, que siempre han estado ahí y en especial a aquellos que no están y no han podido ver la consecución de esta gran meta.

Agradecer el apoyo a todos mis amigos fuera de la escuela que me han hecho distraerme y desconectar en los momentos delicados y a los nuevos amigos que se cruzaron en mi camino en la escuela, desde los que no pudieron acabar esta etapa, hasta los que me han acompañado hasta el final de ella: Bacete, Dani, Jasón, Fer, Estefanía, Fran, Miguel… mención especial a los dos compañeros con los que he compartido la experiencia de realizar los proyectos en la empresa, Jose y Carlos, especialmente con este último, con el que he compartido la mayoría de mis trabajos y prácticas, un gran compañero de viaje que estoy seguro que coincidiremos en alguna etapa más de la vida.

Agradecer a todo el profesorado que me han enseñado en este tiempo, agradecer a Francisco Pascual por darme la oportunidad de poder cursar esta beca, a Jesús Galindo en nombre de DocPath por poder cursar el TFG en su empresa, de la que me siento muy agradecido de conocer y coincidir con los compañeros de oficina, y por último a Arturo y Manuel Ángel por guiarme en este TFG y por darme consejos sobre el mismo.

En general, muchas gracias a toda la gente que me ha tenido que aguantar, soportar y disfrutar este tiempo y perdón por los errores que haya podido cometer. Ha sido una etapa bonita de mi vida de la que me llevo a bastante gente. Gracias

David García

Índice general

Índice de figuras

Capítulo 1

Introducción

En este capítulo se introduce el tema a tratar en el Trabajo de fin de Grado. No es idea del mismo, ni tampoco una idea suministrada por su profesor o tutor, es un proyecto basado en la necesidad que tiene una empresa específica, y por ello, tiene que ser desarrollado en la misma, utilizando sus herramientas de trabajo.

Está encuadrado en el programa FORTE, cuyo objetivo es incentivar a los alumnos a hacer prácticas en empresas reales, beneficiándose de poder cursar el TFG en esa empresa, mientras realizas un trabajo de prácticas y te formas con conocimientos y capacidades técnicas de ese trabajo. Se comienza describiendo el tema principal del trabajo y se continúa con la situación de la empresa y sus beneficios, finalizando con el desarrollo del proyecto y sus posteriores trabajos o proyectos futuros.

1.1 Motivación

La gestión documental es el conjunto de técnicas y prácticas usadas para administrar el flujo de documentos de cualquier tipo de una organización. Es un término que se define para la época actual, pero que se practicaba en la antigüedad, desde la aparición de la escritura. Desde entonces, todas las civilizaciones se han comunicado a través de la escritura, quedando constancia en algunos manuscritos o documentos antiguos, que tenían que ser organizados y clasificados para su conservación y clasificación. Apareciendo términos importantes para la gestión documental como las bibliotecas en el siglo VII a.C. Posteriormente, aparecen nuevos términos e inventos importantes para la gestión documental y mejoras de los métodos archivísticos. Aparece la imprenta de la mano de Gutenberg y con ella surge la necesidad de centralizar todo el contenido, en grandes archiveros o bibliotecas con el fin de obtener grandes beneficios. Más adelante, los métodos o soportes físicos o en papel empiezan a quedarse estancados con la nueva tecnología emergente y aparecen los nuevos soportes magnéticos u ópticos ligados al concepto de computación y con el desarrollo de los primeros ordenadores se adquiere un nuevo avance a la hora de almacenar documentos. Las organizaciones que más tienen esa necesidad de organizar los documentos son las propias empresas, y necesitan localizar y centralizar todo el conocimiento y el negocio en un lugar determinado y conocido.

En conclusión, se podría decir que la información de una empresa es el activo más valioso de la misma y la manipulación y gestión de la misma es vital. Por eso de la importancia de la gestión documental en las empresas.

DocPath® es una empresa líder en el desarrollo de software de gestión documental, que facilita a las compañías, optimizar e implementar sus procesos avanzados, pero tiene todos sus productos enfocados a la gestión de esa documentación localmente. Esto significa que el cliente tiene que provisionarse de una infraestructura en la que crear, gestionar y almacenar toda la información relevante y valiosa para su organización. Nuestro proyecto no supondría nada nuevo, pero es la forma de adaptar todos los productos existentes al *cloud* y ofrecer otro nuevo tipo de solución para abrir un nuevo mercado de clientes para la empresa. Se proporcionaría una plataforma para albergar esos mismos productos en un *cloud* en vez de desplegarse en su propia infraestructura, *on-premise*.

Con él, se genera una plataforma que daría la posibilidad de gestionar los productos de la misma manera por los clientes, pero sin tener la propia máquina física en la que se instala o despliega el producto DocPath®, lo que permite ahorro en coste y, sobre todo, beneficios a la hora de actualizar los productos y sus versiones

1.2 Objetivo

El objetivo principal del TFG consiste en la creación de un modelo básico de la plataforma, sobre la cual una vez aprobada, desarrollar el resto de la plataforma. La plataforma tiene que ofertar los mismos productos que ahora están desarrollados para funcionar en un *hardware* local, pero hacerlo para un "*hardware cloud*". Debe permitir:

- Un rápido despliegue de los productos que los clientes quieren contratar.
- Facturar en función del uso del cliente.
- Auto-configurar y agrupar los productos de los clientes.
- Permitir cambiar la capacidad de generación de las máquinas.
- Mejorar la comunicación interna y colaboración en la empresa.

Muchos de estos objetivos engloban pequeños objetivos o metas técnicas del proyecto. Pero por motivos de diferente índole, no podemos asegurar que se cumplan, algunos de ellos dependen del despliegue, tipo de solución, tiempo e incluso coste, por lo que al ser un proyecto primerizo en una empresa y disponer de un sólo desarrollador y un tiempo determinado, no se asegura que el "prototipo" cumpla todos los objetivos. Esos objetivos serían de un proyecto de mayor envergadura y que podría ser la continuación del mismo.

Con esto anterior, no se quiere decir que no se tenga fijado el alcance del proyecto, si no que el proyecto podría tener un alcance aún mayor. Todo esto se detalla en la parte de objetivos y en la especificación de requerimientos, estudiado en la parte de *Inception*.

1.3 Metodología

La metodología que se utiliza es un desarrollo ágil, ya que nuestro fin prioritario es tener una entrega en un periodo de tiempo reducido de un proyecto funcional y que pueda ser evaluado y validado. Este tipo de metodología es muy afín a proyectos que necesitan ser flexibles y que se puedan corregir las tareas y el rumbo del mismo proyecto según el cliente, aunque siempre teniendo muy en cuenta los objetivos principales fijados con anterioridad.

No se puede elegir una metodología ágil en concreto, ya que estas métodos o modelos, están orientados para un equipo de trabajo como mínimo de tres personas, y en nuestro caso, nuestro equipo está formado por una sola persona. Aunque no se puede seguir exactamente una metodología en concreto, se hace una aproximación. Se utilizan prácticas de algunas metodologías ágiles, por ejemplo, de Scrum y de Programación Extrema. En su apartado concreto avanzaremos más en el tema.

1.4 Estructura de la obra

Capítulo 1. Introducción: Es la presentación del TFG. En este apartado se cuentan los aspectos importantes del proyecto escuetamente.

Capítulo 2. Objetivos: Fija los objetivos a satisfacer, concluido el proyecto.

Capítulo 3. Estado del arte: Se hace una exposición acerca de los antecedentes al producto, se describe el estado actual de la gestión documental, se habla sobre los principales competidores y todo lo relacionado con el tema del proyecto.

Capítulo 4. Metodo de trabajo: Se analiza todo lo que engloba el proyecto, desde los propios requisitos del mismo, la metodología seguida, medios hardware y software que han sido utilizados para la realización del proyecto, posibles soluciones y la solución a la que se pretende llegar.

Capítulo 5. Resultados: Muestra y explica las etapas que ha ido pasando el proyecto hasta llegar a la solución final. Desde la etapa de Inception o etapa de estudio previo, pasando por todas las soluciones intermedias, hasta llegar a la solución final.

Capítulo 6. Conclusiones: Se enuncian las conclusiones alcanzadas tras haber finalizado el proyecto y se proponen posibles actividades o mejoras a realizar en un futuro.

Capítulo 7. Bibliografía: Detalla las referencias utilizadas para documentar el proyecto.

Capítulo 8. Anexos: Muestra los datos añadidos al proyecto, que por extensión se han encuadrado en un informe aparte.

Capítulo 9. Acrónimos: Definición de siglas utilizadas en el proyecto.

Capítulo 2

Objetivos del TFG

En este capítulo se relata el objetivo principal que se pretende resolver, así como una serie de objetivos más específicos que se han propuesto para lograr alcanzar el objetivo principal.

2.1 Objetivo principal

El objetivo principal es la creación de una plataforma que será la base del producto y que, el cliente pueda contratar un producto de la empresa a través de ella, pero en vez de desplegarlo en su máquina local, se ofrezca otra máquina remota donde hacerlo.

Así, nuestra plataforma "*DocPath CloudPlatform*" tiene que ser capaz de contratar un servicio a otras plataformas cloud y en esa ubicación instalar el producto, para que el cliente, sin necesidad de saber la tecnología que se está utilizando, pueda conectarse y usar su producto contratado del mismo modo que si estuviera instalado en su propio hardware.

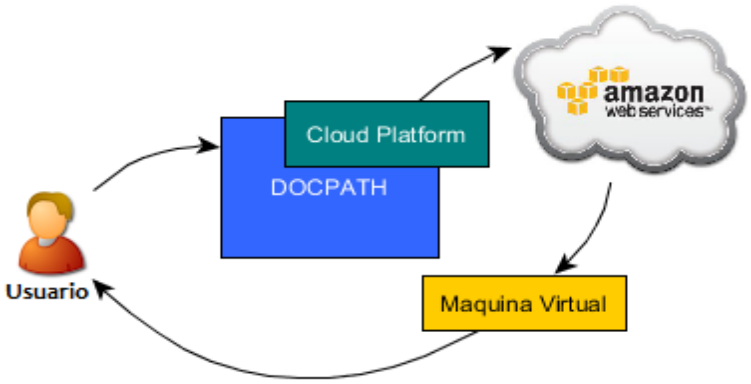

Figura 1: Alcance marcado como objetivo

En la imagen anterior se ofrece una vista más esquemática del alcance y solución a la que se quiere llegar. La plataforma *DocPath CloudPlatform*, actúa de intermediario entre la petición del cliente que quiere un producto en concreto y el servidor cloud que nos permite desplegar el producto en la nube, para que posteriormente el propio cliente use el producto.

2.2 Objetivos específicos

El objetivo del TFG consiste en la creación de una plataforma que será la base del producto. En ella, se tienen que permitir unos objetivos específicos descritos a continuación:

Objetivo 1**: Rápido despliegue**.
El sistema contrata un espacio *cloud,* para facilitar la máquina *hardware* en la que el cliente desplegará el producto. Las soluciones actuales solo se almacenan localmente. Esto permite que el cliente se olvide de instalaciones y actualizaciones del *software*, el cliente solo tiene que elegir el producto y las características y el sistema por sí solo le instala todo lo necesario en un cloud lo suficientemente grande para el producto que va a desplegar, todo ello sin que el cliente se preocupe de instalarlo, ni actualizarlo.
Objetivo 2: **Facturación en función de uso.**
Al controlar el tiempo que la plataforma está funcionando y la generación de documentos realizada, es posible facturar en función de tiempo real de uso, alquiler de la plataforma y acciones o rendimiento real. Esta funcionalidad se llevará a cabo a través de una base de datos que controle el uso del cliente. En ella se guardarán todos los datos del cliente y las diferentes acciones y espacios que el cliente manipula y controla. Es útil para clientes con poco uso o con un uso muy escalonado y les supone a la vez, un gran ahorro para las empresas, sobre todo, empresarios noveles que hacen un gasto en una infraestructura inicial y contratan sólo la potencia que se necesita en este momento.
Objetivo 3**: Permitir cambiar la capacidad y potencia de generación de las máquinas.**
Tener la posibilidad de cambiar en cada momento las capacidades de generación de las máquinas y por consiguiente de los productos. Está ligado al objetivo anterior, ya que es muy aconsejable tener en cuenta, además de poder parar las máquinas y no facturar por ese periodo de tiempo, el momento concreto que por cualquier motivo una organización aumente o disminuya su producción documental.
Objetivo 4**: Autoconfiguración**
Esta funcionalidad u objetivo, trata de dar la posibilidad de poder gestionar la configuración de dos o más productos de manera automática. Es decir, si un usuario tiene varios productos contratados, poder manipular las características conjuntamente para ambos. Este requisito hace posible el cambio a nivel general y no producto a producto.
Objetivo 5**: Apoyo a la generación**:
Un cliente podría tener instalado un producto *on-premise* (localmente), y para generaciones "puntuales", si un cliente necesita potencia extra o una generación masiva y puntual, podría contratar un servicio o producto *cloud*, sin necesitar costosas arquitecturas hardware al proporcionar nosotros el mismo. Este objetivo trata de apoyar al cliente que utiliza el producto a hacerlo de esta manera y suponerle un ahorro de infraestructura hardware que de hacerlo de manera *on-premise*, el cliente tiene que asumir y adquirir.
Objetivo 6**: DocPath® no necesita tener dicha infraestructura:**
Los servicios como Microsoft Azure y AWS permiten contratar dicha infraestructura de modo IaaS, de modo dinámico, permitiendo contratar mucha cantidad en un momento dado, eliminar la suscripción y permite facturar por ellos en función del uso de recursos.

De modo IaaS, significa "*Infrastructure as a service*", que consiste en contratar una infraestructura como un servicio, que es justamente lo que queremos. Hay otras alternativas muy relacionadas como PaaS y SaaS, que significan: *Platform y Software as a service* respectivamente, que tienen un nivel de abstracción distinto. Posteriormente, DocPath® es informada del coste de desplegar el sistema ahí y traslada estos costes a nuestros clientes.

Objetivo 7**: Mejora la colaboración y comunicación de la empresa**:

Ayuda a departamentos de ventas. Esta aplicación generaría entornos DocPath® en minutos que la empresa podría utilizarlos para cualquier venta o demo, sin necesitar máquinas virtuales u otras aplicaciones, otorgando vistosidad al cliente, dejando acceder a las máquinas sin miedo al uso de modo "indefinido", ya que tenemos el control de ellas.

Además, se podría hacer *testing,* sin necesidad de tener físicamente el *hardware.* Se crearía cualquier despliegue para visualizar y probar de manera automática el resultado de la nueva implementación y así poder visualizar y localizar los primeros fallos. También ofrece la posibilidad de *feedback* y compartir opinión, entre los propios departamentos de empresa. Podría haber realimentación entre departamentos.

2.3 Propuesta

La propuesta es simple, creación de productos DocPath® más baratos y eficientes que los actuales. En pocas palabras, lo que toda empresa busca, obtener el mismo beneficio con menor coste, para ello el único ámbito donde se puede recortar costes es en la infraestructura.

Hasta ahora DocPath dispone tan sólo de un tipo de producto, un producto on-premise, todo lo que se genera, se hace en la propia máquina. Normalmente, a la hora de obtener cualquier servicio o programa se descarga un ejecutable que se ejecuta e instala en la máquina y después, se trabaja con él. Todo lo anterior, supone un coste de infraestructura y equipos informáticos muy alto para el cliente y a veces no rentable.

La solución propuesta es hacer ese mismo servicio, pero en vez de utilizar tu propia máquina para instalarlo e utilizarlo, contratar un cloud donde tener una máquina en la que se instale y se utilice desde allí, ahorrando todos los costes de infraestructura.

¿Cómo lo podríamos hacer?:

Tendríamos un esquema muy parecido al que tenemos actualmente en la empresa, en concreto el producto sería el mismo, pero el usuario no tendría que disponer de una infraestructura local específica para desplegarlo. De esta manera, crearemos una plataforma que se comunique con cualquiera de las tecnologías cloud (*AWS, Azure, Google Cloud...*).

Se contrata un espacio *cloud* a esas empresas y a su vez en ese espacio, se colocaría una máquina que gestione el producto, tal y como se observa en la imagen:

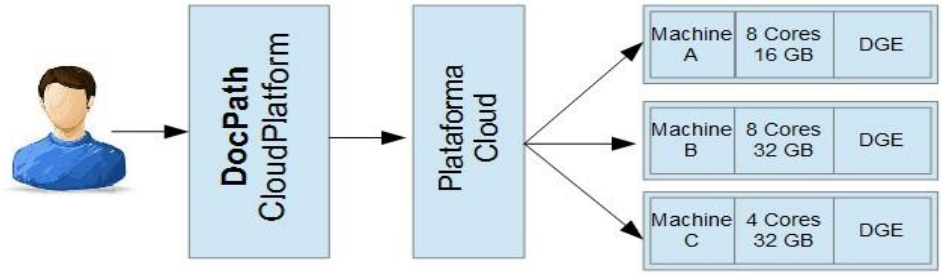

Figura 2:DocPath CloudPlatform resumen

El cliente puede elegir la cantidad de almacenamiento y cómputo quiere de su máquina y elegir cualquier otro tipo de configuración de la misma, así como el producto de DocPath® que se quiera desplegar. La tecnología proporciona el espacio suficiente y la empresa gestiona ese *cloud* y se lo facilita al cliente ya configurado previamente, para que utilice el producto. En otras palabras, nuestra plataforma sería una plataforma que gestione un servicio IaaS, "*infrastructure as a service*" y se comporte de intermediario entre el cliente y la propia máquina que gestiona y proporciona el servicio. Todos los movimientos sobre contrataciones, clientes y servicios quedan almacenados en una base de datos con el fin de facturar tanto a los clientes como a la propia plataforma que nos suministra el cloud.

¿Qué beneficios podrían proporcionarnos nuestros productos en un cloud? y ¿Qué beneficios tendrían para el cliente?

Muchas de las respuestas a las cuestiones sobre los beneficios que podría tener son los impedimentos que teníamos antes como, por ejemplo:

- Ofrecería un rápido despliegue. El cliente no tiene que preocuparse por montar su propia infraestructura, puede contratarlo y dejarlo funcionando en un tiempo ínfimo. Además de ahorrarse todo tipo de actualizaciones, dejando todo a cargo de la empresa.
- Un usuario podría tener muchos tipos de documentos auto-configurados creados y listos para utilizar, por los que el cliente está pagando. Antes, al tener un producto *on-premise*, la forma de pagar los productos sería por cantidad de productos y tiempo de contratación, pero ahora al ser una plataforma *cloud*, pueden ser controlados según su uso.

- Un cliente puede tener contratada una potencia concreta de *core*, memoria u otros aspectos, y en cualquier momento si se tiene la necesidad de aumentar o disminuir su producción, no habría problema alguno en poder cambiar la potencia.
- Configurar todos los productos de un cliente de manera modular. Formarían un módulo, que tendría una configuración común en aspectos que pueden crear conflicto.
- Si hay un cliente que tiene un producto *on-premise* de la empresa actualmente y requiere un servicio puntual del mismo, no tendría que contratar más potencia y comprar una infraestructura para utilizarla en ese momento, bastaría con ampliar esa contratación.
- No necesitará tener dicha infraestructura, que desarrolle y despliegue el producto, ya que estará contratada a otros servicios como Microsoft Azure o AWS que ofertan el servicio IaaS, permitiendo contratar mucho volumen en un momento dado.
- Además de los clientes, la empresa también se beneficiaría de ello. El departamento de ventas podría beneficiarse creando demos en entornos DocPath® de manera rápida, sin utilizar máquinas virtuales o físicas, para poder enseñar el producto al cliente. Se podría probar cualquier modificación en un entorno DocPath® y realizar feedback entre los propios miembros de la empresa y llegar a tener una última versión activa para pruebas...

2.4 Análisis de requisitos

A partir de la propuesta teórica de lo que se pretende realizar, se han formado una serie de requisitos categorizados en base al método MoSCoW. Esta peculiaridad se hace porque en este proyecto se sigue una metodología ágil y los requisitos no son fijos desde un principio. Se establece una escala de deseabilidad, dividiendo los requisitos en categorías: *Must, Should, Could* y *Would o Won't*, que indican objetivos que tiene, debería, es deseable y podría tener el producto, pero no tendrá.

Para el alcance de este TFG, será necesario realizar los requisitos de la categoría *Must*, pero, los requisitos pueden variar según el *PO* considere, y se pueden priorizar otros:

MUST

1. **Registrar todas las acciones.** Dado que será necesario contratar/facturar los servicios proporcionados a un proveedor de servicios ajeno a DocPath®, es muy importante auditar todas las acciones realizadas y registrarlas correctamente en una base de datos, así como calcular su precio para DocPath® y para facturar al cliente.

La base de datos tendrá los siguientes campos: un tipo de acción, hora y fecha, un usuario o cliente y producto afectado y una descripción breve de la tarea.

2. **Analizar todos los elementos facturables.** Cuando se crea una máquina, es necesario ver qué cantidades van a ser cargados a DocPath®, dependiendo también del producto a desplegar, ya que alguno no tendrá alguna de las siguientes:
    - Tiempo de funcionamiento de la máquina.
    - Tipo de máquina (sistema operativo, capacidad, memoria...)
    - Accesos a disco. Peticiones HTTP.IPs, tiempo. (cantidad de transacciones)
    - Tipo de contratación, es decir, si es por tiempo de uso o por franjas de tiempo.
    - Cantidad de almacenamiento de archivos.
3. **Automatización de la contratación de la máquina.** Es necesario que cuando un cliente contrate un servicio, se ejecute un proceso que auto-contrate la máquina en Azure o AWS y registre ese proceso. Es importante que sea automatizado, ya que supondría una reducción de tiempo, que es un requerimiento del cliente.

   Puede que esta funcionalidad sea más prioritaria, ya que, si no se tiene servicio disponible, no se puede registrar ningún proceso, ni facturar por ningún motivo.
4. **Crear *scripts* para crear / destruir la máquina.** Creación de los scripts o procesos necesarios para crear una nueva máquina con la configuración seleccionada y despliegue inicial del producto. Esto incluiría todo lo necesario para:
    - La creación de la máquina, asignación de recursos e instalación del producto
    - Licenciar a través de una licencia. Para que sea automático se tendría que implementar en "Magallanes", como con las versiones on-premise.
    - *Scripts* o procesos necesarios para destruir.
5. **Crear *Scripts* para arrancar/parar la máquina**. Creación de los *scripts* necesarios para iniciar la máquina y desplegar las aplicaciones. Para parar la máquina habría que hacer *scripts* necesarios para replegarlas correctamente y apagar la máquina.

SHOULD

6. **Crear WebApp.** Diseño y creación de la aplicación web, aunque sea una web funcional, que permita la configuración de los productos, visualización de contenidos y ejecución de acciones permitidas. Tener una interfaz visual para gestionar todo.
7. **Dar de baja de servicios.** El usuario podrá decidir dar de baja el servicio y asumirá la pérdida de datos, configuración y servicios guardados, advirtiendo antes de ello.

Aun así, se tiene que dejar "ocultos" los servicios durante un tiempo para poder reactivar el servicio y tener la posibilidad de volver al estado anterior.

8. **Arrancar/Parar servicios.** El primero debe iniciar la máquina, así como todos los productos que hay en su interior al iniciar dicha máquina. Mientras que parar, tiene que replegar todos los productos y acto seguido apagar la máquina.

9. **Mostrar lista de servicios contratados.** El usuario verá una lista de los servicios contratados. Al pinchar sobre uno, verá un reporte con la información, incluyendo:
    - Nombre o id, producto, sistema elegido (configuración de la máquina donde se despliega el producto) y estado actual.
    - Botones para iniciar/parar el proceso y botón para eliminar el servicio
    - Información para poder interaccionar con cada parte.
10. **Crear de pantalla de Configuración.** Cada producto tendrá una página especifica de configuración, donde cambiar aspectos técnicos, es decir: puertos, rutas de las aplicaciones.... Aún no se puede evaluar este punto.
11. **Crear *scripts* para configuración del producto automáticamente**. Crear un tipo de aplicación que a partir de una configuración en un formato "estandarizado" por DocPath®, que actualmente no existe, auto-configure nuestras aplicaciones. Como el punto anterior, aún no se puede evaluar este punto.

COULD

12. **Downgrade/Upgrade de la máquina.** Cuando un cliente tiene una máquina con una potencia, memoria y almacenamiento específico, es necesario permitir el cambio hacia una máquina con más o con menos recursos.
13. **Almacenamiento.** Los datos generados por las aplicaciones podrán ser almacenados en las propias máquinas virtuales. A la hora de realizar la contratación, deberemos preguntar al usuario si desea un almacenamiento adicional.

WOULD

14. ***Login* de usuarios.** Pantalla de *login* para usuarios del sistema. En principio los usuarios no podrán registrarse, ya que el proceso de dar de alta un usuario debe ser realizado desde alguna aplicación interna y de modo controlado.

15. **Mostrar lista de servicios disponibles y contratación.** Debe existir una página para contratar productos, dicha página debe mostrar una lista de los productos disponibles para ser contratados. En un principio se tiene pensado integrar en la página propia.

Los últimos 15 puntos serían los requisitos que tendría nuestro proyecto, pero el alcance del mismo estaría superado con los que están encasillados en la categoría Must. El proyecto, está dividido en *epic*, que en este caso podría ser equivalente a un requisito amplio, y a su vez se pueden dividir en *user story*, que serían pequeñas funcionalidades sencillas y concisas. Es una manera de administrar los requisitos de los usuarios sin tener que elaborar una gran cantidad de documentos formales.

Bill Wake definió un término o regla nemotécnica que define mediante unas siglas, el significado de las *user story*, cada inicial de la palabra INVEST indica una cualidad:

- ***I****ndependent*: Independiente. Todas tienen que ser independientes, si no lo son, seguro que no son *user story* y se pueden dividir aún más.
- ***N****egociable*: La historia no es lo suficientemente explícita como para esclarecer su alcance y éste debe dejarse explícito bajo la forma de pruebas y validación
- ***V****aluable*: La implementación tiene que coincidir con la petición del cliente.
- ***E****stimable*: El tiempo en realizarlas tiene que ser estimable para facilitar su planificación.
- ***S****mall.* Pequeñas. Si son muy grandes, son difíciles de estimar e imponen restricciones sobre la planificación de nuestro desarrollo iterativo.
- ***T****estable.* Verificables. Cuando se cubren requisitos, son generalmente verificables.

Además, los requisitos que se lleven a cabo tendrán que cumplir un criterio de calidad que estará definido mediante criterios de aceptación (*acceptance criteria*), que establece unas reglas o un "contrato", en el que se define, cuando se considera que un requisito está bien realizado.

Capítulo 3

Antecedentes

En esta sección se hace una exposición acerca de los antecedentes al producto, se describe el estado actual de la gestión documental y todo lo relacionado con el tema principal del proyecto, finalizando con una crítica al estado del arte actual para posteriormente poder mejora sobre esos aspectos.

Lo primero de todo será explicar todo lo relacionado con la empresa en la que se implementa el proyecto y después hablaremos de los dos temas relacionados con el proyecto, el cloud, que será el tipo de proyecto que se generará y la gestión documental que serán los tipos de productos que se quieren integrar en la solución.

3.1 Introducción al estado del arte

3.1.1 Introducción a la empresa

DocPath®[1] es una empresa líder en el desarrollo de software de gestión documental, que facilita a las compañías, optimizar e implementar sus procesos avanzados. Como objetivo prioritario, está enfocada a crear y consolidar las relaciones con sus clientes mediante la optimización de los procesos de *Customer Communications Management (*CCM*)* y a la gestión documental, desde la generación, hasta la distribución de documentos de comunicación (DOM).

- **CCM:** es una estrategia que se lleva a cabo mediante una combinación de software documental que tiene como objetivo la creación, almacenamiento, recuperación y distribución de las comunicaciones internas y salientes de una empresa, que tiene el fin de captar, fidelizar y retener a los clientes.
- **DOM:** es el proceso que cubre desde la generación del documento mediante diferentes entradas, hasta su distribución o almacenamiento, teniendo como fin optimizar la calidad, duración y costes asociados a todo el proceso.

[1] http://www.docpath.com/es/

DocPath® fue creada en 1992, en Madrid y cuenta con varias sedes como: Sao Paulo(Brasil) y Atlanta(USA), aunque su sede central se encuentra en Madrid, contando con centros de desarrollo en Ciudad Real y Miguelturra, lugar donde se desarrollará el proyecto.

Su función principal, es hacer posible que sus clientes puedan diseñar, generar, entregar y eliminar sus documentos críticos a sus clientes. Posiblemente, el gran éxito de DocPath® reside en que gran parte de sus ingresos, los destinan al área de I+D+i, aspecto que le hace ser más competitivos y darse a conocer entre sus potenciales clientes nuevos. Por eso tiene importantes clientes como bancos o aseguradoras de talla mundial.

Soluciones de DocPath® según el **tipo de empresa** y su utilización:

- **Industria, logística y distribución**: hacen soluciones que posibilitan la generación, impresión y distribución de la factura, albarán o sucedáneos.
- **Turismo**: la tendencia actual es la contratación por web, diferenciarse de la competencia es prioritario para que se puedan tener beneficios.
- **Telecomunicaciones**: ofrecer la posibilidad de tener apps capaces de imprimir y enviar documentos masivamente, facturar e innovar.
- **Sector sanitario**: para facilitar la gestión de, por ejemplo, la petición de una cita y garantizar la confidencialidad de ella.
- **Sector bancario**: ofertar procesos con la habilidad de diseñar, generar, gestionar y distribuir documentos en múltiples plataformas y entornos.
- **Aseguradoras**: disponer de software para garantizar una mejor relación con el cliente, como contratos, pólizas y documentos importantes.
- **Textil**: caso similar al turismo, para mejorar a la competencia.
- **Comercio**: herramientas para ofrecer un mejor servicio a los clientes.
- **Sector público**: ofrecer soluciones para cubrir todo el ciclo de vida de un documento, desde el diseño hasta la distribución impresa o electrónica

Soluciones según las **necesidades de negocio**:

- **Creación de documentos**: consiste en la fusión de datos en diferentes formatos con plantillas.
- **Distribución e-mail/fax**: sistemas de reintentos, consulta de estados, proceso integrado y automático.
- **Factura electrónica**: servicio público y servicio privado.
- **Almacenamiento y consulta**: distribución controlada SQL, Oracle...
- **Digitalización**: Indexación manual y por códigos de barras, interfaz configurable.
- **Impresión**: PCL, PostScript, AFP, etiquetas… producción masiva y bajo demanda, postproceso.
- **Generación de documentos web en tiempo real(PDF)**: acceso a la base de datos para recuperar información, Java, .NET, Webservices…
- **Integraciones con otros sistemas**: salida XML, APIs, Webservices…
- **Formularios electrónicos**: traducción automática de plantillas a webs.
- **Acounting**: control de impresión y asignación de cuotas.

3.1.2 Gestión documental. Evolución hasta el cloud

La gestión documental es el conjunto de técnicas y prácticas usadas para administrar el flujo de documentos de cualquier tipo de una organización, como es el diseño y creación de los propios documentos, la recuperación de los mismos y de la información que contienen, la determinación del tiempo que se deben conservar los que aún son de utilidad y eliminar los que ya no sirven o no tienen valor. Todo este proceso, supone un ahorro de coste y tiempo en la gestión y el control de los documentos críticos de cualquier organización.

Esto no es nada nuevo, todos estos procesos empezaron con la propia escritura. En la antigüedad, todos los tratos se hacían de palabra, se llegaba a un acuerdo y tenías que fiarte del vendedor o cliente con el que intercambiabas un producto hasta que, con la invención de la escritura, ya se podría dejar constancia del trato establecido, con un documento.

El gran cambio se produce en los años 90, con la aparición de la gestión documental de documentos electrónicos. En 2006 surge el nuevo paradigma tecnológico de la nube, que provoca un cambio generacional en la gestión documental, es cuando se produce la aparición del término: *Cloud Computing*, que permite ofrecer servicios de computación a través de una red, usualmente Internet.

Ahora mismo en la actualidad, cada vez son más las empresas que confían en la gestión de sus documentos a través de internet. Pero aún hay un rechazo por parte de la sociedad por temas de seguridad y en parte tienen razón, porque es el principal inconveniente. Las empresas prefieren acumular sus documentos críticos para que no puedan ser manipulables por nadie. Se han producido alguna vez errores de seguridad, pero hay numerosos avances en la actualidad, como para ser tan tradicionales en ese sentido. A continuación, se detallarán algunas soluciones a esos problemas tradicionales.

Por todo lo demás, la mayoría de las empresas que se dedican a la gestión de esos documentos tienen que evolucionar con el tiempo y ofrecer a esos clientes este tipo de servicios que les benefician más que perjudican ya que es cuestión de tiempo, que las personas confíen más en estos avances, al igual que pasó con por ejemplo los pagos con tarjetas de crédito. En este sentido la empresa pretende renovar e innovar en este tipo de soluciones por eso la creación de este trabajo. Con el paso del tiempo, la gestión documental está cada vez más ligada al cloud o gestión en la nube.

3.1.3 La nube. Cloud Computing

La nube se puede definir como un conjunto de hardware y software, almacenamiento, servicios e interfaces que facilitan la entrada de la información como un servicio. Actualmente, el rechazo a utilizar estos servicios que nos proporciona la nube es mucho menor, ya que se tienen muchas alternativas y muchos tipos de nubes, dependiendo de las necesidades de cada empresa, modelo de servicio ofrecido o la implementación de ella.

Figura 3: Definición gráfica del Cloud Computing

La computación en la nube se basa en la necesidad de compartir recursos como fin de conseguir economía de escala. Es decir, beneficiarse en términos de coste de la expansión de la empresa y no endeudarse en exceso pensando a largo plazo. Con lo anterior se quiere decir que la nube, permite a las empresas evitar o minimizar los costes asociados al despliegue de las infraestructuras, facilitando a las empresas en centrarse en su negocio y poder escalar el coste inicial.

Las empresas pueden escalar su potencia de cómputo según quieran, si necesitan más rendimiento pues aumentan la potencia y por el contrario la disminuyen. Esto no se puede realizar normalmente porque desde un principio tienes que comprar una infraestructura y no puedes amoldarte a esos picos de producción.

En general, el cloud computing, cada vez más está siendo más demandado debido a las grandes ventajas que les proporciona a las empresas tener un alto desempeño de computación a un bajo coste, con un gran rendimiento, escalable y seguro.

Con relación a la seguridad y al rechazo de alguna parte de la sociedad a estas tecnologías, hay que decir que hay muchas alternativas y tipos diferentes de nubes. La diferencia más significativa entre cada tipo es la privacidad, actualmente podemos distinguir cuatro tipos de nubes:

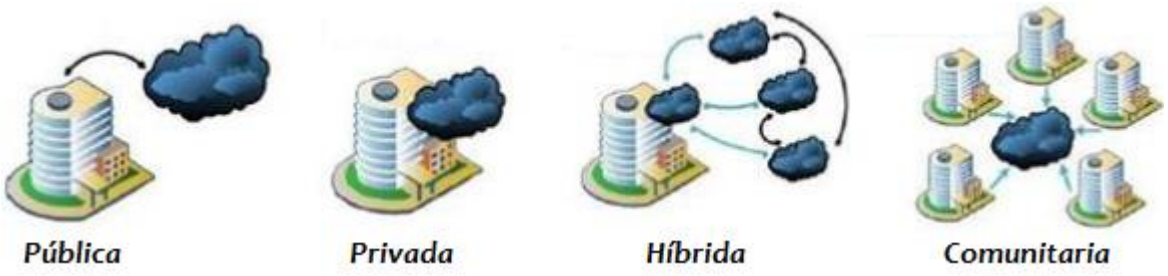

Figura 4: Tipos de nubes

- **Nubes privadas**: accesibles únicamente desde una determinada organización y gestionadas por la misma. Tienen una mayor seguridad y privacidad de los datos. Son una muy buena opción para las compañías que necesitan una alta protección de datos. La organización que la posee gestiona el servidor, red y disco y pueden decidir que usuarios están autorizados para utilizar la infraestructura.

- **Nubes públicas**: abiertas al público y propiedad del proveedor que las gestiona. Proporcionan un ahorro de coste y gran flexibilidad a cambio de menor seguridad. En este tipo de nubes, los datos y los procesos de los clientes se mezclan en los servidores, sistemas de almacenamiento y otras infraestructuras de la nube. Aunque los usuarios que las utilizan no saben dónde se almacenan todos los datos de los demás clientes. Son buena opción para manejar datos que no sean trascendentes, pero no se tiene ningún tipo de privacidad con este tipo de nube.
-
- **Nubes híbridas**: es una mezcla de las dos anteriores y es capaz de portar aplicaciones y datos como característica principal. Utilizan la parte pública para servicios genéricos y la privada para sus datos analíticos. Es decir, el usuario es propietario de algunas partes, pero a su vez comparte otras, aunque de una manera más controlada que en las nubes públicas. Está enfocada a aplicaciones simples que no requieran ninguna sincronización o que trabajen con bases de datos complejas. Un ejemplo puede ser los sistemas de correo electrónico de cualquier empresa.

- **Nubes de comunidad**: ofrecen una infraestructura compartida por varias organizaciones, pueden estar alojadas en las instalaciones de los usuarios y son gestionadas por las propias organizaciones.

Además de los cuatro tipos existe el "autoservicio bajo demanda", consiste en que un consumidor puede proveerse de tiempo y almacenamiento de la red, a medida que lo necesite, sin requerir una interacción entre el usuario y el proveedor dueño del servicio.

- Modelos de **despliegue**: se refieren a servicios referentes a la posición y administración de la infraestructura de la nube.
- Modelos de **servicios**: se refieren a los servicios específicos a los que se puede acceder en una plataforma de *cloud computing*. A su vez, los servicios pueden ser de tres tipos según el tipo de servicio que se ofrece:

	¿Qué ofrece?	Uso recomendado	Contenido en la nube
IaaS	Una infraestructura virtual alquilada por uso	Cargas de trabajo variable y gran cantidad de tareas en paralelo.	Sistema operativo o máquina virtual
PaaS	Una plataforma para ejecutar el código deseado a través de aplicaciones	Ejecución de aplicaciones simples que no requieren un control sobre la red	Código fuente de las aplicaciones o herramientas
SaaS	Software preparado para usarse a través de la web	Herramientas ofimáticas y base de datos de baja complejidad	Datos y procesos de negocio

- *Infrastructure as a service* (**IaaS**): Es el modelo de servicio en la nube por excelencia. Ofrece servicios a APIs de alto nivel que se utilizan para otros APIs de un propósito más específico. Los recursos informáticos compartidos más comunes suelen ser espacio en servidores virtuales, conexiones de red, ancho de banda, balanceadores de carga, backups…
- *Platform as a service* (**PaaS**): Ofrecen un entorno de desarrollo para los desarrolladores. Suelen desarrollar una serie de herramientas y estándares que permiten este tipo de desarrollo. En este modelo, la mayoría de las veces se provee de una plataforma con un sistema operativo instalado, un entorno con un lenguaje determinado, una base de datos como almacén de los mismos y un servidor web. Algunos ejemplos son Microsoft Azure y Google Engine.

- *Software as a service* (**SaaS**): Ofrecen acceso a un software y a un almacén de datos. Los proveedores gestionan la infraestructura y la plataforma en la que se ejecuta el software o la aplicación. Con este tipo de solución te olvidas de instalar y ejecutar las aplicaciones en tus propios equipos y tan sólo tienes que preocuparte de utilizar la herramienta proporcionada.

En resumen, después de todos los tipos de nubes y servicios que nos proporciona el cloud, es más que evidente que este tipo de soluciones se van a ir imponiendo cada vez más, ya que presentan numerosas ventajas y apenas poseen inconvenientes, sólo tienes que tener claro el tipo de nube que debes escoger y sus características básicas en función de la privacidad, accesibilidad, rendimiento y disponibilidad

3.1.4 Gestión documental en la empresa.

En DocPath®, si existe una solución que define a la empresa, es DocPath® Areca. Mediante esta solución, se pueden generar documentos a partir de plantillas diseñadas por los clientes en función de sus necesidades, junto con los datos proporcionados por ellos mismos, ya sea en XML, base de datos o archivos de texto. Hay muchos productos finales de la empresa que tienen esta parte integrada en ellos, como: *Business Suite Pro*. Los productos de la empresa son adaptaciones de este producto, según las necesidades que tenga el cliente y su forma de gestionar esos documentos, pero en todos está esta base.

Con esta solución, todos los procesos necesarios, desde la creación de la plantilla de los formularios, la unión de la misma con los datos, la compilación de los documentos, la generación, la impresión o envío se puede realizar de una forma visual y sencilla.

La solución consta de varios módulos que integran una solución de mayor alcance:

- ***Designer:*** parte donde se diseñan los formularios. Cada uno de los formularios pueden estar formados por varias partes o páginas y pueden ser estáticas o dinámicas:
    - Las partes estáticas están formadas por campos que serán rellenados posteriormente con los datos de los clientes: tablas, gráficos...
    - Las partes dinámicas se designan por un icono y cambian en función del cliente, son partes específicas e individuales.

- ***Builder*:** parte que se encarga de la configuración del documento. En ella se establece aspectos como: las páginas que forman el documento, formatos que se van a generar, si van a ser almacenados o enviados, el origen de los datos y orden de los datos...
- ***Dynamic*:** configura las partes dinámicas, dependiendo de diferentes variables puede cambiar su contenido. Por ejemplo, dependiendo de la edad del cliente se aporta una u otra información o se presenta con un color diferente…
- ***Job Flow*:** define los datos de entrada que rellenarán los formularios y las salidas, como el tipo de almacenamiento o envío. Definen las entradas y se define también el formato (PCL, PDF, HTML...) y tipo de salida (pc, mail, fax...).
- ***Pagination*:** configura qué páginas conforman el documento que se va a generar.
- ***Definer*:** Interpreta el esquema de los datos del cliente y une esos campos con los campos que tiene el formulario, para que cuando el DGE vaya a realizar la generación de los documentos, los formularios estén rellenados con los datos del cliente.

DocPath® Areca, crea proyectos con extensión IDF, que posteriormente son compilados y transformados en archivos de tipo IPF. Estas extensiones son internas y el cliente sólo utilizará las salidas que si serán formatos (PCL, PDF, HTML...). Los archivos podrán también guardarse, ser enviados por correo, fax o ser impresos, según se prefiera.

Todo lo anterior serían los procesos que se siguen en la parte de diseño. La segunda de las partes, la parte de generación la forma el DGE, que es el encargado de generar los documentos y archivos y está compuesto por las siguientes partes:

- ***Controller*** o ***JobLauncher:*** procesa y gestiona los procesos de generación.
- ***DataProcessor:*** enlaza las variables de usuario con los datos suministrados.
- ***DocProcessor:*** genera el documento lógico.
- ***MailFax Processor***: envía mail/fax informando del envío de los trabajos sugeridos, es un ejemplo de módulo encargado de la salida.

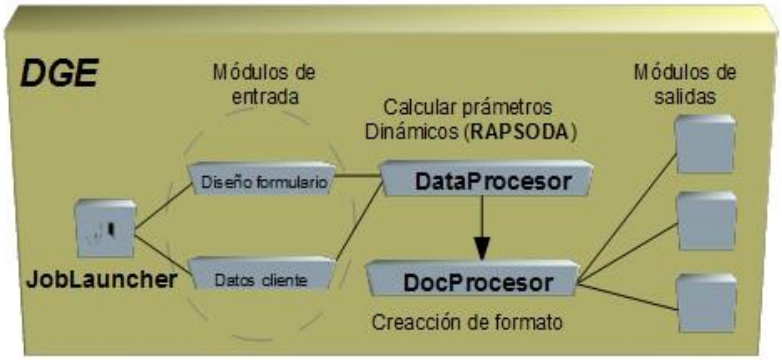

Figura 5:Generación de gestión documental DGE

En primer lugar, el *JobLauncher* inicia o lanza el trabajo, posteriormente, la información que le llega por los módulos de entradas es enviada al *DataProcessor* que calculará la parte dinámica del documento. Posteriormente, pasará al *DocProcessor* que generará el documento en el formato que se indique (PDF, PCL...) y por último tendremos el módulo de salidas donde se elige si se almacena, envía por correo, fax o imprime el documento. Hay que resaltar que entre la parte de diseño y la parte de generación existe un vínculo que las gestionaría que sería el Controller. El Controller es necesario ya que la salida del diseñador, que será una plantilla de un documento, es una de las entradas del generador.

En el Anexo B, se detalla mucho más toda esta parte, ya que el despliegue del proyecto tiene mucha relación con esta parte, en la que se explica cómo será la instalación del servicio.

3.1.5 Soluciones actuales de la empresa

Todas las soluciones DocPath® son modulares, ya que un usuario puede acoplar varias juntas. Las soluciones de DocPath®, se pueden dividir en dos tipos, *Enterprise*, soluciones para grandes empresas con un gran volumen de trabajo y *Business*, enfocadas a las pymes, medianas y pequeñas empresas. Todas estas soluciones son las soluciones que se integrarán de manera cloud gracias a la plataforma que se pretende crear, por lo que se considera que es interesante ver por encima sus características.

3.1.5.1 Soluciones Enterprise

I. Remote Office Printing

Su idea fundamental es la generación y distribución remota de documentos. Consta de un motor de generación de DocPath® que estará alojado en una sede central y desde cualquier sede podemos enviar información al motor de generación para generar los archivos necesarios. Destinado a empresas con diferentes sucursales. Funcionalidades:

- Diseño y almacenamiento de plantillas en un repositorio, generación de documentos en formato archivo y formato impresión.
- (opcional) Conexión segura entre cliente-servidor. Envío de datos desde impresora.

Figura 6:Remote Office Printing

II. High Volume Document Processing

Solución para la generación de miles de documentos y distribución a través del correo. Es útil, para una empresa de publicidad que generan información y envían a sus clientes. El funcionamiento es similar al producto anterior, pero con algunos añadidos:

- Lenguajes avanzados de impresión. Herramientas de integración y transformación.
- Envío de documentos automático por correo o fax y envío de datos por impresora

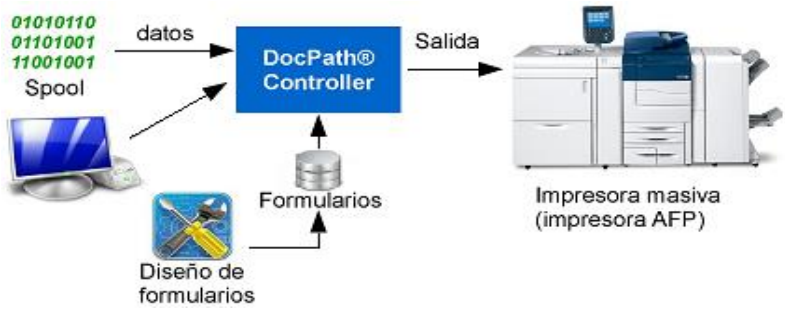

Figura 7:High Volume

III. Webdocs Filler

Es un diseño de plantillas de documentos que consta de un servidor web en el que están esas plantillas. Aún no está completamente terminado, es modular y se puede añadir a cualquier otro producto. Funcionalidades:

- Diseño de plantillas de documentos, almacenamiento de plantillas en repositorio y generación de documentos en formato archivo.
- Generación de formularios y documentos web.

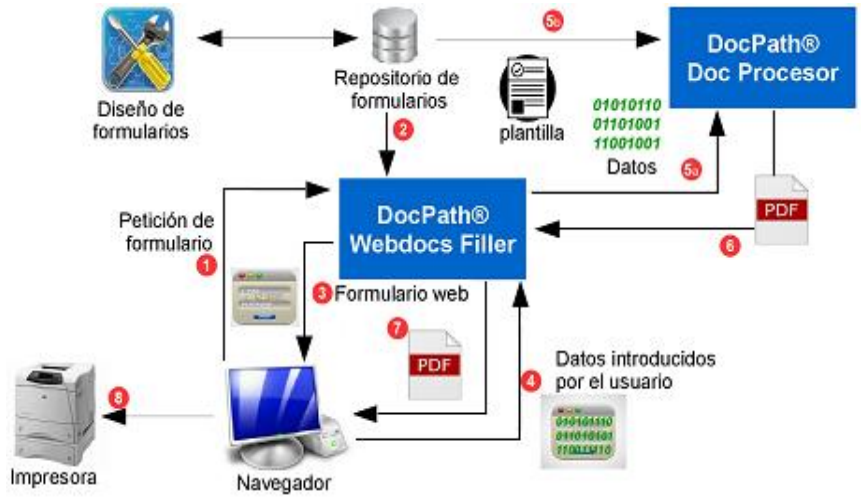

Figura 8:Webdocs Filler

IV. DocPath for Citrix Xenaap

Es producto hecho sólo para un cliente concreto. Está desarrollado sólo para la v 4.0 y 4.1 y en 5.0 no se desarrollará. Genera documentos, pero en entornos *Citrix (*entornos remotos). Se gestiona el software como aplicación de *Citrix* para usar en esa tecnología. Además, el tipo de salida correspondiente a este producto es por envío, correo o fax.

V. Webdocs Generation

Generación de documentos a través de la web. Un ejemplo puede ser una empresa de seguros con tres roles: central, sucursales y corredores de seguros. Cuando este último, vende un seguro, tiene que meter los datos del cliente en su app y enviarlos por correo a la central. Para hacer esto posible, se crea una conexión segura entre el corredor y la sede.

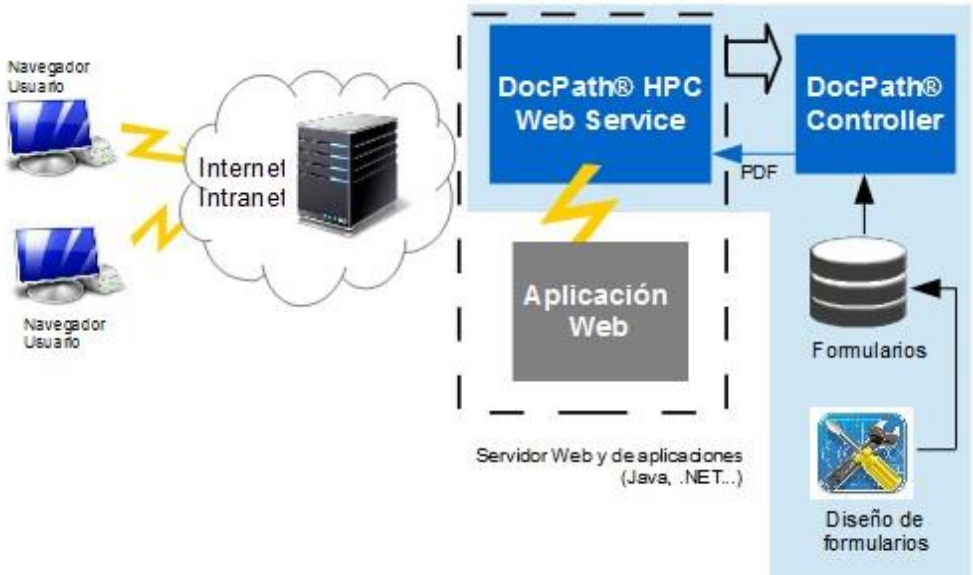

Figura 9:Webdocs Generation

VI. ASPEN

Es la solución más grande que dispone DocPath®. Además de un generador de documentos tiene un gestor documental y un digitalizador de documentos. El gestor documental es un almacén donde guardar documentos. Mediante una aplicación web, se pueden visualizar los documentos, hacer búsquedas, eliminar, modificar...

Por otro lado, está el digitalizador, que consiste en: transformar elementos impresos a documentos digitales. Se puede hacer de manera manual, o de manera automática con ayuda de un código de barras.

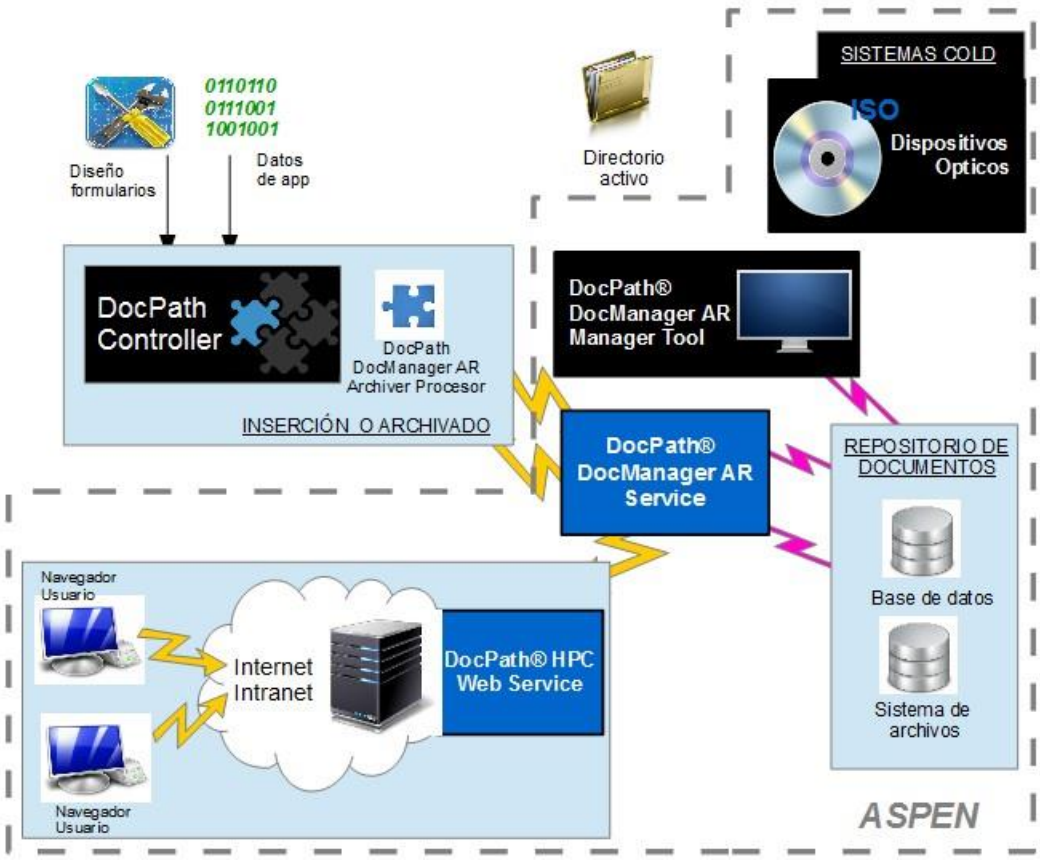

Figura 10:ASPEN

3.1.5.2 Soluciones Business

Son soluciones enfocadas a pequeñas y medianas empresas con un volumen de gestión documental más bajo que las empresas que desarrollan soluciones *Enterprise*.

I. Business Suite Essential

Se trata de una generación de documentos muy básico, que únicamente puede generar impresiones en PCL. En resumen, es un diseñador y generador de PCL.

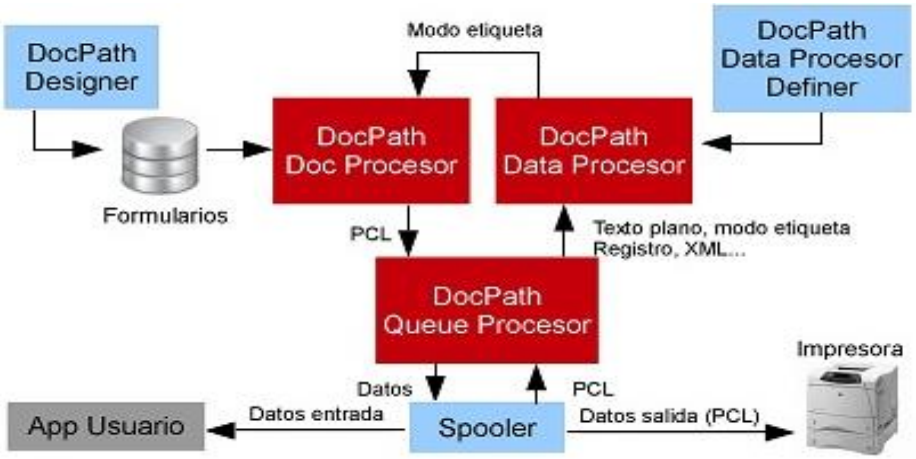

Figura 11:Business Suite Essential

II. Business Suite Pro

Misma solución que *Business Suite Essential*, pero distintas salidas: PDF, correo y fax.

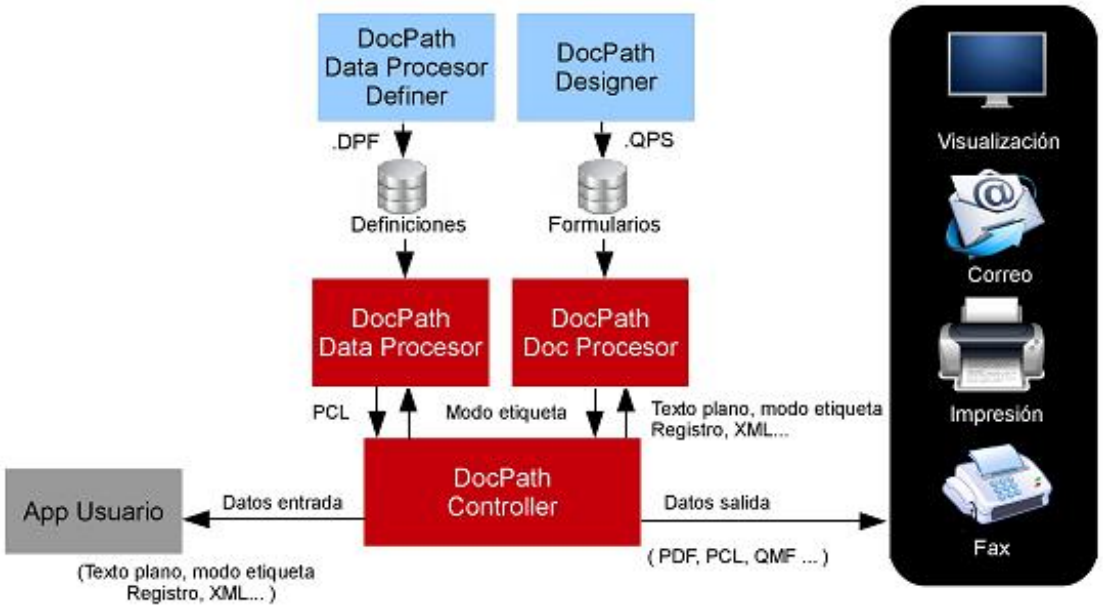

Figura 12:Business Suite Pro

III. Business Suite Industrial

Sirve para generar impresiones, pero en lenguajes de impresión diferentes a PCL, generalmente enfocada a la impresión de etiquetas. También se hace en etiquetas RFID, que contienen un chip en su interior con información.

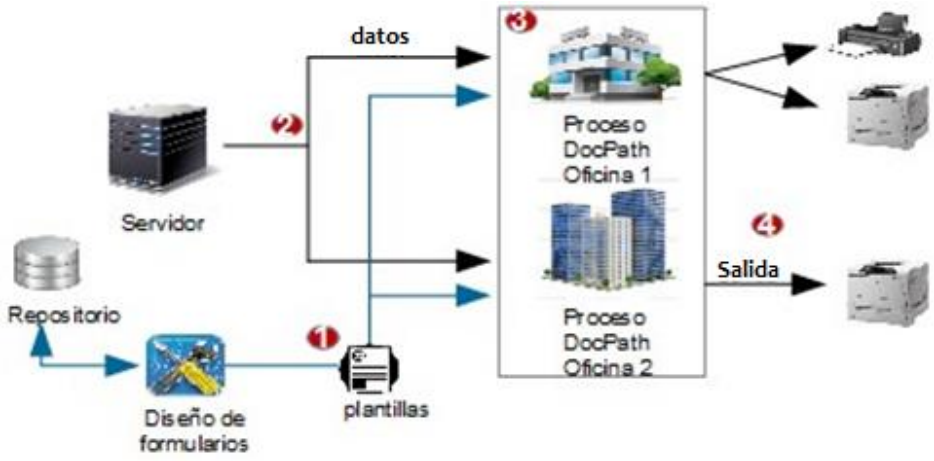

Figura 13:Business Suite Industrial

IV. Boulder Suite

Tiene las mismas funcionalidades que los productos *Business Suite*, pero con la peculiaridad de que transforma productos de una empresa extinguida (*InfoPrint*) a productos DocPath. Enfocado a clientes que trabajaban con la competencia anteriormente.

- *Boulder Suite Essential*: realiza impresiones AFP.
- *Boulder Suite Pro*: realiza impresiones de todo tipo y guarda en diferentes formatos para posteriormente poder enviar por correo o fax.
- *Boulder Suite Advanced*: realiza impresiones masivas, para más volumen de trabajo. La diferencia entre el Pro y el Advanced, tan sólo se diferencia en lo anterior.

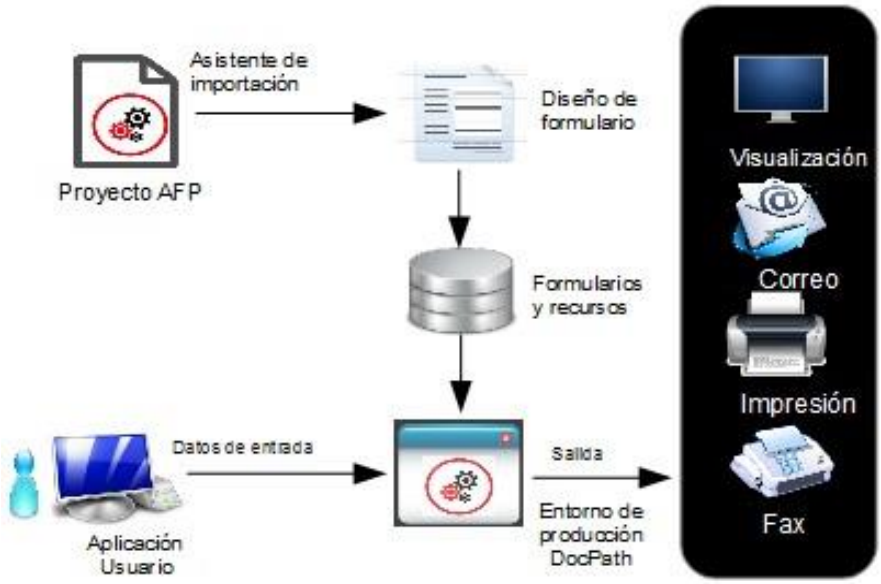

Figura 14:Boulder Suite

V. Penn Central

Es un producto ya desaparecido y era un gestor de impresiones que no llegó a generar ninguna solución funcional útil, presentó muchos problemas y se decidió matar el producto.

3.1.5.3 Otras soluciones

I. Toner MIST

Es una solución que tiene como objetivo principal reducir los altos costes de consumo de tóner o tinta de una organización e incrementa los niveles de eficiencia, productividad y rendimiento en los procesos de impresión. Características o ventajas principales son:

- Es un proceso fácil de integrar a su infraestructura existente
- Ofrece soporte para plataformas *Windows* (*Windows 7, 8.1, 10...*)
- Es compatible con impresoras locales y de red
- Compatible con otras soluciones *DocPath*, como *Remote Office Printing*.

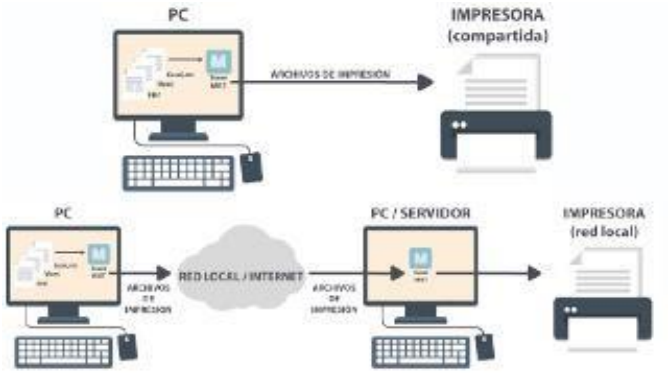

Figura 15:TonerMIST (imagen propiedad de DocPath)

TonerMIST se instala en cada equipo de trabajo y opera entre la app del usuario y la impresora. De cara al usuario, no existen cambios, sigue el procedimiento habitual para imprimir su documento. *TonerMIST* puede operar localmente y en red.

II. Output Dynamic

Es una solución que cubre los procesos de diseño, generación y el pre y el postproceso de documentos profesionales. Para empresas con un alto volumen de documentos en batch (archivos sin formato) y requieren capacidades altas en gestión documental. Capacidades:

- Altas capacidades de integración con otras apps de negocio existentes: *ERPs* y *CRMs*
- Múltiples formatos de salida: *PDF, AFP, PCL, PostScript, EPL, ZPL, HTML5*, archivos por fax o correo, almacenamiento en disco o base de datos…

Figura 16:Output Dynamic

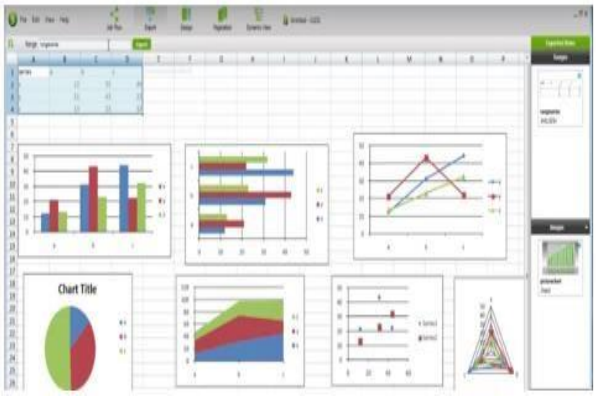

Figura 17:Excel Dynamic

III. Excel Dynamic

Es una herramienta desarrollada para proveer de una potente herramienta al sector financiero, con la que ofrecer informes detallados y flexibles. Crea hojas de cálculo a partir de la fusión de plantillas previamente diseñadas y los datos proporcionados por aplicaciones financieras existentes. Mediante el uso de Ipanema Technology, Excel Dynamic ofrece una interfaz de desarrollo fácil de usar y una app de Java muy similar a Excel.

3.2 Crítica al estado del arte

La situación actual en la empresa es muy distinta a la que queremos llegar con nuestra propuesta. La empresa funciona con productos que se instalan en la propia máquina del cliente. Para adquirir cualquier producto DocPath® se tiene que hacer a través de "Magallanes", es una especie de ERP/CRM desarrollado por DocPath® donde se alojan todas las licencias, divididas por versiones. Por un lado, están las versiones 3.3 y 4.0 y por otro las versiones 4.1 y 5.0.

En primer lugar, hay que diferenciar si es la primera vez o tan sólo se quiere renovar la licencia que se tenía anteriormente. En el caso de que sea la primera vez que se intente contratar este producto, se tiene que descargar un instalador, que necesita un *Instalation Code (IC)*. Cuando se tiene el producto instalado correctamente en el hardware del cliente, para activar el producto se necesita o bien un *Activation Code* (AC) que suele ser un número. para versiones antiguas (3.3 y 4.0.) y para las versiones 4.1 y 5.0 es un archivo ". lic".

En el caso de ser una nueva actualización del producto y ya lo tienes instalado en tu infraestructura, no hace falta hacer el primer paso, y habría que pasar directamente a la etapa

de activación y dependiendo del tipo de versión del producto se hace con un método u otro, como se muestra en la imagen.

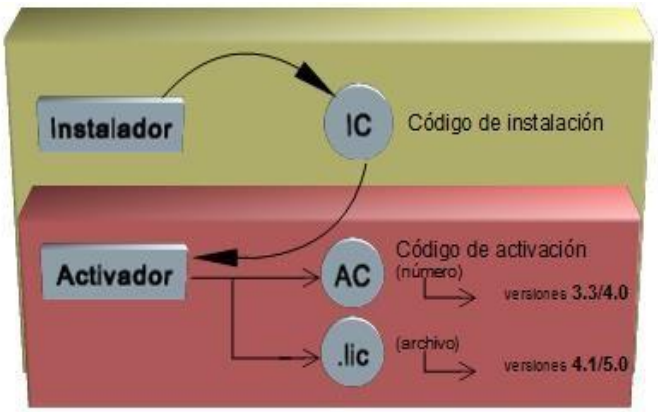

Figura 18: Instalación de un producto DocPath

En la actualidad, todo funciona así y no hay ningún problema, pero:

- Para seguir el proceso descrito anteriormente de instalación o activación de la licencia de un producto, hay que seguir un proceso arduo y pesado, que puede tener una extensa espera o incluso rechazo por el cliente, mientras que se instala.
- El cliente normalmente no tiene un conocimiento de cómo actualizar ciertos productos y se produce cierta espera, hasta que se pone en contacto con la empresa.
- Debido a la crisis que sacude a la sociedad, el cliente quiere recortar gastos y piensa que hay muchas horas que el producto no tiene, por ejemplo, el horario de trabajo es de 9h a 18h, por tanto, en otro horario que no sea ese, ese producto no es útil y creen que una manera de poder recortar gastos podría estar ahí.
- En muchas empresas, dependiendo del ámbito que trabajen, no le es útil un tipo de producto que no puedan limitar temporalmente la variación de la potencia suministrada. Sucede en cualquier empresa con picos temporales de trabajo.
- Los clientes con varios productos los gestionan de manera unitaria y quizás preferirían hacerlo en común, sobre todo en aspectos como moneda, tiempo…

Es cierto, que muchas de las críticas anteriores, se pueden gestionar de diferente forma o llegar a un acuerdo con el cliente, pero no hay una forma o un producto que pueda resolver esas cuestiones anteriores.

Capítulo 4

Método de trabajo

En este capítulo se define el proceso y los métodos que se han utilizado para elaborar el proyecto. Muchas de los aspectos que se narran en este capítulo han tenido que ser estudiados y evaluados mediante un informe previo en una fase previa al desarrollo, llamada Inception. Es una etapa previa porque no se puede considerar un sprint más, ya que la salida de un Sprint es un Incremento, es decir, software potencialmente liberable.

En esta fase se investiga sobre las soluciones *cloud*, metodología a seguir y herramientas a utilizar. También se analiza el alcance inicial y las alternativas con las que se puede solucionar el problema, comparándolas entre ellas y otros aspectos importantes para el desarrollo. Los estudios realizados en esta fase están en este capítulo y en anexos.

4.1 Condiciones de trabajo

El producto por realizar es nuevo e innovador en la empresa y es el alumno, el único desarrollador del mismo. Durante 6 meses, que es la duración de la beca Forte del alumno, la composición del equipo se mantendrá, Al finalizar ésta, se puede continuar del mismo modo, ampliarse con más miembros o cesar el desarrollo del producto, por algún motivo como falta de viabilidad en el proyecto. Además, el alumno estará guiado por un empleado de la empresa que hará labores similares a las de Scrum Master, que resolverá cualquier duda que el alumno tenga durante el desarrollo del proyecto.

El alumno debe amoldarse al plan o metodología que se sigue en la empresa, por ello tiene que formarse sobre diferentes aspectos:

- **Estándares respecto herramientas utilizadas en la empresa**: tiene que aprender a utilizar las herramientas con las que se trabaja en la empresa, algunas de estas herramientas son: *Maven* y *Nexus* empresarial, *Git*, *Redmine*...
- **Coordinación con el resto de equipos**: es posible que se tenga la necesidad de colaborar con otros equipos para llevar el producto a cabo. La coordinación será colaborativa y continua desde el momento que se requiera.

- **Metodología**: el alumno tiene que trabajar en un proceso ágil, ya que así se trabaja en DocPath®, por lo que debe formarse en este tipo de metodologías y concretamente en el *framework* de *Scrum*, aunque se pueden valorar otras opciones.

4.2 Metodología de trabajo

Se debe escoger una metodología que se ajuste al proyecto, que satisfaga la dinámica de trabajo y se amolde a la situación y condiciones de trabajo de la empresa. A continuación, se valorarán los dos tipos de marcos de trabajo más comunes y conocidos en el desarrollo de este tipo de proyectos.

4.2.1 Proceso Unificado[2]

El Proceso Unificado es un marco de desarrollo software que se caracteriza por estar dirigido por casos de uso. Puede estar definido por cuatro aspectos básicos:

- **Iterativo e Incremental**: El resultado final se obtiene a través de una serie de iteraciones e incrementos. Las iteraciones se pueden definir como pasos a seguir para completar el trabajo y los incrementos a los pequeños hitos que hacen crecer al proyecto. Es decir, el proyecto se divide en pequeños subproyectos que se representan como iteraciones y el resultado de conseguir ese hito, supone un incremento de valor al proyecto.
- **Dirigido por casos de uso:** Los casos de uso se utilizan para capturar los requisitos y definir las necesidades de los clientes o usuarios. Estas necesidades se tienen que recoger al principio del desarrollo y reciben el nombre de requisitos funcionales. La suma de todos estas necesidades o requisitos del usuario se unen en un esquema o modelo llamado: modelo de casos de uso, el cual, representa de una forma global toda la funcionalidad que tendrá el proyecto y que se tiene que conseguir al final del desarrollo del mismo.
- **Centrado en la arquitectura:** Existen diferentes modelos y vistas que definen la arquitectura del proyecto a realizar. Por hacer un símil, en la construcción de cualquier edificio, existen diferentes planos que combinados construyen la solución final, planos como la fontanería, electricidad, gas…

[2] http://yaqui.mxl.uabc.mx/~molguin/as/RUP.htm

- **Enfocado en los riesgos:** El equipo de desarrollo tiene que identificar los riesgos críticos del desarrollo del proyecto. Estos riesgos tienen que ser evaluados en un principio y a partir de ello se planifica todo el posterior desarrollo.

El Proceso Unificado de desarrollo puede ser dividido en cuatro fases para mejorar el desarrollo. Están situadas cronológicamente para favorecer la planificación del proyecto:

- **Inicio**: Fase en la que se obtienen los requisitos funcionales del sistema. En ella se hacen los primeros diseños, diagramas, planes y bocetos de la arquitectura del sistema. También se estima el coste y tiempo de desarrollo del proyecto. En resumen, es una fase de captación, definición planificación y estimación del proyecto.
- **Elaboración**: Fase en la que se especifican los casos de uso del sistema y se termina de diseñar la arquitectura del sistema. Se obtiene una visión refinada del proyecto a realizar. Se evalúan los riesgos y se ajustan aún más las estimaciones.
- **Construcción**: Es la fase más larga y abarca toda la evolución del proyecto, desde el principio hasta convertirse en un producto finalizado. Es decir, es la fase de implementación del proyecto en la que se lleva a cabo todos los requisitos, diagramas y bocetos diseñados en las fases anteriores.
- **Transición**: Fase en la que se prueba el resultado final, el producto. Es la fase llamada comúnmente como testeo en la que se buscan posibles errores y se intentan subsanar.

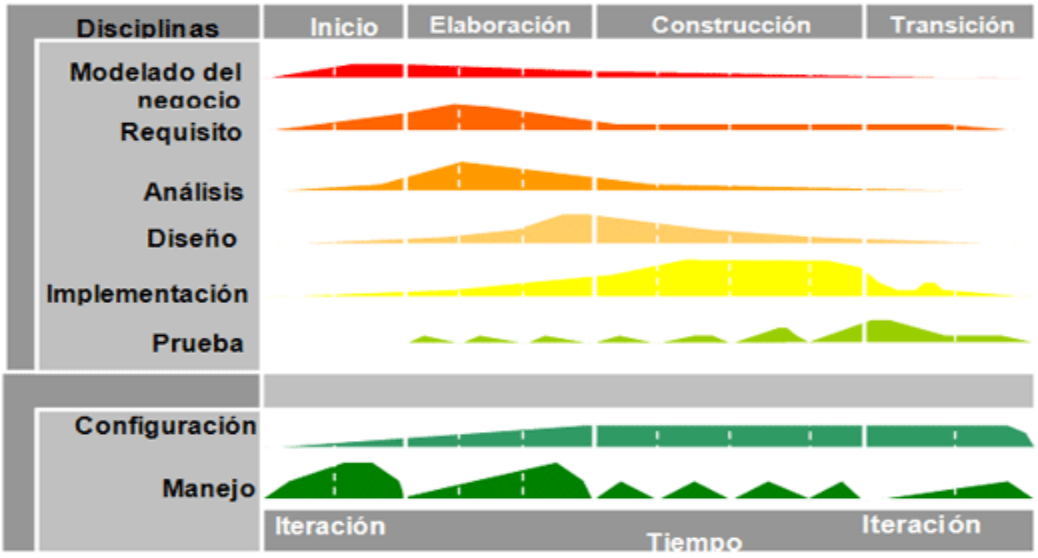

Figura 19: Fases del Proceso unificado

En la figura anterior se distinguen las distintas fases en el proceso unificado de desarrollo. Cabe destacar que, dentro de cada etapa o fase, se realizan distintos procesos o disciplinas, situadas estas en la parte izquierda del gráfico. Se observa que, en las primeras dos fases de Inicio y Elaboración, se hace todo lo relacionado con la planificación del proyecto. Dentro de esas categorías están el modelado de negocio, análisis de requisitos, análisis general del sistema, incluso la gran parte del diseño del proyecto. Por su parte, en la fase de construcción se implementa todo el proyecto de principio a fin dejando para la última de las fases la parte de pruebas.

Con este gráfico nos podemos hacer una idea de que aunque sea un marco de desarrollo iterativo, en cada iteración se realizan distintos trabajos, incluso llega un momento que no se vuelven a analizar ninguna parte del sistema, por lo que, si ocurre cualquier imprevisto, no se tiene reacción y no se puede subsanar cualquier error o cambio en los requisitos establecidos, cosa que posteriormente veremos que no nos beneficia y que con otros marcos de trabajo si se podría subsanar.

4.2.2 Scrum[3]

Scrum se denomina al *framework* o marco de desarrollo ágil en el que se aplican un conjunto de prácticas para colaborar en equipo y obtener un resultado óptimo. En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el cliente. Todas estas entregas son entregas válidas y funcionales desde el principio. Se caracteriza por:

- Adoptar una estrategia de desarrollo incremental, en lugar de una planificación y ejecución completa del producto.
- Basar la calidad del resultado más en el conocimiento de las personas en equipos auto-organizados, que en la calidad de los procesos empleados.
- Solapar las fases del desarrollo, en lugar de realizarlas en un ciclo secuencial.

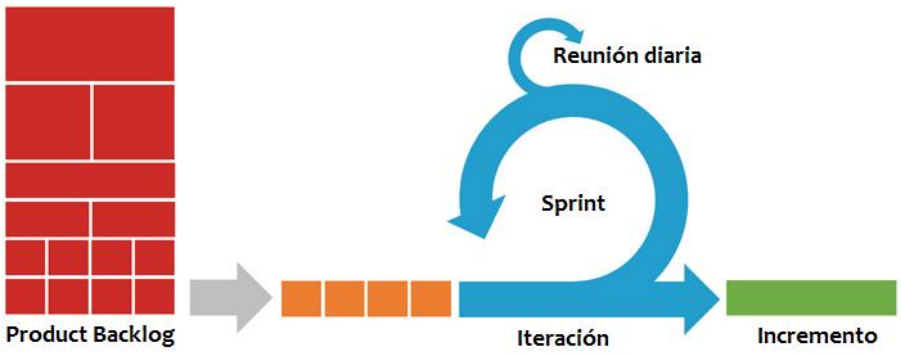

Figura 20: Desarrollo gráfico de una iteración en Scrum

[3] https://proyectosagiles.org/que-es-scrum/

El desarrollo de este marco de trabajo se dividiría en iteraciones y cada una de ellas tendría una secuencia parecida a la figura anterior. En ella se tendrá un Product Backlog o lista de tareas a realizar, se tendrá una breve planificación del Sprint, mediante una serie de reuniones y posteriormente se pasa a una fase de construcción que es el propio Sprint en el que los desarrolladores implementan las tareas de manera programática hasta que se obtiene el resultado esperado, que incrementa el valor de la solución actual.

Este tipo de procesos de desarrollo se recomienda para proyectos que se necesita tener un resultado óptimo temprano, proyectos donde existe mucha incertidumbre y no se está seguro de su viabilidad, proyectos innovadores o complejos donde no se puede hacer una planificación exacta o proyectos en los que las entregas se alargan demasiado en el tiempo.

Scrum se podría definir como un marco de referencia que define un conjunto de prácticas y roles y que puede tomarse como punto de partida para definir el proceso de desarrollo que se ejecutará en un proyecto. Un principio clave de *Scrum* es el reconocimiento de que los clientes pueden cambiar de idea sobre lo que quieren y necesitan, por lo tanto, Scrum adopta una aproximación pragmática, aceptando que el problema no puede ser completamente definido.

Como Scrum está enfocado al trabajo en equipo, se necesita una serie de personas con unos roles específicos y con un trabajo muy marcado:

- ***Product Owner (PO):*** Representa la voz del cliente. Se asegura de que el equipo Scrum trabaje de forma adecuada desde la perspectiva del negocio. Escribe las historias de usuario (*user story*), las prioriza y las coloca en el *product backlog*.
- ***Scrum Master (SM)***: Su trabajo es eliminar los obstáculos que impiden que el equipo de desarrollo alcance el objetivo del sprint actual. No es el líder del equipo, ya que el equipo de desarrollo se auto-organiza, sino que actúa como una protección entre el equipo y cualquier influencia que le distraiga. Se asegura que el proceso Scrum se utiliza como es debido y hace que se cumplan las reglas. El SM y el PO nunca pueden ser la misma persona, el marco obligatoriamente los separa en dos roles distintos.
- ***Development Team (DT)***: El equipo de desarrollo tiene como responsabilidad entregar el proyecto. Es recomendable que sea un equipo de 3 a 9 personas con habilidades necesarias para realizar el trabajo de un sprint completo (análisis, diseño, desarrollo, pruebas, documentación...)

- Otros roles:
    - ***Stakeholders*:** son las personas que hacen posible el proyecto y para quienes el proyecto producirá el beneficio acordado que justifica su desarrollo. Participarían solo durante las revisiones del sprint. Pueden ser clientes, proveedores, vendedores...
    - **Administradores**: establecen el entorno para el desarrollo del proyecto.

Además de todas las definiciones anteriores, se sigue una secuencia de trabajo muy marcada con varias reuniones que evalúan el estado del proyecto, alguna de ellas diaria:

- ***Daily Scrum***: reunión que se hace cada día de un sprint y se evalúa el estado del proyecto, además se tienen reglas específicas: o La reunión se fija a una hora concreta todos los días, en el mismo sitio y a la misma hora y tendrá una duración fija inferior a 15 minutos.
    - Todos los roles pueden asistir a la reunión, pero solo los implicados en el proyecto podrán hablar en ella (PO*, SM* y el *DT)*.

  En ella, cada miembro habla sobre su trabajo en la última jornada, lo que pretende hacer hoy y los problemas que se tienen, para que el s*crum master* haga su trabajo.
- ***Scrum* de *Scrum***: discusión que hacen los miembros del *development team* que comparten parte de un proyecto. Debaten sobre el trabajo, las tareas y los avances.
- ***Sprint Planning Meeting***: reunión que se produce al inicio de cada sprint y en la que se tratan aspectos como: seleccionar el trabajo que se hará., preparar el sprint backlog e identificar y comunicar la cantidad de trabajo a realizar en ese sprint.
- ***Sprint Review Meeting***: reunión en la que se revisa lo que se ha realizado en ese periodo de tiempo, tanto lo completado como lo no completado.
- ***Sprint Retrospective***: reunión después de cada sprint que hacen los miembros del DT junto con el SM y el PO. Cada miembro del equipo cuenta su impresión sobre el sprint realizado. Se hace para mejorar la realización y desarrollo del proyecto.

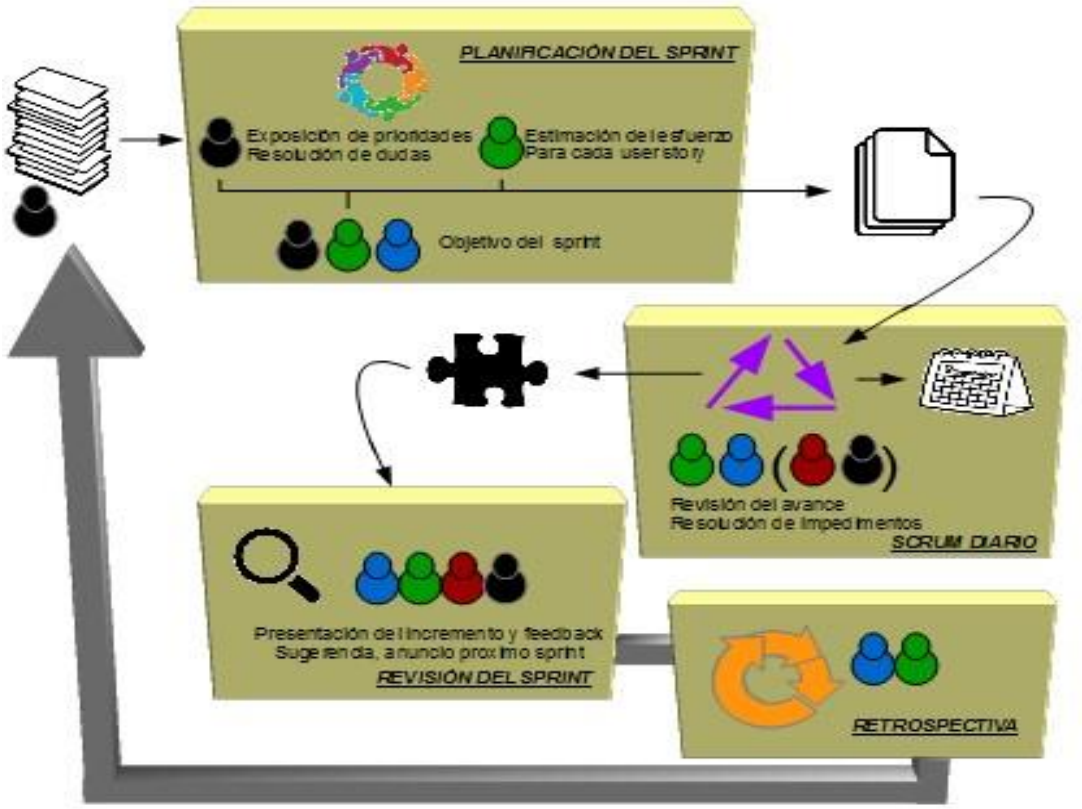

Figura 21:Funcionamiento Scrum

La figura anterior define las actividades que se realizan en Scrum por cada iteración. El *PO*, tiene una pila del producto que contiene la relación de requisitos del producto y el DoD, lista de actividades a realizar. Su función es priorizar y decidir sobre la lista y ponerlo en el *Product Backlog*.

Posteriormente se pasaría a la fase de planificación del sprint, en ella se tienen que hacer 3 cosas antes de tener los requisitos comprometidos por el equipo:

- Exponer las prioridades y resolución de dudas básicas (por el *product owner*)
- Establecer el objetivo del sprint (por el grupo entero)
- Estimar el esfuerzo para cada user story (por el equipo de desarrollo)

En la siguiente fase, nos encontraríamos con el propio Sprint y Scrum diario, que es una reunión previa en la que se revisa el avance. Esta fase acaba cuando se tiene una solución de ese sprint, en ese momento, se tendría un incremento del programa y se pasará a la fase de revisión del sprint por parte de todo el equipo de desarrollo incluido el PO y el SM y por último el equipo de desarrollo y el *SM* se reúnen para ver cómo ha ido y volver a planificar el siguiente sprint y así sucesivamente. Algunos elementos descritos anteriormente serían:

- ***Product Backlog***: conjunto de requisitos del proyecto, que contiene descripciones genéricas de funcionalidades deseables priorizadas. Sólo puede ser modificado por el *PO* y contiene estimaciones que ayudan a estimar la línea temporal.
- ***Sprint Backlog***: es el plan a seguir para conseguir el objetivo del Sprint.

Seguir el marco de desarrollo de Scrum repercute en una serie de beneficios como:

- **Flexibilidad**. Capacidad de reacción para adecuarse a las exigencias del cliente.
- **Reducción del tiempo de desarrollo.** El cliente tendrá una versión con la que trabajar al final del primer sprint, mientras que, si se hace sin una metodología ágil, esta primera versión se tendría al final de todo el desarrollo.
- **Mayor calidad de software y productividad.**
- **Predicción de tiempo y reducción de riesgos.** Se puede establecer el tiempo final al dividirse en pequeñas tareas y el riesgo se reduce con la priorización de requisitos.

4.2.3 Metodología a seguir en el proyecto

La metodología a seguir en el proyecto es una metodología ágil, concretamente será una adaptación de Scrum, adaptada a nuestras necesidades, ya que como hemos comentado, esta metodología está enfocada a grupos de desarrolladores y en nuestro caso sólo hay un solo desarrollador. Además, se utilizan buenas prácticas de otras metodologías ágiles como Extreme Programming (XP).

Se ha elegido este tipo de metodología porque el proyecto que se tiene que desarrollar es un proyecto que necesita tener un resultado óptimo temprano, porque se busca una plataforma mínima funcional en la que se pueda trabajar posteriormente. Además, es un proyecto que se tiene mucha incertidumbre porque la empresa es primeriza en este tipo de soluciones y nunca se han implementado soluciones ni proyectos cloud, lo que supone que

tanto la empresa como el desarrollador tiene que investigar y formarse en estos temas antes de afrontar la etapa de desarrollo.

En este proyecto, al tiempo que se realiza el desarrollo del sistema lo más normal es que aparezcan complicaciones, por lo que es preciso que la metodología sea flexible al cambio, que sea posible corregir la dinámica de trabajo y las tareas, con respecto a los inconvenientes que aparecerán durante el proyecto. Además, al ser un proyecto primerizo, es probable que los requisitos no sean fijos desde un principio y que se tenga que corregir el rumbo.

Como hemos comentado, se ha hecho una adaptación y se han llevado a cabo buenas prácticas del desarrollo de Scrum y de la metodología XP. De esta última, se ha utilizado su forma de desarrollo principal, *pair programming*, se ha utilizado para explicar cualquier duda por parte de compañeros de la empresa, aunque técnicamente no conformen un equipo de desarrollo.

Por parte de *Scrum*, hay algunos roles que sí tendrían a una persona asignada, como el *PO* que es la persona encargada del desarrollador el TFG, pero hay otros que no se pueden seguir exactamente, como: el equipo de desarrollo, éste tiene que estar formado por un grupo de 3 a 9 personas mínimo, y en nuestro caso no se cumple. Este aspecto nos limita a la hora de realizar algunas reuniones, por lo que el plan a seguir y elementos a utilizar serían los siguientes:

FASES*:*

- ***Inception***: fase inicial en la que se planifican, estudian y valoran las posibles soluciones propuestas y se discute sobre el funcionamiento y el plan a seguir. Se le suele llamar algunas veces *Sprint 0*, pero se hace de manera errónea, ya que para ser un *Sprint* se necesita tener algo potencialmente liberable, un incremento del proyecto.
- **Construcción**: fase que contiene las iteraciones que dividen el desarrollo del producto. La dificultad, preparación o adaptación, pueden variar el tiempo de estas etapas. Suelen ser de 1 a 4 semanas.

ROLES*:*

- ***Product Owner***: persona que conoce los requisitos funcionales que va a tener el sistema. Encargado de maximizar el valor del producto. Será el responsable de crear la lista de funcionalidades del sistema o también llamada *Product Backlog*.
- ***Scrum Master***: persona que facilita las reuniones de trabajo y trata de resolver los problemas que tiene el equipo. En nuestro caso los dos ultimo roles están representados por el jefe como PO y el SM sería el tutor en la empresa.
- ***Development Team***: grupo de desarrolladores para las funcionalidades del proyecto. En nuestro caso, nosotros solos.

REUNIONES*:*

- ***Daily Scrum***: reunión que se hace cada día y se evalúa el estado del proyecto y lo que se va a hacer ese mismo día. Al ser un solo desarrollador, no se hace una reunión, pero se establece un guion de lo que se va a realizar ese día y se estudia el día anterior.
- ***Sprint Review Meeting***: reunión en la que se revisa lo que se ha realizado en ese periodo de tiempo, tanto lo completado como lo no completado.
- ***Sprint Retrospective***: reunión organizada después de cada sprint en las que asisten, el desarrollador/es junto con el *SM* en la que el desarrollador cuenta su impresión sobre el sprint realizado. Tiene el propósito de mejorar la realización del proyecto.

COMPONENTES Y TÉCNICAS:

- ***Product Backlog***: lista de objetivos o PBIs (*Product Backlog Ítems*) ordenados por el PO, según un criterio concreto (económico, funcional, temporal...)
- ***DoD***: artefacto de transparencia que facilita el entendimiento de qué significa que un PBI está hecho correctamente. En él se detallan aspectos como el acuerdo de calidad o la cantidad de test que son necesarios para que un PBI esté terminado...
- ***Epic***: forma de dividir a los requisitos, pequeñas metas a la que se tiene que llegar que, a su vez, están divididos en *user story* y a su vez se descomponen en PBIs.
- ***Pair programming***: Es una técnica utilizada en *eXtreme Programming*(XP) y que consiste en agruparse por parejas y mientras que uno programa el otro dirige,

cambiando los roles en un periodo de tiempo corto. Es la manera que se ha seguido para las explicaciones y dudas por parte de los miembros de la empresa.

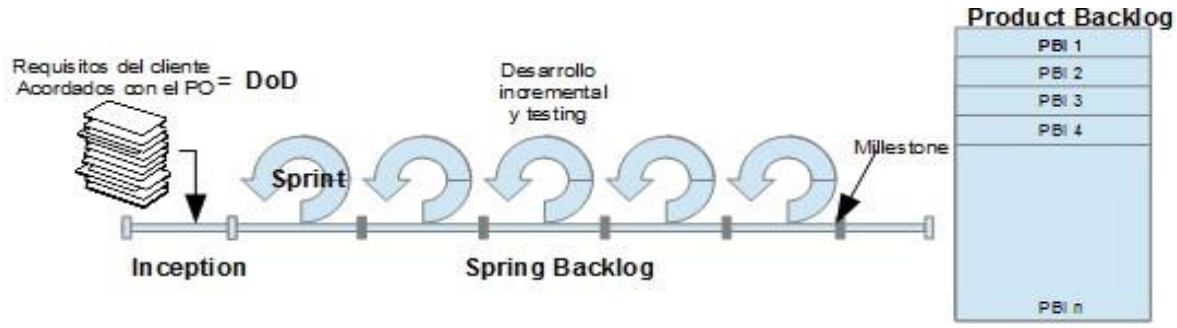

Figura 22:Etapa de desarrollo de metodología

Resumiendo, la primera fase es la de Inception, fase de aprendizaje y aclimatación al proyecto en la que se estudia y organiza el proyecto, el PO acuerda los requisitos y se establece el product backlog. Posteriormente, se realizan tantos sprint del tiempo establecido como se necesite hasta llegar a una meta (*millestone*).

Al acabar cada sprint, el SM organiza una reunión con el desarrollador, en la que debaten sobre la solución actual y acuerdan el nuevo product backlog y así, hasta concluir el proyecto o llegar a la meta establecida.

4.3 Marco tecnológico

A continuación, detallaremos todas las herramientas utilizadas para el desarrollo, gestión o documentación del proyecto, clasificadas por el ámbito en el que se utilizan.

4.3.1 Herramientas para la gestión del proyecto

- **Git**[4]: es un software de control de versiones diseñado por Linus Torvalds, pensado en la eficiencia del mantenimiento de versiones de aplicaciones cuando tienen un gran número de archivos de código fuente. El diseño de Git mantiene una enorme cantidad de código distribuida y gestionada por mucha gente. Lo utilizamos para guardar las versiones del proyecto.

[4] https://github.com/

- **Maven**[5]: es una herramienta de software para la gestión y construcción de proyecto, con un modelo de configuración de construcción está basado en XML. Utiliza un POM para describir el proyecto de software a construir, dependencias y componentes externos. Las dos características clave, que lo diferencia de los demás es, que está listo para poder utilizar en red y está construido usando una arquitectura basada en plugins.
- **Redmine**[6]**:** es una herramienta web para la gestión de proyectos de código abierto bajo la licencia GNU. Soporta base de datos MySQL entre otras, es multiplataforma y está compuesta por: un sistema de seguimiento de incidentes, un calendario de actividades, diagramas de Gantt para representar el tiempo en los proyectos, control de flujo de trabajo basado en roles, se puede integrar con *Subversion*, CVS, Git o *Darcs* y se pueden adjuntar archivos a los proyectos. Lo utilizamos para la planificación y asignación de tareas.

4.3.2 Herramientas para el modelado del proyecto

- **Visual Paradigm**[7]: herramienta para el desarrollo de aplicaciones utilizando el modelado UML, ideal para ingenieros de software, analistas de sistemas y arquitectos de sistemas. El modelado UML es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. En Visual Paradigm, se realizan diagramas de diferente índole y es para lo que lo utilizamos.
- **yEd**[8]: herramienta utilizada para la creación de los diagramas y figuras del documento. Es una herramienta gratuita.

4.3.3 Herramientas y tecnologías para el desarrollo del proyecto

- **Netbeans**[9]: es un entorno de desarrollo integrado libre, hecho principalmente para Java. Existen un gran número

[5] https://maven.apache.org/

[6] https://www.redmine.org/

[7] https://www.visual-paradigm.com/

[8] https://www.yworks.com/products/yed

[9] https://netbeans.org/

de módulos de extensiones que contienen clases de Java escritas que sirven para interactuar con diferentes APIs y así poder desarrollar diferentes tipos de proyectos.

- **Java**[10]: es un lenguaje de programación de propósito general, concurrente y orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Desde 2012, es el lenguaje en uso más popular. Su intención es permitir que los desarrolladores escriban el programa una vez y lo ejecuten en cualquier dispositivo.

- **JavaScript**[11]: es un lenguaje de programación interpretado. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en el lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario. Actualmente se utiliza para enviar y recibir información del servidor con ayuda de tecnologías como AJAX.
- **CSS3**[12]: significa "hoja de estilo en cascada". Usado para definir y crear la presentación de un documento estructurado escrito en HTML o XML. Se creó para separar la estructura de un documento web de su presentación.

- **HTML5**[13]: lenguaje básico de la web. Es la única versión en la que la versión clásica de lenguaje y la variante de XHTML se han desarrollado en paralelo. Una de las novedades principales es la inclusión del *Document Object Model*, que describe la estructura de un documento de acuerdo con el paradigma de la orientación a objetos.
- **AJAX**[14]: es un conjunto de tecnologías que permiten comunicarse asíncronamente en arquitecturas cliente servidor. Se utiliza para la parte web. Concretamente, se ha utilizado para cargar el contenido web sin actualizar la página.
- **Apache TomCat**[15]: servidor web integrado en el entorno de desarrollo y que funciona como contenedor y permite implementar todas las especificaciones de las

[10] https://www.java.com/es/

[11] https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/JavaScript_basics

[12] https://www.w3schools.com/css/css3_intro.asp

[13] https://www.w3schools.com/html/html5_intro.asp

[14] https://www.w3schools.com/js/js_ajax_intro.asp

[15] http://tomcat.apache.org/

Java Server Pages y de los servlets. Se ha utilizado para visualizar el contenido de la parte web integrada en Netbeans.

- **JSON**[16]: formato de texto para las aplicaciones. Se ha utilizado para comunicarse entre el cliente y el servidor y hacer posible la comunicación.
- **JUnit**[17]: conjunto de bibliotecas y clases utilizadas para la programación de pruebas unitarias en aplicaciones Java. Utilizado para hacer las pruebas del servicio desplegado.
- ***Bootstrap***[18]: es un *framework* de código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, menús y otros elementos de diseño basado en HTML y CSS. Bootstrap es modular y consiste esencialmente en una serie de hojas de estilo que los desarrolladores adaptan y convierten en páginas "*responsive*".
- ***JQuery***[19]: biblioteca de JavaScript que permite simplificar la manera de interaccionar con los documentos HTML, manipular el árbol DOM, manejar eventos, agregar interacción con la técnica AJAX y desarrollar animaciones. Fue creada en el 2006 y es de las bibliotecas más utilizadas en la actualidad.
- ***Hibernate***[20]: es una herramienta de mapeo objeto-relacional para la plataforma Java, que facilita el mapeo de atributos entre una base de datos relacional y el modelo de objetos de una aplicación, mediante XML o anotaciones en las clases. Hibernate es software libre.
- ***JPA***[21]: es la API de persistencia desarrollada para la plataforma Java EE. *Framework* del lenguaje de programación Java que maneja datos relacionales usando la plataforma Java en sus ediciones *Standard* (Java SE) y *Enterprise* (Java EE).

Con respecto a lo comentado anteriormente en Hibernate, se pueden utilizar archivos XML o anotaciones en las clases para realizar el mapeo entre los objetos y

[16] https://www.w3schools.com/js/js_json_intro.asp

[17] http://junit.org/junit5/

[18] https://getbootstrap.com/

[19] https://jquery.com/

[20] http://hibernate.org/

[21] https://www.tutorialspoint.com/es/jpa/jpa_introduction.htm

las tablas en las bases de datos, y en nuestro caso, se ha optado por utilizar *Hibernate + JPA*, utilizando así anotaciones en los objetos, en lugar de usar archivos XML.

4.3.4 Herramientas para la persistencia o base de datos del proyecto

- ***MySQL***[22]: sistema de gestión de bases de datos relacional desarrollado por Oracle Corporation y está considerado como la base de datos "*open source*" más popular del mundo. Está desarrollado por una comunidad pública, pero los derechos de autor están en poder de una empresa privada, por eso se distribuyen dos tipos de licencias, una *Community*, bajo licencia pública y otra *Enterprise*.
- ***MySQL Workbench*:** es una herramienta visual de diseño de bases de datos que integra desarrollo de software. Permite el diseño, creación y mantenimiento de una base de datos de manera visual.

4.3.5 Herramientas para la documentación del proyecto

- ***Mozilla Firefox* y *Zotero***[23]: Mozilla Firefox es un navegador web libre y de código abierto, desarrollado para *Windows, Android, OS X y GNU/Linux*. Usa el motor *Gecko* para renderizar páginas web. Lanzado el 22 de septiembre de 2004. La navegación es por pestañas, contiene un corrector ortográfico y búsqueda, marcadores, formato XML para compartir el contenido web y navegación por georreferenciación. Además de para buscar información relacionada con la documentación del proyecto, se ha utilizado para visualizar los cambios visuales en la interfaz web.

  Por parte de ***Zotero***, es un gestor de referencias bibliográficas, libre, abierto y gratuito que funciona también como servicio. Como aplicación es posible instalarlo como extensión en *Firefox*, o como programa independiente y está disponible para *Windows, Mac OS X y GNU/Linux.*

[22] https://www.mysql.com/

[23] https://www.zotero.org/

4.4 Plan de trabajo

En esta parte se narrará todo el trabajo a realizar, empezando por la estrategia a seguir y continuando por la descripción del plan de trabajo.

Hay partes que no se incluyen en el desarrollo del proyecto y que, se han realizado en la fase de *Inception*, como es el caso de la elección de proveedor que nos facilitará el servicio cloud y otros aspectos similares.

Todas estas partes estarán situadas en diferentes anexos y algunas incluso no se han incluido en este documento.

4.4.1 Estrategia técnica inicial

La estrategia técnica inicial, quedará resumida contando la arquitectura de la tecnología utilizada y la arquitectura de la interfaz. Para la primera parte, se establece el diagrama de pila de la arquitectura (a*rchitectural stack diagram*) y el diagrama de red (n*etwork diagram*), para la segunda parte aún no se ha establecido un diseño de la interfaz, pero al final del documento, en la parte de resultados se puede ver el diseño final de la misma.

4.4.1.1 Diagrama de arquitectura (architectural stack diagram)

Es un esquema donde se plasma de forma gráfica y simplificada lo que se pretende construir o diseñar. Su objetivo es ofrecer una visión simplificada del sistema, de forma que una persona, con tan sólo mirarlo, pueda entender lo que se quiere conseguir:

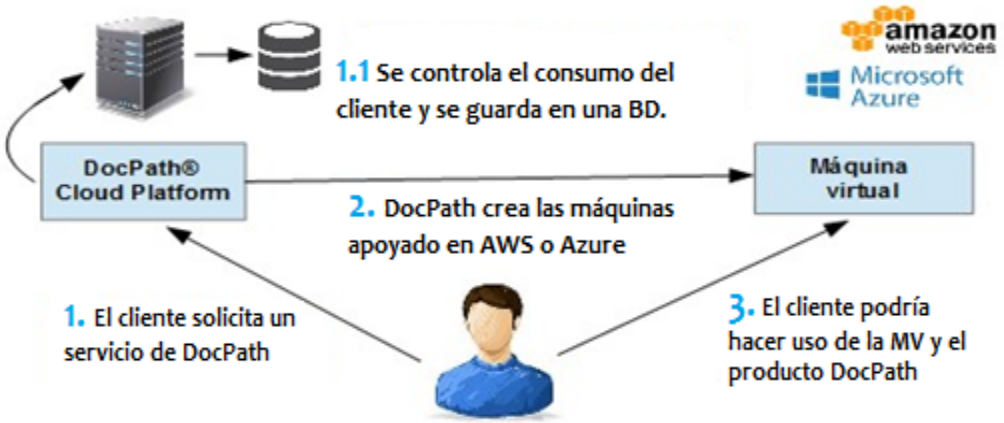

Figura 23:Diagrama de arquitectura

La plataforma por desarrollar está situada a la izquierda del diagrama. Primero, el cliente solicita un servicio a la empresa a través de la web (máquina en un *cloud* que permita desplegar el producto que solicita de DocPath®). A continuación, a través de nuestra plataforma se controla el consumo del cliente y lo guarda en una base de datos. El siguiente paso es que la plataforma cree las máquinas apoyado en la tecnología que proporciona *AWS*, para ello, se establece la configuración referente a *core*, almacenamiento, procesamiento... que el cliente haya elegido y a partir de ahí, hay que adquirir el espacio *cloud* suficiente para desplegar esa máquina y configurarla para que la máquina esté completamente lista para usarse, con el propio producto dentro de la máquina ya instalado y configurado. Por último, el cliente podrá hacer uso del producto, sin preocuparse de la instalación ni configuración.

4.4.1.2 Diagrama de red (network diagram)

El diagrama de secuencia es una forma gráfica de ver los PBIs, dependencias entre ellos y la ruta o camino crítico. El camino crítico es el camino que más coste tiene para el desarrollador, en caso de tiempo será el que más tiempo hay que dedicarle.

En nuestro proyecto, al ser un desarrollo utilizando una metodología ágil, se supone que ninguno de los PBIs tiene que ser dependiente de otro, ya que tienen que cumplir una característica prioritaria, que es la de la independencia. Si nos ceñimos estrictamente al método de priorización de requisitos, tendríamos una figura con una secuencia fija, donde hasta que no se tengan los PBIs de la primera de las categorías no podría empezarse la segunda, pero la priorización se puede replantear en los diferentes *Sprints*.

De toda la lista de tareas, las únicas tareas que seguro se van a llevar a cabo son las categorías clasificadas en la categoría Must, por lo que, las únicas que se va a cumplir la planificación de manera exacta son ellas, las demás pueden variar y su planificación en el tiempo puede ser variable. La planificación de las tareas a priori será la siguiente:

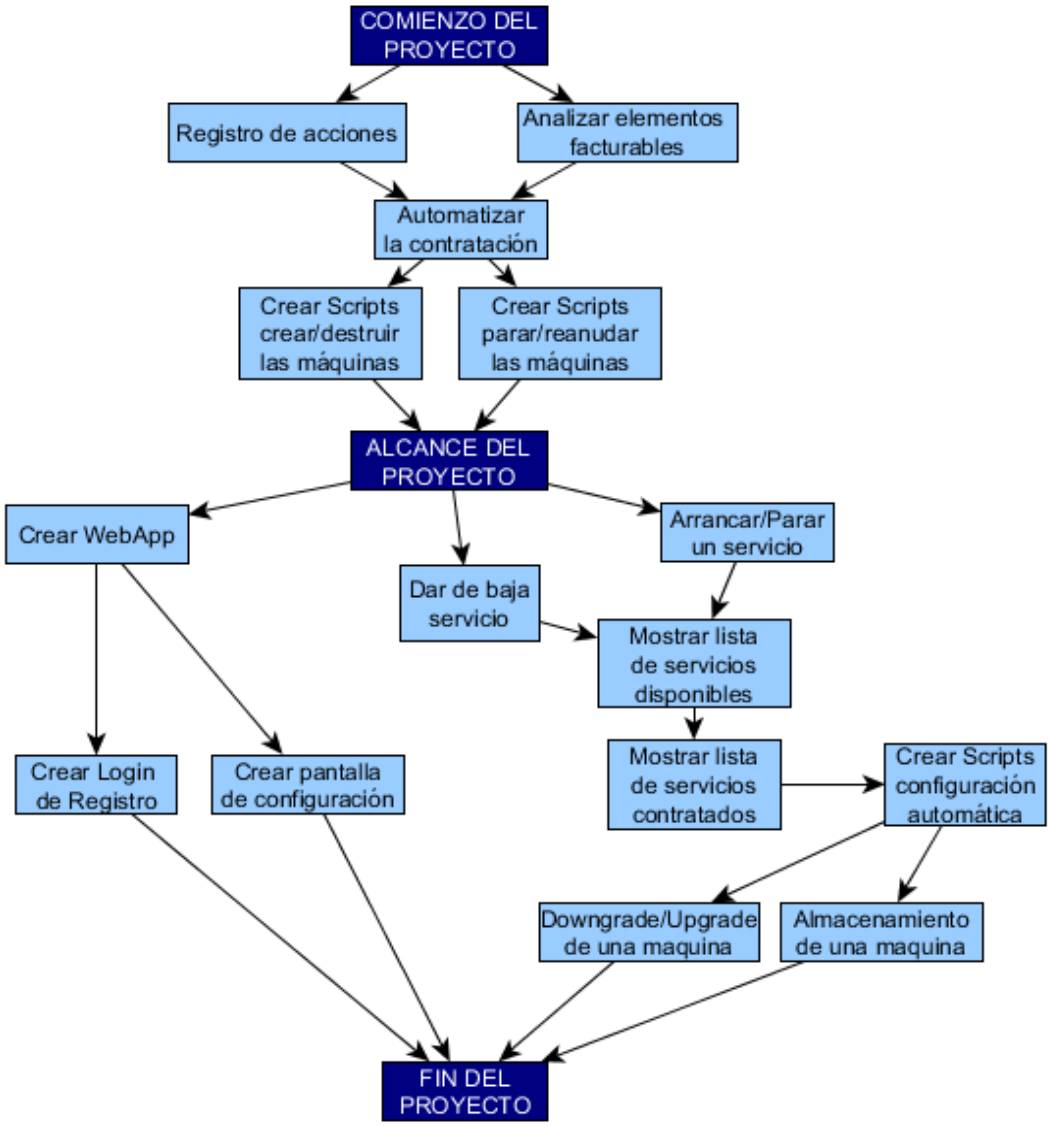

Figura 24: Diagrama de red

Como hemos comentado, este esquema es una aproximación, ya que los requisitos o PBIs pueden ser replanificados, y el fin del proyecto puede estar mucho antes que el indicado en la imagen. Es decir, podría darse el caso de que las tareas que comprende el *Should*, *Could* y el *Would* no se lleguen a realizar y por tanto con las tareas *Must* se finalizaría el proyecto. Por eso es importante y se hace énfasis que el desarrollo es iterativo e incremental y se prima más la colaboración con el cliente que la negociación inicial.

El diagrama de red hace referencia a los componentes y conexiones que hay entre ellos. Muestra los componentes de AWS, conexiones y la situación de nuestra plataforma.

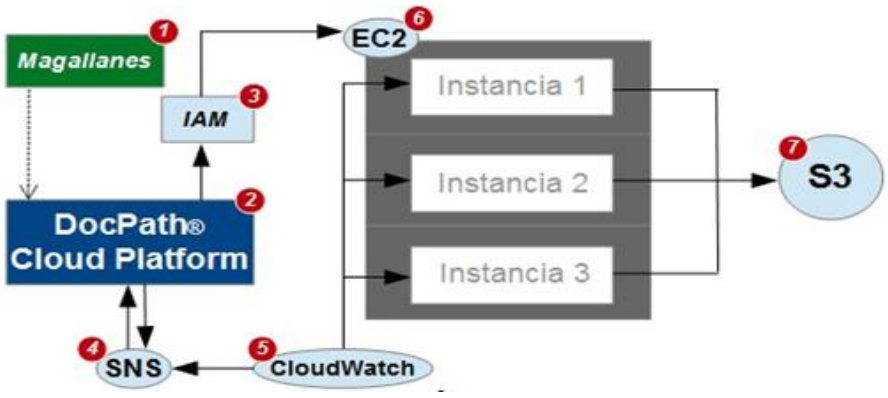

Figura 25:Diagrama de componentes

(1) **Magallanes**: web propiedad de DocPath® en la que se comercializan las licencias de todos los productos actuales de la empresa. La intención es situar, junto a las actuales licencias, un enlace a nuestra plataforma para gestionar los nuevos productos *cloud*.

(2) **DocPath® *Cloud Platform***: plataforma que actuará como centro de todo el sistema y el objeto a crear por parte de este proyecto, en ella se tendrá acceso a los productos DocPath® *cloud* que cada usuario tenga contratados.

(3) **IAM**: *Identify and Access Management*. Herramientas de AWS que sirven para crear, administrar y gestionar los usuarios. Cuando se crea cualquier usuario, se registra a través de *id* y *password* que se generan en un fichero (*key pair*) con el que se gestiona el acceso.

(4) **SNS**: *Simple Notification Service*. Es un servicio de notificaciones push, rápido, flexible que le permite enviar mensajes individuales o a varios destinatarios.

(5) ***CloudWatch**:* servicio de monitorización de los recursos de la nube de AWS y de sus aplicaciones, se utiliza para recopilar, realizar métricas y para establecer alarmas, es útil para avisar de posibles subidas de valores como coste, rendimiento...

(6) **EC2**: *Elastic Compute Cloud*. Dentro de todos los servicios de AWS, es el servicio central que nos proporciona el servicio *IaaS* que se pretende. Con este servicio somos capaces de crear instancias en las que instalar una AMI o máquina virtual concreta en la que desplegar nuestro servicio. Una AMI no son *ISOs*, son imágenes preparadas para ser usadas, incluso alguna puede ser preparada por nosotros con el *software* ya configurado.

En el propio producto de EC2 se tienen dos productos integrados muy interesantes: **ELB** es un balanceador de carga que configura y redirige el tráfico que llega por él, lo dirige a las máquinas que corresponda y establece un chequeo de máquinas para ver las que están operativas. ***AutoScaling***, encargado de elegir cuando se terminan las máquinas o es necesario que se creen otras, según la demanda de procesamiento en ese momento.

(7) **S3**: *Simple Storage Service*. Es un servicio que ofrece almacenamiento seguro para cualquier tipo de datos en los centros de datos de Amazon. Puede ser de utilidad para guardar cualquier aspecto relacionado con EC2, e incluso el almacenamiento adicional que es una de las funcionalidades pedidas por el cliente.

En resumen, la web principal de la empresa, Magallanes estará conectada con la plataforma que, a su vez, estará conectada con el servicio principal de AWS, EC2. A través de IAM se controla el acceso de los usuarios a las instancias. Con *CloudWatch* se puede monitorizar la información y crear alarmas para saber el consumo actual y el rendimiento del almacenamiento alquilado en S3.

4.4.2 Planificación de Release

Concepto de plan de release

La planificación de release consta de dos partes: la obtención inicial de una pila de producto, *product backlog*, con una estimación inicial de los elementos que lo componen y la replanificación del mismo, teniendo en cuenta aspectos surgidos en etapas anteriores, como cambios en los PBI o aspectos de *Sprints* anteriores. Es decir, una planificación anterior al propio desarrollo de las tareas.

Plan de release de nuestro proyecto

Para afrontar el plan de *release* de nuestro proyecto, hay que tener en cuenta una serie de condiciones que se tienen que tomar, ya que el proyecto es un proyecto peculiar y está enmarcado en unos plazos específicos e inamovibles. El proyecto está enmarcado dentro de un programa finito, llamado convenio Forte, por el cual, se tiene la posibilidad de cursar el proyecto en una empresa real con todos los beneficios que ello ofrece, pero el único inconveniente es que está acotado a unos plazos y por lo tanto se tiene una fecha final de entrega del mismo fijada, en el que se tiene que tener una plataforma funcional.

La estimación de las tareas estará hecha, por el desarrollador del proyecto. Es una estimación y no tiene por qué acotarse a ella. La cadencia de los *Sprint* se establecerá en un acuerdo entre todos los integrantes del *Scrum Team*.

En nuestro caso, viene fijada desde el principio, pero hay técnicas para estimar las duraciones de las tareas, por ejemplo, la técnica de *Delphi* en concreto una técnica llamada: *Planning Poker*, que trata de que todos los desarrolladores acuerdan en una votación privada la estimación de las tareas. No se puede realizar esta técnica porque sólo se dispone de un único desarrollador.

La duración de los *Sprint* tiene que ser fija, ya que es un principio de *Scrum.* Todos los eventos tienen una duración establecida y nunca puede ser variable, si se tiene cualquier problema y no se llega a tener lo que se desea en esa iteración o *Sprint,* se queda para el siguiente *Sprint. P*osteriormente, se consultarán los problemas que se han tenido. También hay que añadir que las estimaciones en el *Planning Poker* no se hacen en días concretos, son medidas arbitrarias.

Nuestro proyecto tiene una peculiaridad y es que antes de los sprint de desarrollo hay que estimar la duración de una fase de estudio y conocimiento en la que se estudiarán las posibles soluciones y tecnologías. La siguiente tabla expresa las estimaciones de las *tareas*:

Inception			
Obligatoria	Estudiar diferentes tecnologías, metodologías y herramientas	• Tener conocimiento sobre la empresa y los productos que se hacen en ella. • Aprender y aclimatarse a la metodología con la que trabajan. • Estudiar herramientas que se utilizarán. • Realizar un estudio de viabilidad y alternativas al proyecto • Realizar un estudio de proveedores para desplegar nuestros servicios. • Evaluar qué producto es de más utilidad para realizar el despliegue. • Realizar análisis e informes importantes como este, antes de entrar en el desarrollo del proyecto.	34

Release 1: Gestión de parámetros y base de datos.			
Prioridad	Necesito…	Para…	Estimación
Sprint 1			
1 (Must)	Registrar todas las acciones, las del cliente y las de la plataforma.	Para tener un control de todas las acciones y facilitar la gestión y facturación de todas ellas, necesario para la financiación de la empresa y coste para el cliente.	8
2 (Must)	Analizar los elementos facturables.	Saber los parámetros que se le tienen que facilitar al proveedor para la creación de la máquina y que luego facturarán.	5
Release 2: Contratación de máquinas			
Prioridad	Necesito…	Para…	Estimación
Sprint 2			
3 (Must)	Automatizar la contratación de la máquina	Tener una máquina creada en el menor tiempo posible y poder operar con ella. Es necesario tener una automatización de la máquina para acotar el tiempo que tarda en crearse la máquina.	13
Release 3: Creación de máquinas y funciones básicas			
Prioridad	Necesito…	Para…	Estimación
Sprint 3			
4 (Must)	Crear scripts crear / destruir la máquina	Para crear la máquina, las tareas a realizar serán: crear la máquina, asignar recursos, instalar y validar el producto, mientras que, en el caso de destruir, será justo, al contrario.	13
Sprint 4			
5 (Must)	Crear scripts parar / arrancar la máquina	Iniciar la máquina y desplegar las aplicaciones para arrancar la máquina, mientras que, para el caso de parar la máquina, el proceso es: apagar el servicio y apagar la máquina.	13

ALCANCE DEL PROYECTO

Release 4: Creación web y visualización de parámetros.			
Prioridad	Necesito…	Para…	Estimación
Sprint 5			
6 (Should)	Crear WebApp	Visualizar toda la configuración realizada en etapas anteriores de manera gráfica, gestionar las posibles acciones y visualizar los resultados cuando se realiza un cambio. Hacer más visual todo el proceso.	13
Release 5: Manipulación de servicios: Arrancar, parar y borrar			
Prioridad	Necesito…	Para…	Estimación
Sprint 6			
7 (Should)	Dar de baja servicios.	Eliminar todos los servicios contratados, con lo que conlleva: perder los datos, configuraciones y servicios guardados. Avisar y prevenir durante 48 horas en caso de nueva respuesta afirmativa, pasado este tiempo se borra todo.	8
8 (Should)	Arrancar/parar servicios	Iniciar la máquina y los productos instalados en su interior y replegar y apagar la máquina, en el otro caso	8
Sprint 7			
9 (Should)	Mostrar lista de servicios contratados.	Listar y ver los servicios que tienen contratados y ver un reporte de los mismos, incluyendo: producto, configuración, estado y botones para interactuar.	5
10 (Should)	Crear pantalla de configuración para aspectos técnicos.	Tener una página específica donde cambiar la configuración de los productos como: puertos, rutas, estándares...	8

Release 6: Añadir funcionalidades y mejoras técnicas a la plataforma.			
Prioridad	Necesito…	Para…	Estimación
Sprint 8			
11 (Should)	Crear scripts para configuración automática.	A partir de un formato, que aún no está creado, auto-configurar las aplicaciones según un documento en ese formato, para que toda esta configuración se haga de manera automática.	13

Sprint 9			
12 (Could)	Downgrade / Upgrade de la máquina	Añadir o quitar recursos en la máquina elegida. Crear más instancias en EC2 para soportar un crecimiento de volumen o lo contrario para un decrecimiento.	13
Sprint 10			
13 (Could)	Almacenamiento de la máquina	Elegir si se quiere tener un almacenamiento interno añadido para guardar los documentos generados.	13
Release 7: Añadir futuras mejoras funcionales.			
Prioridad	Necesito…	Para…	Estimación
Sprint 11			
14 (Would)	Login de usuarios (no registro)	Poder logearse los usuarios del sistema, ya que, para registrarse, se hará mediante alguna aplicación interna y de modo controlado.	8
15 (Would)	Mostrar lista de servicios disponibles.	Ver los productos que pueden ser contratados como productos cloud de la empresa. Esto estará implementado en la propia web de la empresa junto a las soluciones actuales.	5

Como se puede observar, el alcance del proyecto entero es demasiado grande para realizarlo en cinco meses que es el tiempo que se dispone en dicha empresa, por lo que la planificación y el P*lanning Release* se verá también afectado y se tendrá en cuenta tan sólo los siete primeros elementos del *Backlog*, que es el objetivo y alcance real del TFG, los demás son añadidos al mismo y posibles incrementos o implementaciones posteriores. Son estimaciones, puede que, ante cualquier imprevisto, la planificación varíe.

En la tabla anterior, se indica una fase antes de los propios *Sprint*, llamada *Inception*, es una fase difícil de acotar o estimar en el tiempo, ya que es una fase para consolidar el trabajo y el estudio y asegurarse de que el camino elegido es el correcto, esto queda reflejado en la estimación con un valor alto en comparación con los demás *ítems.*

4.5 Otros aspectos importantes

A continuación, se explican algunos aspectos que en la fase de Inception se han estudiado. Normalmente no se hace una planificación tan detallada en un proyecto de Scrum, ya que en esta metodología no se les da mucho valor a los temas de documentación, pero hay que estudiar varios temas como la financiación y el impacto que puede tener cualquier imprevisto en el proyecto.

4.5.1 Financiación

El proyecto está realizado en DocPath®, empresa líder en el desarrollo de *software* de gestión documental. El proyecto estará financiado como una beca, concretamente una beca Forte, que consiste en un acuerdo que la universidad establece con las empresas de un sector concreto y facilitan al alumno poder realizar prácticas y TFG en la misma empresa.

Además, ese trabajo se verá remunerado por la empresa que le pagará en función del contrato establecido por la universidad, incluyendo el coste de los diferentes de equipos y herramientas que utilizará el alumno en su estancia en la empresa.

4.5.2 Identificar Riesgos

En este apartado se detallarán los posibles riesgos que podría tener el desarrollo del proyecto, que pueden ser de varios tipos: riesgos de negocio, técnicos o de proceso:

- **Riesgos de negocios:** en esta categoría podríamos clasificar como riesgo, el posible cambio de requisitos, es decir: ¿Cómo gestionar los cambios en los requisitos? El *PO* puede en cualquier momento cambiar el orden de los requisitos, priorizando aspectos que no tenían tanta importancia para los clientes. Además de la re-priorización de los requisitos, también permite añadir o eliminar los requisitos en el *PB.*

  Otro aspecto podría ser, que un requisito sea demasiado grande y ambiguo para poder estimar el tiempo para realizarlo. Si esto ocurre en el análisis inicial no pasa nada, pero si el problema es detectado en el propio desarrollo, es mucho más severo. Las medidas serían subdividir esa *user story* y volver a planificar y priorizar los requisitos y el *PB.*

- **Riesgos técnicos:** podríamos hablar de riesgos como: identificar mal algunos aspectos en la estrategia técnica inicial y diferir más de lo acordado en las decisiones arquitecturales y de diseño. Podría haber cambios de componentes o plataformas. Se

tiene que estar preparado, sobre todo, para el posible cambio de plataforma, arquitecturas o herramientas de desarrollo sin que supongan un cambio especial. Estos riesgos se evalúan en la *Inception Phase*.

- **Riesgos de proceso:** este tipo de riesgos son los riesgos de cambio de metodología, pero en ese sentido estamos muy seguros de que la metodología elegida, tiene que ser una metodología ágil, ya que, si no es así, todo lo anterior que hemos contado en riesgos, no podría ser posible. Si no se sigue una metodología ágil, aspectos como la re-priorización o eliminación de requisitos, no sería posible realizarlos. En otras metodologías no hacen un desarrollo incremental con diferentes *Sprints* o iteraciones, aumentando las funcionalidades del producto, sino que se marcan una meta fija desde el principio. Al ser un proyecto experimental, se busca una meta mínima funcional.

4.5.3 Modelos del proyecto

Para entender mejor el proyecto, se hará por cada requisito un estudio de tres modelos o diagramas, que serán el modelo de dominio, proceso e interfaz. No necesariamente tiene que haber uno por cada requisito, pero sí uno general que recopile toda esa información.

- **Modelo de dominio**: modelo que puede utilizarse para capturar, expresar y facilitar el entendimiento ganado en un área bajo análisis, es una parte del diseño de un sistema.
- **Modelo de procesos**: modelo en el que pueden apreciarse con facilidad las interrelaciones existentes entre distintas actividades, definir los puntos de contacto con otros procesos, así como identificar los subprocesos comprendidos.
- **Modelo de interfaz de usuario**: Las necesidades de la interfaz de usuario quedarán recogidas en prototipos de interfaz de usuarios. Se pueden hacer en papel o utilizando screen sketches, en ellos se puede ver un boceto de la interfaz de usuario.

Todos estos modelos quedarán representados en un anexo A, en la parte final de todo el proyecto, allí se mostrarán los propios bocetos de los modelos y una breve explicación.

Capítulo 5

Resultados

En este punto se detalla todo lo relativo a la evolución e implementación de la solución propuesta, descrita en apartados anteriores. Este apartado solo explica la parte programática del proyecto, ya que toda la parte de investigación estará detallada en el apartado previo a éste. Hay que recalcar que los apartados anteriores pueden variar, ya que la metodología a seguir no es una metodología fijada, es una metodología agil en la que puede cambiar la priorización de requisitos.

5.1 Sprint 1. Análisis y registro

En la primera iteración o sprint se realizan las dos primeras tareas correspondientes al registro de las acciones y al análisis de los elementos facturables.

5.1.1 Introducción

Se considera que, para un mejor entendimiento, se haga de manera inversa al orden de prioridad establecido. El alcance inicial para registrar todas las acciones, administrar el tipo de máquinas virtuales que un cliente solicita y contrata y tener un control de todas las acciones que se realizan con las instancias/máquinas virtuales y para la segunda de ellas, analizar los elementos facturables, sería analizar los parámetros que se le tiene que facilitar al proveedor para la creación y facturación, tanto al cliente como a la empresa.

Analizar todos los elementos facturables.

Realizar un estudio detallado de las características de una máquina virtual y sus elementos facturables, es un trabajo demasiado largo, arduo y costoso como para establecerse de esa manera tan exhaustiva, es por ello que se ha considerado, que la mejor solución sea amoldarse a los tipos de instancias que AWS proporciona, estudiar lo que ofrece cada una y ofrecerle al cliente un amplio abanico de posibilidades.

En nuestro caso, el coste varía según las características de la AMI que se elija, aunque hay módulos y herramientas que permiten un balanceo y un escalado óptimo, para ajustar

mucho el presupuesto. No es una "facturación por uso" pero es una aproximación. Se ajusta la MV y se gestionan todas las acciones con ella para acotar el uso que se da de la misma.

Registrar todas las acciones.

Las dos primeras historias de usuario están muy ligadas, ya que al cambiar la segunda de ellas repercute a la primera. Al acotar la solución final y amoldarse a los tipos de máquinas ya definidos en AWS. Sólo hay que centrarse en controlar y apuntar el tipo de máquina que se usará, el precio por tiempo que supondrá y la cantidad de tiempo que esa máquina estará en funcionamiento. Para saber todo esto, sólo hay que tener en cuenta el estado de la máquina y apuntar los diferentes cambios de estados que la máquina. Con ello se tendrá un informe de cuando la máquina está operativa y se podrá facturar en función de su uso.

5.1.2 PBI 1: Analizar todos los elementos facturables

I. Introducción

El alcance inicial de este PBI sería: analizar todos los elementos variables para poder facturar en función de ellos, es decir, particularizar el producto cloud para cada cliente y almacenar la configuración de los productos en sí, para ser facturados por ello posteriormente. Tras estudiar la viabilidad, hacer un estudio de la tecnología en la fase de *Inception*, se considera innecesario este estudio tan específico y minucioso sobre los elementos facturables que se pide para esta segunda *user story* y se ha considerado que es mejor opción amoldarse a las máquinas o AMIs que ya están creadas y que facilita AWS.

La opción elegida es elegir una máquina virtual personalizada, dentro del rango que se le ofrece al cliente, ya que elegir entre todos los aspectos, hace que se tenga una cantidad ilimitada de configuraciones y una cantidad de aspectos que controlar demasiado grande.

II. Tipos de instancias de AWS[24]

Para uso general.

➢ **Instancias T2**: Instancias de desempeño con ráfagas. Instancias acumulan créditos cuando están inactivas y los utilizan cuando están activas. Son muy buena opción para trabajos que no usan la CPU por completo o constantemente, pero que de vez en cuando se tiene picos o ráfagas que si necesitan de su potencia. Características:

- Procesadores Intel Xeon de alta frecuencia.
- CPU en ráfagas, que se rige por créditos CPU y desempeño de base constante.
- Equilibrio entre recursos de informática, memoria y red.

Modelo	CPU virtual	Créditos por hora	Memoria (GiB)	Almacenamiento	Precio
T2.nano	1	3	0.5	Solo EBS	0.007 $
T2.micro	1	6	1	Solo EBS	0.014 $
T2.small	1	12	2	Solo EBS	0.028 $
T2.medium	2	24	4	Solo EBS	0.056 $
T2.large	2	36	8	Solo EBS	0.112 $

Los precios son orientativos y varían según la región en la que se contrate la instancia de EC2. La instancia t2.micro es gratuita el primer año gracias a la capa gratuita que ofrece AWS, por lo que ese sería el precio cuando la capa gratuita no esté activada.

➢ **Instancias M4 y M3:** Son la última generación de instancias de uso general. Proporciona un equilibrio de recursos informáticos, de memoria y red, por lo que constituye una buena opción para muchas aplicaciones. Generalmente se usan para bases de datos pequeñas y medianas, ejecución de servidores *back-end* para SAP y para la informática en clústeres. Características:

[24] https://aws.amazon.com/es/ec2/instance-types/

M4:

- Procesadores Intel Xeon E5-2686 v4 de 2,3 GHz o Intel Xeon E5-2676 v3.
- Optimizados para EBS de manera predeterminada y sin costes adicionales.
- Soporte para redes mejoradas.

M3:

- Procesadores Intel Xeon E5-2670 v2 de alta frecuencia
- Almacenamiento de instancia basado en SSD para un rápido desempeño E/S.

Con optimización informática.

➢ **Instancias C4 y C3**: Disponen de procesadores de mejor desempeño y ofrecen la mejor relación precio/desempeño informático de EC2. Generalmente solucionan: flotas *front-end* de alto desempeño, servidores web, análisis distribuidos, app de ingeniería de alto desempeño, entrega de publicidad y codificación de vídeo.

C4:

- Procesadores Intel Xeon E5-2666 v3 alta frecuencia optimizados para EC2.
- Optimizados para EBS de manera predeterminada sin costos adicionales.
- Capacidad de controlar las configuraciones del tipo de instancia c4.8xlarge.
- Soporte para las redes mejoradas y el almacenamiento en clústeres.

C3:

- Procesadores Intel Xeon E5-2680 v2 de alta frecuencia
- Soporte para redes mejoradas y soporte para clústeres
- Almacenamiento basado en SSD

Optimizadas para memoria.

- **Instancias X1**: Optimizadas para aplicaciones en memoria a larga escala de clase empresarial y ofrecen un coste más bajo por GiB de RAM. Se recomienda utilizar para bases de datos en memoria, motores de procesamiento de *Big Data* y aplicaciones informáticas de alto desempeño. Características:
  - Procesadores Intel Xeon E7-8880 v3 de alta frecuencia
  - Hasta 1 952 GiB de memoria de instancia basada en DDR4
  - Almacenamiento SSD y optimizadas para EBS por defecto sin coste adicional
  - Posibilidad de controlar la configuración del estado del procesador
- **Instancias R3**: Es similar a las X1, pero para bases de datos de alto desempeño, cachés de memoria distribuida e implementaciones más grandes. Características:
  - Procesadores Intel Xeon E5-2670 v2 de alta frecuencia
  - Almacenamiento en SSD
  - Soporte para redes mejoradas

Para informática acelerada.

- **Instancias P2 y G2:** Las instancias P2, están destinadas a aplicaciones informáticas de GPU de uso general, mientras que las G2 están optimizadas para aplicaciones con un uso intensivo de gráficos. Algunos usos para las P2 son: aprendizaje virtual, bases de datos de alto empeño o finanzas computacionales, mientras que por parte de G2, están más relacionadas con aplicaciones 3D y codificación de vídeo. Características:

  P2:

  - Procesadores Intel Xeon E5-2686 v4 de alta frecuencia
  - GPU NVIDIA K80 de alto desempeño, con 2496 núcleos de procesamiento.
  - Compatible con comunicación GPU punto a punto (GPUDirect)
  - Proporciona redes mejoradas con el adaptador de red elástico de EC2.
  - Optimizadas para EBS de manera predeterminada sin costos adicionales.

G2:

- Procesadores Intel Xeon E5-2670 de alta frecuencia
- GPU NVIDIA, con 1536 núcleos CUDA y 4 GB de memoria de vídeo.
- Cada GPU tiene un codificador de vídeo para soportar hasta 8 trasmisiones de vídeo HD (720p a 30fps) o 4 trasmisiones Full HD (1080p a 30fps).
- Soporte para codificación y captura de fotogramas de baja latencia.

Optimizadas para almacenamiento.

➢ **Instancias I2:** Son instancias de E/S elevadas, con alta capacidad de almacenamiento que ofrecen un almacenamiento muy rápido respaldado por SSD y optimizado, que proporcionan IOPS (operaciones de entrada/salida por segundo) elevadas a un bajo costo. Enfocadas para bases de datos NoSQL como MongoDB.

- Procesadores Intel Xeon E5-2670 v2 de alta frecuencia
- Almacenamiento en SSD y soporte para TRIM.
- Alto desempeño de E/S aleatoria.

➢ **Instancias D2:** Son instancias de alta densidad de almacenamiento que ofrecen hasta 48 TB de almacenamiento local basado en HDD. Enfocadas para el almacenamiento masivo MPP (*Massively Parallel Processing*) y sistemas de archivos distribuidos.

- Procesadores Intel Xeon E5-1676 v3 de alta frecuencia
- Almacenamiento HDD y alto desempeño en el momento del lanzamiento
- Alto rendimiento de disco y soporte para redes mejoradas de EC2

III. Características de las instancias

A continuación, se comparan los aspectos más importantes y característicos de las instancias que hemos definido anteriormente, esos aspectos son la CPU virtual, memoria, almacenamiento, desempeño en redes, tipo de procesador, velocidad de reloj y precio:

Uso general							
Tipo	CPU Virtual	Memoria (GiB)	Almacenamiento (GB)	Desempeño de redes	Tipo Procesador	Velocidad de reloj (GHz)	Precio (por hora)
t2.nano	1	0.5	Solo EBS	Bajo	Familia Intel Xeon	Hasta 3.3	$0.007
t2.micro	1	1	Solo EBS	Bajo / Moderado	Familia Intel Xeon	Hasta 3.3	$0.014
t2.small	1	2	Solo EBS	Bajo / Moderado	Familia Intel Xeon	Hasta 3.3	$0.028
t2.medium	2	4	Solo EBS	Bajo / Moderado	Familia Intel Xeon	Hasta 3.3	$0.056
t2.large	2	8	Solo EBS	Bajo / Moderado	Familia Intel Xeon	Hasta 3.0	$0.112
m4.large	2	8	Solo EBS	Moderado	Intel Xeon E5-2676	2.4	$0.132
m4.xlarge	4	16	Solo EBS	Alto	Intel Xeon E5-2676	2.4	$0.264
m4.2xlarge	8	32	Solo EBS	Alto	Intel Xeon E5-2676	2.4	$0.528
m4.4xlarge	16	64	Solo EBS	Alto	Intel Xeon E5-2676	2.4	$1.056
m4.10xlarge	40	160	Solo EBS	10 gigabits	Intel Xeon E5-2676	2.4	$2.641
m4.16xlarge	64	256	Solo EBS	20 gigabits	Intel Xeon E5-2686	2.4	$4.226
m3.medium	1	3.75	1 x 4 SSD	Moderado	Intel Xeon E5-2670	2.5	$0.073
m3.large	2	7.5	1 x 32 SSD	Moderado	Intel Xeon E5-2670	2.5	$0.146
m3.xlarge	4	15	2 x 40 SSD	Alto	Intel Xeon E5-2670	2.5	$0.293
m3.2xlarge	8	30	2 x 80 SSD	Alto	Intel Xeon E5-2670	2.5	$0.585
Con optimización informática							
Tipo	CPU virtual	Memoria (GiB)	Almacenamiento (GB)	Desempeño de redes	Tipo Procesador	Velocidad de reloj (GHz)	Precio ($ por hora)
c4.large	2	3.75	Solo EBS	Moderado	Intel Xeon E5-2666	2.9	$0.119
c4.xlarge	4	7.5	Solo EBS	Alto	Intel Xeon E5-2666	2.9	$0.238
c4.2xlarge	8	15	Solo EBS	Alto	Intel Xeon E5-2666	2.9	$0.477
c4.4xlarge	16	30	Solo EBS	Alto	Intel Xeon E5-2666	2.9	$0.953
c4.8xlarge	36	60	Solo EBS	10 gigabits	Intel Xeon E5-2666	2.9	$1.906

c3.large	2	3.75	2 x 16 SSD	Moderado	Intel Xeon E5-2680	2.8	$0.120
c3.xlarge	4	7.5	2 x 40 SSD	Moderado	Intel Xeon E5-2680	2.8	$0.239
c3.2xlarge	8	15	2 x 80 SSD	Alto	Intel Xeon E5-2680	2.8	$0.478
c3.4xlarge	16	30	2 x 160 SSD	Alto	Intel Xeon E5-2680	2.8	$0.956
c3.8xlarge	32	60	2 x 320 SSD	10 gigabits	Intel Xeon E5-2680	2.8	$1.912
Optimizadas para memoria							
Tipo	CPU virtual	Memoria (GiB)	Almacenamiento (GB)	Desempeño de redes	Tipo Procesador	Velocidad de reloj (GHz)	Precio ($ por hora)
x1.16xlarge	64	976	1 x 1920 SSD	10 gigabits	Intel Xeon E7-8880	2.3	$8.003
x1.32xlarge	128	1952	2 x 1920 SSD	20 gigabits	Intel Xeon E7-8880	2.3	$16.006
r3.large	2	15.25	1 x 32 SSD	Moderado	Intel Xeon E5-2670	2.5	$0.185
r3.xlarge	4	30.5	1 x 80 SSD	Moderado	Intel Xeon E5-2670	2.5	$0.371
r3.2xlarge	8	61	1 x 160 SSD	Alto	Intel Xeon E5-2670	2.5	$0.741
r3.4xlarge	16	122	1 x 320 SSD	Alto	Intel Xeon E5-2670	2.5	$1.482
r3.8xlarge	32	244	2 x 320 SSD	10 gigabits	Intel Xeon E5-2670	2.5	$2.964
Optimizada par informática acelerada							
Tipo	CPU virtual	Memoria (GiB)	Almacenamiento (GB)	Desempeño de redes	Tipo Procesador	Velocidad de reloj (GHz)	Precio ($ por hora)
p2.xlarge	4	61	Solo EBS	Alta	Intel Xeon E5-2686	2.3	$0.972
p2.8xlarge	32	488	Solo EBS	10 gigabits	Intel Xeon E5-2686	2.3	$7.776
p2.16xlarge	64	732	Solo EBS	20 gigabits	Intel Xeon E5-2686	2.3	$15.552
g2.2xlarge	8	15	1 x 60 SSD	Alta	Intel Xeon E5-2670	2.6	$0.702
g2.8xlarge	32	60	2 x 120 SSD	10 gigabits	Intel Xeon E5-2670	2.6	$2.808
Optimizadas para almacenamiento							
Tipo	CPU virtual	Memoria (GiB)	Almacenamiento (GB)	Desempeño de redes	Tipo Procesador	Velocidad de reloj (GHz)	Precio ($ por hora)
i2.xlarge	4	30.5	1 x 800 SSD	Moderado	Intel Xeon E5-2670	2.5	$0.938
i2.2xlarge	8	61	2 x 800 SSD	Alto	Intel Xeon E5-2670	2.5	$1.876

i2.4xlarge	16	122	4 x 800 SSD	Alto	Intel Xeon E5-2670	2.5	$3.751
i2.8xlarge	32	244	8 x 800 SSD	10 gigabits	Intel Xeon E5-2670	2.5	$7.502
d2.xlarge	4	30.5	3 x 2000	Moderado	Intel Xeon E5-2676	2.4	$0.735
d2.2xlarge	8	61	6 x 2000	Alto	Intel Xeon E5-2676	2.4	$1.470
d2.4xlarge	16	122	12 x 2000	Alto	Intel Xeon E5-2676	2.4	$2.940

Nosotros, para desplegar el primer prototipo de servicio en cloud, contrataremos el servicio que nos proporciona la capa gratuita de AWS, ya que lo consideramos más que suficiente para desplegar nuestro DGE, tal y como explicaremos en otros puntos del proyecto. Se probará con diferentes tipos de instancias para ver el resultado o eficiencia y a cuál se amolda mejor.

IV. Desarrollo del PBI

Como hemos comentado anteriormente, elegiremos varias instancias que se amolden al producto a desplegar. En nuestro caso como el objetivo es hacer un despliegue experimental, empezaremos con una instancia normal, de uso común ya que el producto o la parte del producto a desplegar es asequible según las prestaciones.

La instancia elegida, en un principio será la instancia t2.micro. Es una instancia barata, con un coste de 0.014 $ la hora de uso de esta instancia, aunque hay que decir que, para usuarios nuevos en Amazon, se puede obtener su uso gratuito durante el primer año, siempre que no se pase de las 750 h al mes de uso de la instancia. El almacenamiento EBS (Elastic Block Store) es un tipo de almacenamiento de Amazon que proporciona volúmenes de almacenamiento de nivel de bloques persistentes. Cada volumen de EBS se replica automáticamente dentro de una zona de disponibilidad para protegerle frente a fallos de componentes y para ofrecer alta disponibilidad y durabilidad.

Posteriormente, en un futuro se mejorará esta parte para tener un diseño más exclusivo por parte del cliente de la creación de sus máquinas o instancias, para un mejor rendimiento.

5.1.3 PBI 2: Registrar todas las acciones

I. Acciones a registrar

En este PBI, se persigue que la empresa tenga un control del uso que un cliente le da a su producto y a su vez, se tenga constancia de la cantidad de tiempo y uso que tiene que pagar al proveedor que le facilita ese espacio cloud, para el despliegue del producto. Al acotar la solución final y amoldarse a los tipos de máquinas ya definidos en AWS. Sólo hay que centrarse en guardar el tipo de máquina que se usará, el precio por tiempo que supondrá y la cantidad de tiempo que esa máquina estará en funcionamiento. Partes que registrar:

- El nombre del cliente al que pertenece ese servicio que se está proporcionando
- El tipo de AMI en la que está desplegado, incluido nombre y lugar de alojamiento.
- Las acciones que se hacen con esa instancia:
    - Crear la máquina.
    - Destruir la máquina.
    - Arrancar la máquina.
    - Parar la máquina
    - Arrancar el servicio
    - Parar el servicio
    - Dar de baja el servicio

Cada vez que el cliente haga cualquier tipo de acción con su máquina, al final de cada proceso se guardará en la base de datos esa acción, así, cada movimiento quedará justificado y se podrá hacer un cálculo del tiempo que el cliente ha tenido esa instancia en funcionamiento. Esto nos será útil tanto para costear los gastos y generar la factura al cliente, como para cerciorarnos de los cobros que nos hace la propia plataforma.

Todo lo anterior se resume gracias al cambio en el PBI anterior, ya que si no en vez de tener que almacenar el tipo de AMI y su precio unitario y estandarizado por ese servicio, se tendría un precio que sería la suma de muchas características distintas, porque se podría personalizar todos los aspectos de la máquina y resultaría menos productivo y más complicado. Es una de las posibles mejoras que se le podría aplicar al sistema.

II. Alternativas de registro

Para todo el tema del registro hay varias posibilidades. Estas son alguna de ellas:

1. Si lo que necesita es un servicio de base de datos relacional con una administración mínima, la posibilidad es **Amazon Relational Database Service (RDS).** Es una herramienta que facilita la configuración y funcionamiento de una base de datos relacional en el *cloud.* Ofrece una base de datos económica y redimensionarle que permite elegir entre diferentes motores como son: el propio de *Amazon, Amazon Aurora, Oracle, MySQL o MariaDB* entre otros y proporciona varios beneficios:

    - No es necesario instalar o mantener un software específico de base de datos, ya que utiliza la consola de administración de AWS, la interfaz de comandos y sencillas llamadas a su API para acceder a las capacidades de la base de datos.
    - Se ejecuta en la misma infraestructura de EC2 y por eso dispone de una alta disponibilidad y durabilidad.
    - Puede escalar los recursos informáticos con una sola llamada a la API.
    - Es muy asequible económicamente y se paga según el uso.

2. Si necesita un servicio de base de datos no relacional rápido y de alta escalabilidad, la solución podría ser ***Amazon DynamoDB***. Proporciona una base de datos *NoSQL* administrada que ofrece un desempeño muy rápido, escalable, fiable y de bajo coste entre otras muchas más características. Permite evitar las cargas administrativas que supone tener que utilizar y escalar bases de datos distribuidas y pasa a ser controlado por AWS.

3. Si necesita tener caché en memoria sencilla para operar, la solución podría ser ***Amazon ElastiCache***, que es un servicio administrativo de caché en memoria en la nube que ofrece dos motores de almacenamiento como son: *Memcached* y Redis.

    - *Redis*: es un servicio rápido de caché y almacén de datos en memoria de código abierto y Amazon proporciona la facilidad de uso y potencia de *Redis* con la disponibilidad y fiabilidad necesaria.

- *Memcached*: es un sistema de almacenamiento en caché de objetos de memoria con un uso generalizado. *ElastiCache* actúa como protocolo compatible con *Memcached* y hace que las herramientas populares que utiliza actualmente con los entornos existentes de *Memcached* funcionen.

4. Si necesita un veloz almacén de datos a escala de petabytes, lo más parecido o apropiado puede ser ***Amazon Redshift***, servicio de almacén de datos que permite analizar todos los datos con herramientas de inteligencia empresarial de forma sencilla y rentable.

5. Si necesita una base de datos relacional que pueda administrar por su propia cuenta la solución ya es más complicada, o bien se elige otra instancia de Amazon que esté preparada, es decir o una AMI relacional de EC2 que proporcionan un almacenamiento y computación escalable y allí controlarlo, o bien en tu propia máquina instalar todas las herramientas para alguna base de datos.

En todo caso, en primera instancia se opta por tener un almacenamiento local de las acciones, aunque posteriormente se quiere instalar un almacenamiento más apropiado.

5.2 Sprint 2. Automatización de la máquina

5.2.1 Introducción

El desarrollo del PBI siguiente trataría de que, cuando un cliente contrate un servicio, ejecutar un proceso que auto-contrate la máquina en AWS y registre ese proceso. Todas estas secuencias sí deben ser automáticas, la idea es que para cualquier nuevo cliente la única interacción sea un portal web nuestro, que pueda elegir el tipo de máquina y sus características y que no se preocupe de nada más.

Lo primero que se tiene que tener claro, son las entidades que constarán en el proyecto y las partes que definen cada una de ellas. A modo de resumen, hay que tener en cuenta que para iniciar una instancia se requiere un usuario IAM, un Key Pair y un Security Group, es decir, un perfil de usuario, un par de claves para acceder a las máquinas virtuales y un grupo de seguridad donde se definan los estándares y normas.

Muchas de estas tareas no son automatizables, es decir, se crean y son comunes a todos los usuarios o instancias de nuestra solución, como es el caso del grupo de seguridad, que en un principio no es necesario crear más de un grupo de seguridad. Nos centraremos en este

PBI a mostrar y describir las tareas o hitos que se han hecho previamente a la propia auto contratación de las máquinas virtuales.

5.2.2 Creación de estructura de entidades, clases y relaciones.

En este esquema, hay que diferenciar entre clases y entidades. Programáticamente es muy sencillo, ya que las entidades tendrán una anotación: "*@Entity*" justo encima de la declaración de la clase correspondiente. Esto significa que será una clase persistente, una clase que se guardará en memoria. La clase se guardará como una tabla en nuestra base de datos, además de guardar los atributos como el índice: "*@Id*" o los atributos normales "*@Column*" serán columnas de la tabla o entidad.

El proyecto estará desarrollado utilizando el marco de desarrollo de Spring Boot en el que se distinguen las partes que consta el proyecto: dominio, persistencia, servicio y controlador, ya que se crea una carpeta en la jerarquía del proyecto.

- Domain (dominio). En esta carpeta se almacenarán todas las clases o mejor dicho entidades que se almacenarán en la base de datos.
- Repository (repositorio). En esta carpeta se almacenarán todas las clases con las acciones, métodos de búsqueda y filtros de las clases entidades anteriores, es decir, si se quiere un método específico para buscar por el id, estarán en clases de este tipo. Muchos métodos en nuestro caso, al utilizar *Spring Boot*, no es necesario implementarlos, ya que esta tecnología te los proporciona automáticamente.
- Service (servicio). Es donde realmente se realiza toda la parte de la implementación del servicio. En ella estarán todas las clases con los métodos correspondientes para en nuestro caso, crear la conexión, crear las máquinas, borrarlas… etc
- Controller. Es un paquete intermedio, antes de service, donde se llama a los métodos creados en esas clases. Sirve de pasarela y para el control de errores, en nuestro caso, sirve para controlar todos los posibles retornos en caso de fallo, error o éxito.

Explicaremos concretamente nuestro caso. El paquete de dominio, tenemos tres clases una para cada entidad principal. User que guardará todos los parámetros que correspondan al usuario, VirtualMachine que guardará todas las características elegidas de las máquinas virtuales y Action que guardará todas las acciones que se realice por usuario y máquina. Esta

última es importante, ya que con ella se puede tener un control de todas las acciones que el usuario hace con las máquinas y productos para luego realizar el cálculo del presupuesto.

Por otro lado, están las clases de los repositorios que habrá una por entidad. Todos los repositorios extenderán de JpaRepository para que las opciones principales de buscar por el atributo y demás no haga falta implementarlas, el propio Spring las trae embebidas, pero si se construyen otros métodos de búsqueda como “findByVmId” “deleterByVmId” ...

Los dos últimos paquetes son realmente los paquetes donde está la implementación más pesada. Está el paquete Service que habrá una clase principal donde estarán todos los métodos que nos permitirán crear, borrar, parar y reanudar las máquinas que se creen, es decir, estará todo el servicio completo, toda la funcionalidad y posteriormente en el último de ellos, en la carpeta Controller estarán todos los aspectos de control, desde el manejo de excepciones hasta los mensajes de retorno, devueltos con un logger.

5.2.3 Creación de un cliente IAM

Para la creación de un cliente IAM, primero hay que delimitar que permisos tendrá o a qué podrá acceder un cliente normal, se supone que un cliente normal, solo tiene que tener acceso a la propia instancia y sin tener que programar, tan sólo tener acceso a su máquina Windows, Linux, etc. Esto se puede hacer a través de SSH o programas como PuTTy para establecer esa conexión o creando una aplicación que lo permita.

Se ha establecido 3 Roles de IAM, uno para cada tipo de usuario:

- Administradores: tendrán acceso a todo, desde el acceso a las instancias de EC2 y recursos, como a la gestión de los usuarios IAM, absolutamente a todo. Tipos:
    - AmazonEC2FullAccess: Total acceso a cualquier aspecto de EC2.
    - IAMUserChangePassword: Permisos para poder cambiar las contraseñas de los usuarios. Si un usuario pierde la contraseña para entrar, se tendrá que poner en contacto con la empresa para solucionar el problema.
    - AdministratorAccess: el usuario tiene control total sobre toda la infraestructura creada.
- Usuarios: serán los usuarios normales, que solo tendrán acceso de lectura.
    - ReadOnlyAccess: Proporciona acceso de solo lectura para AWS.

- Usuarios con privilegios: Será un tipo de usuario similar a un administrador con acceso y permiso para todo, pero que no podrá gestionar los usuarios de IAM.
    - PowerUserAccess: Con permisos de administrador, pero sin gestión IAM.

Toda esta parte de la gestión de usuarios estará hecha en primer momento por la empresa, ya que es la única que puede dar privilegios. La propia empresa creará el tipo de usuario correspondiente y se le darán las credenciales a los clientes que lo necesiten.

5.2.4 Creación de un grupo de seguridad (security group)

Un grupo de seguridad actúa como un firewall virtual que controla el tráfico de una o más instancias. Al iniciar una instancia se asocia uno o más grupos de seguridad a la instancia. En ellos, se añaden reglas de seguridad que permiten el tráfico hacia cualquier aplicación o de cualquier aplicación a la instancia. Cuando se habla de reglas de seguridad, nos referimos a reglas con las que administrar el tráfico que entra y sale de una instancia.

Es aconsejable que todo este tipo de gestiones las realice internamente la propia empresa y que no queden a las vistas o se puedan modificar por los clientes. Habría un administrador que se ocupase de dar de alta a esos clientes y establecer las normas y los permisos necesarios, ya que podría provocar conflicto un mal uso de estos permisos.

5.3 Sprint 3. Creación y Destrucción de la máquina

5.3.1 Introducción

En esta iteración ya se hacen las operaciones propias con AWS, se hacen las dos primeras acciones importantes, la creación y destrucción de las máquinas. Todas las acciones se hacen a través de la API que facilita Amazon, ya que uno de los motivos para elegir AWS es que tenían APIs de diferentes lenguajes, para Java u otras como para Pyton, C++ etc.

Además, hay que decir que se ha utilizado una API concreta en la que nos hemos basado a la hora de establecer la conexión con AWS, es una API gratuita y la licencia nos permitía utilizarla y modificarla sin ningún problema. La API utilizada se llama Typica[25], cuyo código estaba accesible en la propia página de Amazon.

[25] https://aws.amazon.com/code/DevPay/770

Typica consigue trabajar con uno de los productos esenciales de AWS como es EC2 que nos permite crear esas instancias o máquinas virtuales con las que nosotros trabajaremos.

Antes de crear cualquier tipo de máquina, lo primero y necesario es crear una conexión con Amazon a través también de la propia API. Para ello hay un método que introduciendo el id del cliente, nombre, región de Amazon y la password, crearía la conexión. Concretamente en términos de AWS son los llamados accessKeyId, nameUser, regionUrl y secretAccessKey, con todo eso se crea un objeto Jec2 que almacenará todos esos parámetros y conseguirá la conexión con los servidores de Amazon.

5.3.2 Creación de máquinas

Después de crear la conexión como hemos explicado anteriormente. Hay varios parámetros que son necesarios para la creación de máquinas y otros no, por eso habrá más de un método con el mismo nombre, pero distinta firma, distintos parámetros de entradas. Explicaremos los parámetros utilizados del método con todos los parámetros posibles para tener una mayor personificación.

Método complete:

```
public List<VirtualMachine> createVM(String accessKeyId, String
secretAccessKey, String regionUrl, String imageId, int minCount,
int maxCount, List<String> groups, String userData, String
keyName, boolean publicAddr, InstanceType instanceType, String
availabilityZone, String kernelId, String ramdiskId,
List<BlockDeviceMapping> blockDeviceMappings);
```

Los parámetros de este método son:

- **accessKeyId**: Id de la cuenta con la que se crea la máquina virtual en cuestión.
- **secretAccessKey**: Password de esa cuenta utilizada, facilitada por el propio servicio de Amazon, ya que es una contraseña con un número de caracteres concreto.
- **regionUrl**: Región elegida entre todas las que hay para almacenar en sus servidores la instancia elegida. Según la región se tendrá un precio y prestaciones diferentes.
- **imageId**: Nombre o id de la máquina virtual a exportar, todas las máquinas tendrán un id que será el que se utilizará. Si se quiere crear una máquina virtual con un sistema operativo concreto que no está entre las que oferta la propia plataforma de Amazon, se tendrá que subir una imagen e instalar previamente.

- **minCount**: Serían las instancias mínimas en el caso de que el servicio disminuya en rendimiento. Nosotros tanto para el mínimo como para el máximo utilizaremos 1, porque creemos que con una sola instancia es más que suficiente para cualquier producto de los ofertados. Aunque se pueden hacer pruebas más adelante.
- **maxCount**: Son las instancias máximas que el sistema tiene para incrementar.
- **groupSet**: Grupo de seguridad concreto, que previamente se tiene que tener establecido. Indican las normas y estándares a seguir como puertos DHCP, TCP etc
- **keyName**: Nombre de la clave que se utilizará para la creación de la máquina, elegida por el usuario y creada anteriormente.
- **availabilityZone**: Zona de disponibilidad de Amazon elegida
- **kernelId**: Id del kernel indicado. Se puede dejar null y no habría ningún problema, se establecería uno por defecto. No es necesario y hay un método más sencillo sin él.
- **ramdiskId**: Id de la RAM, al igula que el kernel se puede establecer null.
- **instanceType**: Tipo de instancia elegida, según precio y prestaciones. Desde la t2.micro que es gratuita a cualquier otra instancia específica de Amazon.
- **publicAddr**: Booleano que en caso positivo establece un direccionamiento público.
- **blockDeviceMappins**: Ofrece asignaciones de nombres virtuales a dispositivos.

Todos estos métodos que se especifican con diferentes opciones, unas opcionales y otras fijas se pueden ver en la especificación de la propia API:
http://typica.s3.amazonaws.com/com/xerox/amazonws/ec2/class-use/ReservationDescription.html

Después de introducir todos estos parámetros AWS creará una máquina virtual con esas características, pero tardará un tiempo, muy poco en crearse y mientras tanto no podrá ser utilizada esa máquina virtual.

5.3.3 Destrucción de máquinas

Al igual que ocurre con la creación de las máquinas, también se hace a través de un método específico, pasaremos a explicar los parámetros para llevarlo a cabo:

```
public InstanceStateChangeDescription terminateVM(String
accessKeyId, String secretAccessKey, String regionUrl, String
virtualmachineId);
```

Los parámetros de este método son:

- ***accessKeyId***: Id de la cuenta a la que pertenece la instancia
- ***secretAccessKey***: Password de la cuenta a la que pertenece la instancia
- ***regionUrl***: Region en la que está ubicada la instancia
- ***virtualmachineId***: Id de la instancia. El id de la instancia se crea automáticamente cuando la máquina virtual ha sido creada. Se podría ver por medio de la plataforma o incluso instanciando y mostrando todas las instancias que están operativas.

Hay que decir que Amazon dispone de una interfaz gráfica en la que se puede observar todo este tipo de valores y los cambios que se producen al hacer cualquier operación de este tipo.

5.4 Sprint 4. Reanudar y parar la máquina

5.4.1 Introducción

Nuevamente en esta iteración se vuelve a utilizar conceptos de la API anterior. Pero tenemos que hablar de las restricciones que en el Sprint anterior no se han hablado.

Para la creación de las instancias es evidente que no se puede tener dos instancias iguales, pero esto es imposible que se pueda conceder, ya que todas las instancias creadas por AWS, tendrán un diferente id, aunque tengan la misma configuración o AMI instalada.

Por parte de la destrucción o eliminación de la máquina, la única restricción posible es que esta instancia o máquina virtual esté previamente creada, porque si no esta creada, se mandará un mensaje informando del error.

Por último, por parte de reanudar y parar, ocurre el mismo caso que en la destrucción, si no se tiene ninguna máquina virtual creada con ese id no se podrá eliminar, parar ni reanudar. Todo ello viene definido porque para eliminar, parar y reanudar una instancia o máquina virtual se tiene que introducir el id de la máquina que se quiere operar, por eso si no es válida, el propio servicio de AWS rechazará la respuesta y en vez de devolver un OK o un mensaje de éxito, devolvería un error.

5.4.2 Parada y Reanudación de máquina

Mostraremos las dos cabeceras de los métodos, tanto el de parar como el de reanudar, ya que es muy similar, concretamente comparten atributos:

Parar máquina virtual:

```
public InstanceStateChangeDescription stopVM(String accessKeyId,
String secretAccessKey, String regionUrl, String virtualmachineId) {
```

Reanudar máquina virtual:

```
public InstanceStateChangeDescription startVM(String accessKeyId,
String secretAccessKey, String regionUrl, String virtualmachineId) {
```

Los parámetros de estos métodos son:

- ***accessKeyId***: Id de la cuenta a la que pertenece la instancia
- ***secretAccessKey***: Password de la cuenta a la que pertenece la instancia
- ***regionUrl***: Region en la que está ubicada la instancia
- ***virtualmachineId***: Id de la instancia. El id de la instancia se crea automáticamente cuando la máquina virtual ha sido creada. Se podría ver por medio de la plataforma o incluso instanciando y mostrando todas las instancias que están operativas.

5.5 Sprint 5. Creación de WebApp y login del sistema

5.5.1 Introducción

Esta parte del proyecto se ha desarrollado fuera de la empresa. La intención de la empresa y por eso se ha expresado así en la planificación anteriormente era que la plataforma se integrase en la web de la empresa en la que se contratan los productos on-premise existentes hasta el momento. Pero al no llegar a realizar esta fase en el tiempo de desarrollo, se ha creado una interfaz en la que mostrar todo este proceso de manipulación de las instancias/máquinas virtuales.

La intención de esta parte del proyecto es poder mostrar de forma visual y rápida la destreza del proyecto y que todas las operaciones que se hacen son correctas. Anteriormente, sin esta parte del proyecto, la única forma de evaluar el funcionamiento del mismo era a través de test unitarios. Toda esta parte de test unitarios están realizados en el proyecto y han sido la primera manera de cerciorarnos que todo está bien. El proyecto pasa todos los test y se puede observar que todas las operaciones se realizan correctamente.

Otra forma de evaluar el funcionamiento de la plataforma/servicio es desplegarlo, operar y evaluar en la herramienta gráfica que tiene AWS, en la consola de administración. En esta herramienta se puede observar si se crea, elimina, para o reanudan las máquinas realizadas con nuestra plataforma. A continuación, se detallarán algunas cuestiones de esta herramienta que nos proporciona AWS:

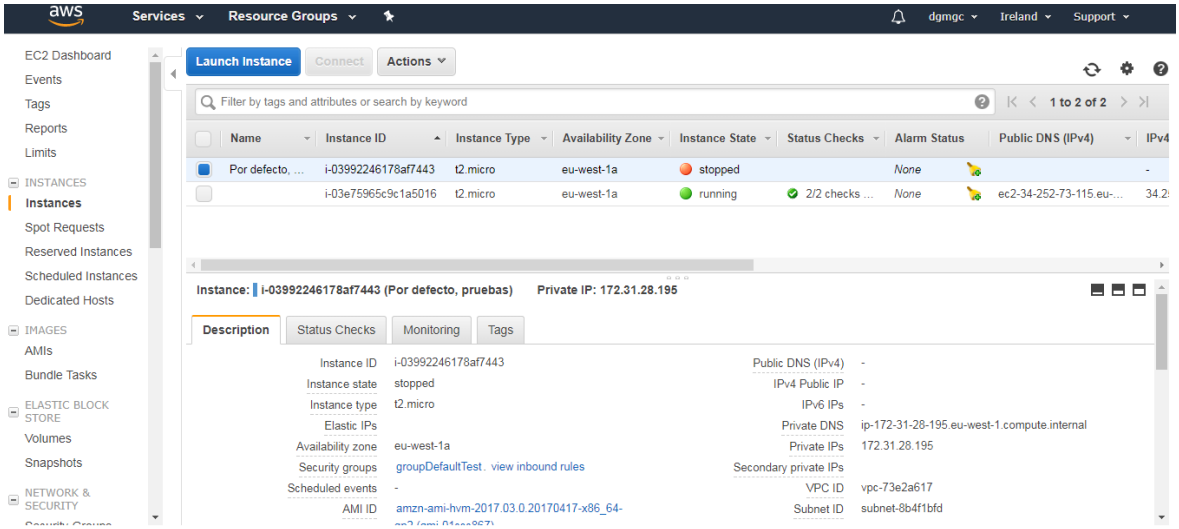

Figura 26:Consola de administración de AWS

En grandes rasgos, en la parte izquierda se encuentran multitud de secciones en las que se puede observar información relevante de todos los procesos abiertos de AWS, tanto las propias instancias, AMIs propias, perfiles IAM, grupos de seguridad… En la parte central de la herramienta pueden verse que en esa captura hay dos máquinas creadas (instancias para AWS). Una de ella está parada y otra arrancada y además se puede observar algunas características básicas. Para más información si se pincha sobre cualquiera de ellas se abre la información más información en la parte inferior, como se puede observar en la captura. Además, se pueden ver informaciones de estado y monitorización de la misma.

Todas las opciones que se pueden hacer con nuestra plataforma se pueden hacer en esta herramienta, pero no tendría la exclusividad para utilizarse sólo por nuestra empresa y no se podría tener tantas *password* como usuarios para que pueda ser viable. Además, hay que decir que nuestra plataforma, es un prototipo o primera plataforma funcional, no solo se crearán máquinas virtuales, sino que se instalarán esos productos en ellas y en esta herramienta, todo eso no se puede hacer.

5.5.2 Explicación de la interfaz de la web

La plataforma tendrá una interfaz sencilla. La primera parte será una simple interfaz con el login del usuario. En ella, se pedirá el usuario y su contraseña. Estas credenciales serán aportadas por la empresa, ya que en un principio se tienen que realizar otros trámites y procesos como la creación de un grupo de seguridad para las instancias de ese cliente, un cliente IAM determinado, etcétera.

Por lo tanto, la interfaz se compone de dos campos para introducir texto y un botón que se accionará cuando ambos campos estén correctamente rellenados. Además, en todas las interfaces tendremos el nombre de la aplicación en la parte superior.

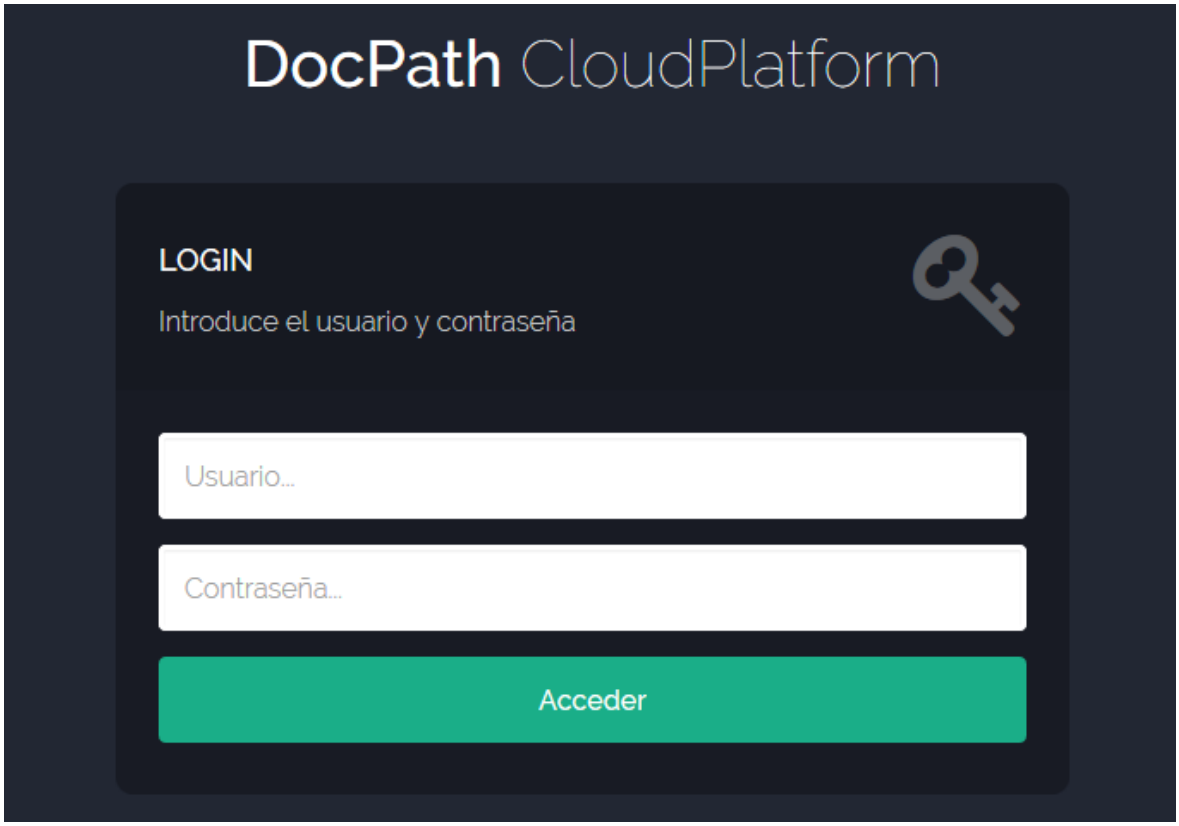

Figura 27:Interfaz de login

Si la respuesta es positiva, justo después, se abrirá la ventana principal donde se pueden observar los cuatro botones principales que corresponden con las cuatro operaciones que se pueden realizar con la plataforma, un botón para: crear máquina, destruir máquina, parar máquina y reanudar máquina. Además, se dispondrá información referente al usuario que se acaba de logear como es su nombre, credenciales de AWS, es decir, credenciales que le tendrá que aportar al servicio como son el *accessKeyId* y el *SecretAccessKey*. Estas

credenciales en un principio no deberían de ser visibles o accesibles de esta manera tan sencilla, pero como es una primera versión se dejan para hacer más sencillo el proceso, posteriormente se dejarán en un sitio menos visible e inaccesible. Además, se podrá ver la última vez que se conectó al servicio. Posteriormente si se quiere salir y logear nuevamente con otro usuario, también dispone de la opción en la parte inferior.

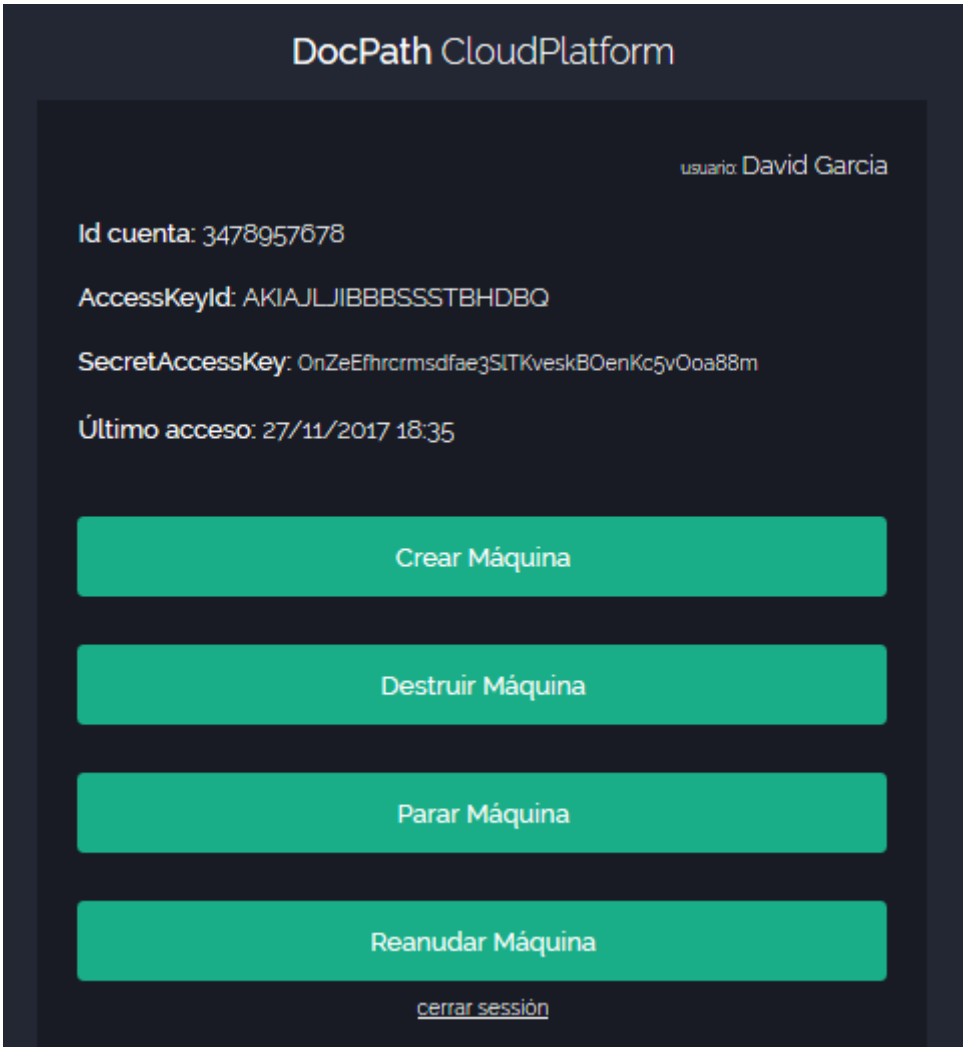

Figura 28:Interfaz principal

Al pulsar cada uno de los botones principales anteriores, se mostrarían las posibles acciones y pediría la información necesaria para crear, destruir, parar o reanudar una máquina. Iremos paso por paso.

La primera que mostramos es la interfaz de creación. En ella se pedirán los datos antes detallados en el propio método y son los parámetros que pueden variar a la hora de crear una máquina virtual. Los parámetros que se piden son: el *accessKeyId* y el

secretAccessKey que se piden en todas las acciones, además pedirán la región donde se quiere crear la máquina virtual, la imagen que se quiere instalar, es decir, si se quiere una máquina con Windows, Linux o cualquier otro sistema operativo o configuración y el tipo de instancia que se quiere contratar. Cuando se tenga todo correcto, el usuario pulsaría el botón de crear y el servicio le devolverá información sobre si se ha tenido algún problema o por el contrario, la instancia se ha creado satisfactoriamente.

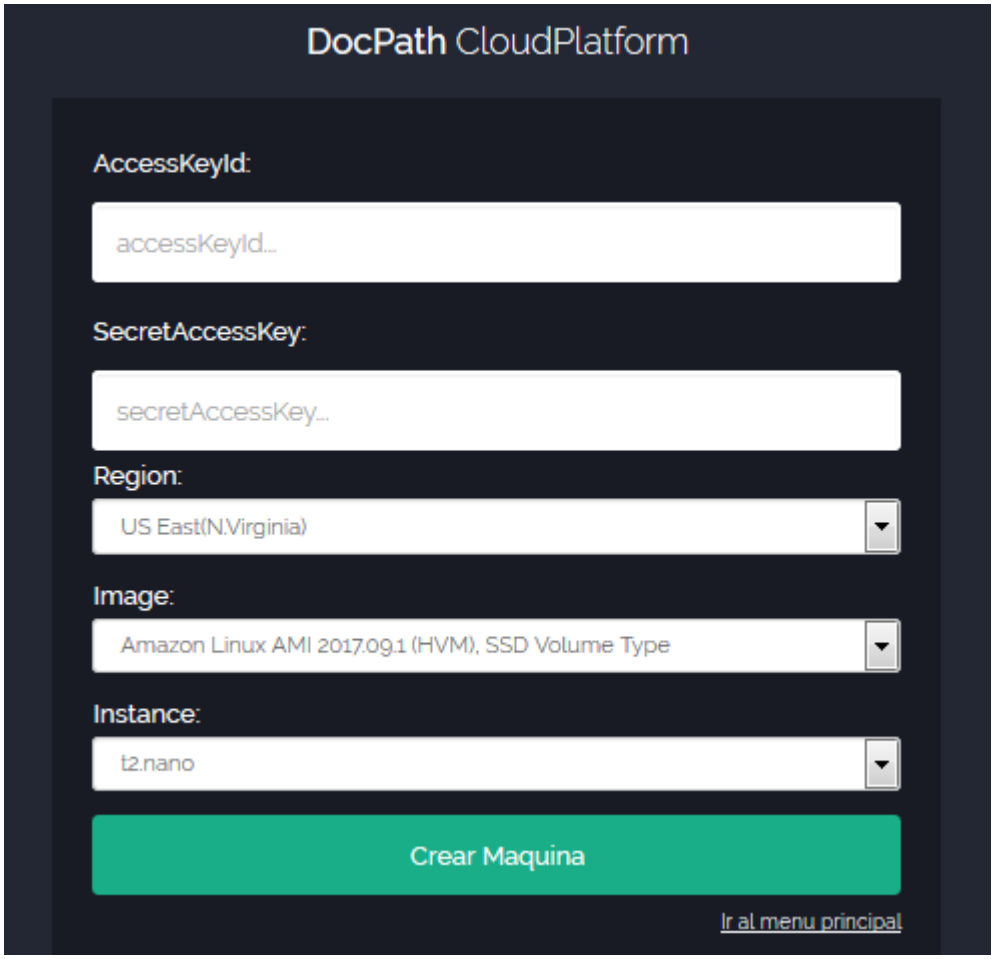

Figura 29:Interfaz de creación

La segunda es la operación de destruir la máquina, por parte de esta operación, los tres datos primeros son absolutamente los mismos, accessKeyId, secretAccessKey y región, lo único que varía es que se tiene que introducir el Id de la máquina que se quiere destruir. También tiene la opción de eliminar todas las máquinas asociadas a ese usuario.

Es cierto que no es una forma muy eficiente la de introducir un id de una máquina, porque tendría que meterlo el usuario de memoria y no es operativo, pero como es una

versión funcional, no es necesario hacerlo de una forma más operativa, lo que se necesita es saber que el sistema funciona y se crea la acción correctamente.

Una posible forma de implementar esta parte es con un select en el que se carguen las máquinas que actualmente el usuario tiene operativas, es decir, en funcionamiento o paradas, para poder elegir la máquina que se quiera sin necesidad de meter su id.

También hay que decir que la opción de eliminar o destruir todas, no sé si es del todo operativo, ya que cualquier error provocaría un caos. Se debería de tener alguna medida para asegurarnos que no se borran todas en caso de equivocarnos. Aun así, se tiene un ligero periodo de tiempo de treinta minutos en el que el usuario puede volver a restablecer todas las instancias y recuperarse de su error.

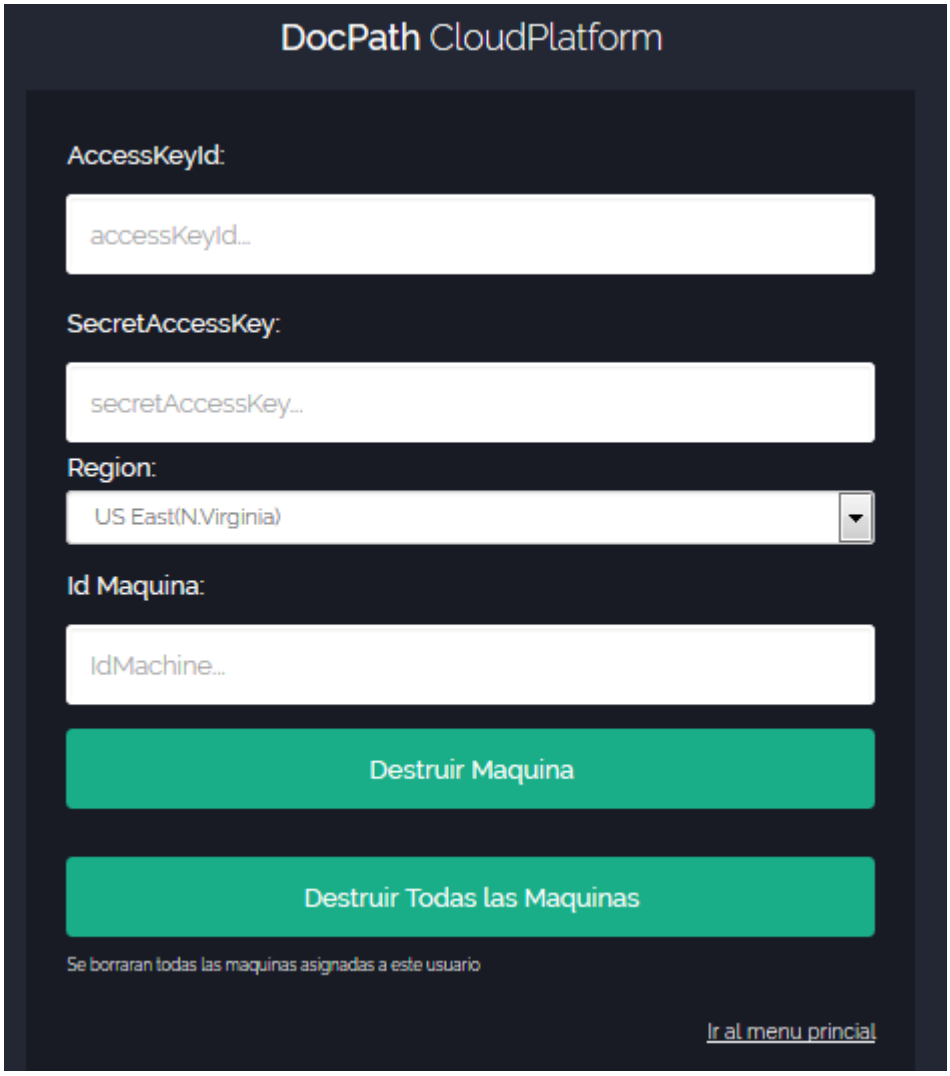

Figura 30:Interfaz de destrucción

Por último, las dos interfaces últimas son muy similares, tan solo cambia la función y el tipo de operación a realizar, pero los parámetros a introducir son los mismos, luego la interfaz no cambia. En ella estarían los mismos cuatro primeros objetos, las dos cajas para introducir el texto del accessKeyId y del secretAccessKey, el selector de región y la respectiva caja para introducir el Id de la máquina a la que se quiere realizar la operación.

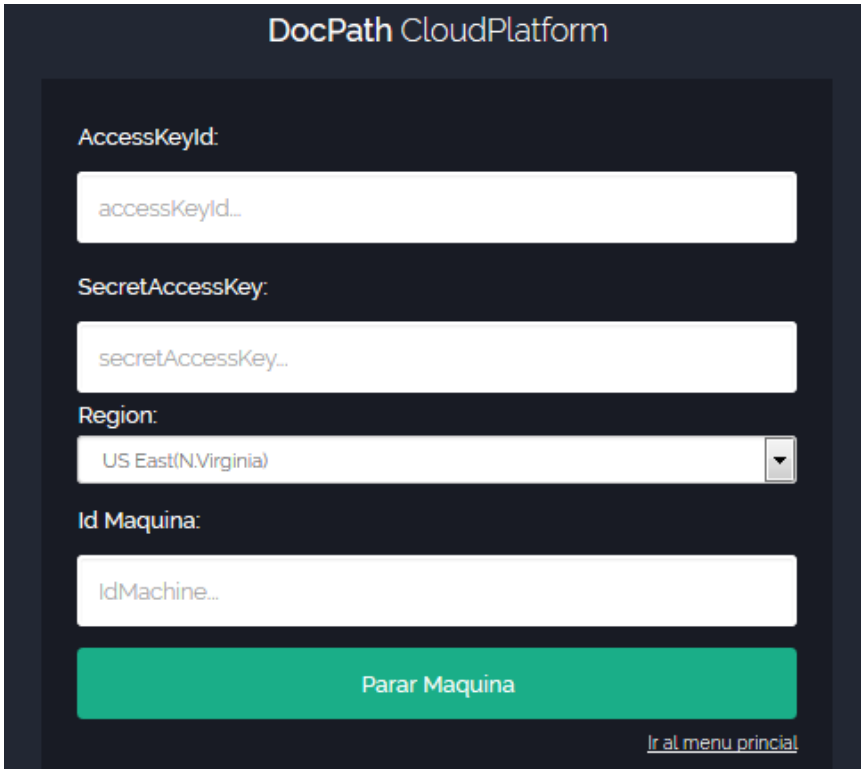

Figura 31:Interfaz de parada

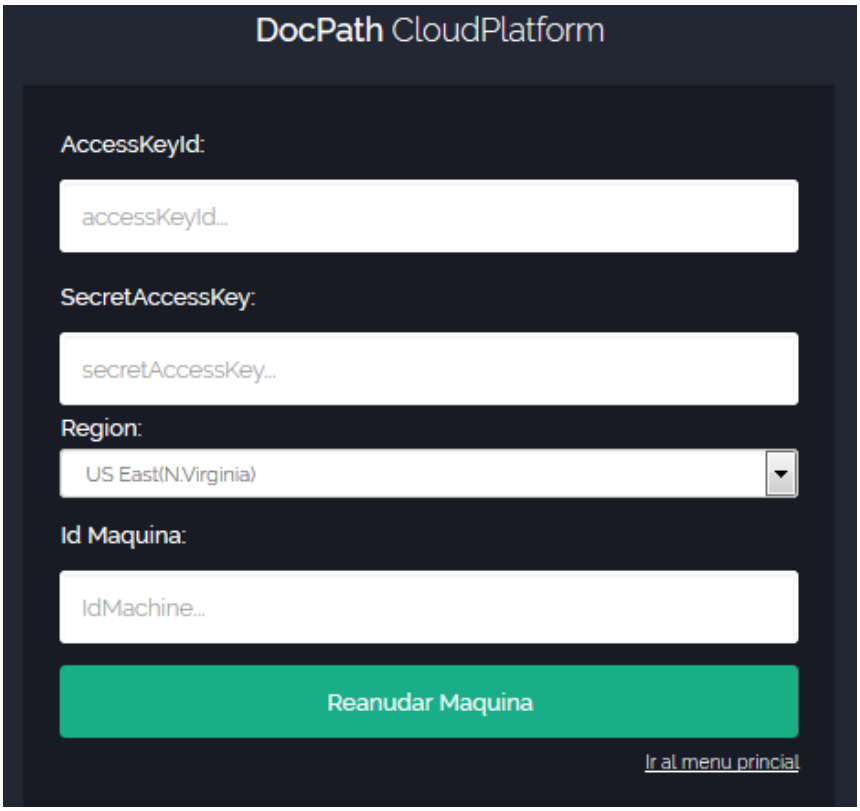

Figura 32:Interfaz de reanudación

Hay que decir que algunos parámetros, sobre todo en la parte de la creación de la máquina son estáticos a la hora de elaborar la petición para AWS, es decir, además de los aspectos que se piden en la interfaz, hay otros aspectos como el grupo de seguridad que en un principio no variará, pero en un futuro se estudiarán si pueden ser susceptibles a cambios o no, ya que es una de las cosas que se tiene que evaluar.

Capítulo 6

Conclusiones

En este capítulo se exponen las conclusiones extraídas de la realización del proyecto, describiendo el grado de consecución de los objetivos establecidos y posibles ampliaciones y trabajos futuros. Por último, se incluye una valoración personal.

6.1 Objetivos alcanzados

Al principio, antes de empezar con el desarrollo del proyecto, en la fase de Inception, se evaluaban y marcaban unos objetivos. Los objetivos se priorizaron según el método MoSCoW y con el paso del tiempo y las iteraciones, ese criterio ha ido variando. A continuación, se mostrará esa planificación primera y se evaluará los objetivos que se han llevado a cabo o no. En el siguiente listado se anotarán los requisitos según la planificación inicial y si ha sido realizado completamente:

Categoría Must:

1.	Registrar todas las acciones, cliente y plataforma.	**Realizado**
2.	Analizar los elementos facturables	**Realizado**
3.	Automatizar la contratación de la máquina.	**Realizado**
4.	Crear scripts para crear y destruir la máquina.	**Realizado**
5.	Crear scripts para parar y arrancar la máquina.	**Realizado**

Categoría Should:

6.	Crear WebApp	**Realizado**
7.	Dar de baja servicios	Pendiente
8.	Arrancar y parar servicios	Pendiente
9.	Mostrar lista de servicios contratados	Pendiente
10.	Crear pantalla de configuración para aspectos técnicos	Pendiente
11.	Crear scripts para configuración automática.	Pendiente

Categoría Could:

12.	Downgrade y Upgrade de la máquina	Pendiente
13.	Almacenamiento de la máquina.	Pendiente

Categoría Would:

14.	Login de usuarios	**Realizado**
15.	Mostrar lista de servicios disponibles	Pendiente

De forma esquemática y según el diagrama de red que se propuso en un principio se pueden ver también las tareas realizadas, pintadas de color violeta:

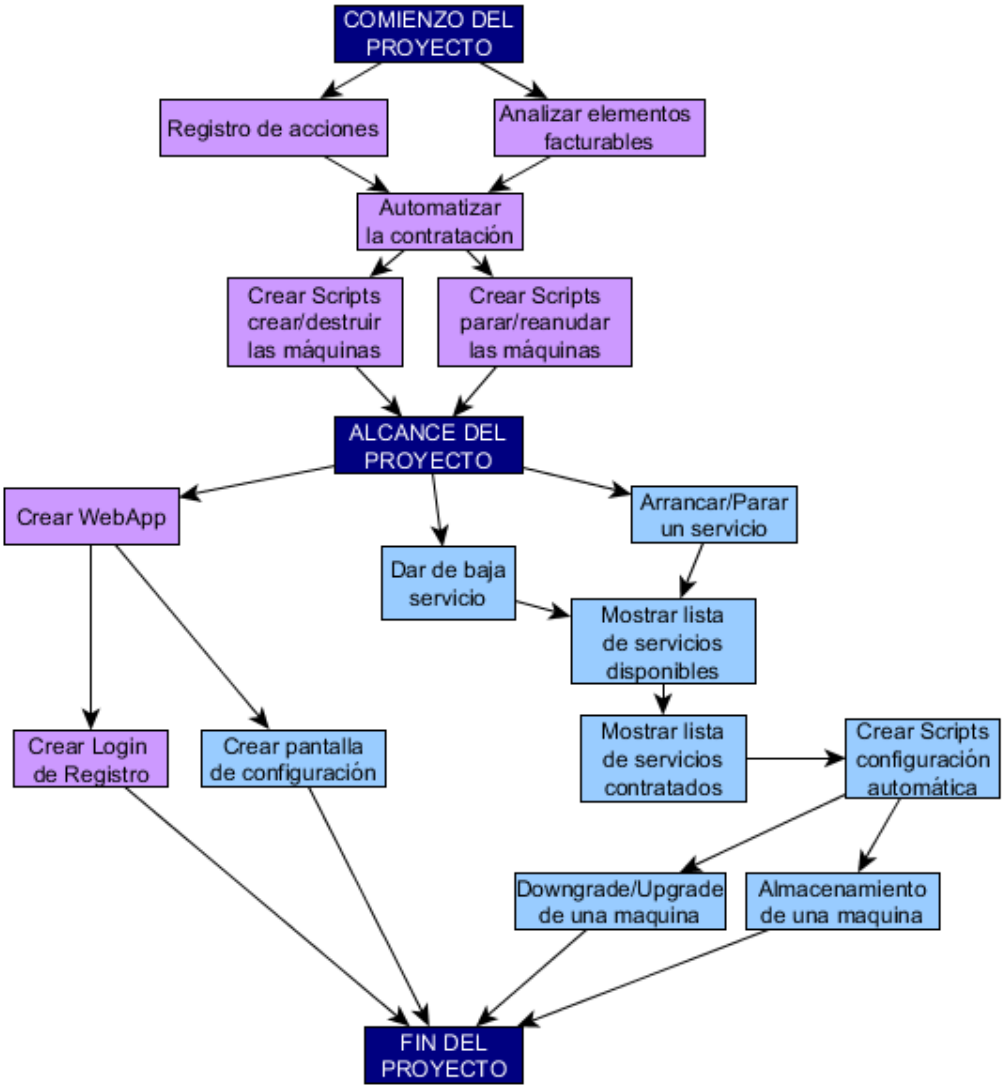

Figura 33: Gráfico con las tareas realizadas

Hay que decir que, al utilizar una metodología ágil, la priorización de requisitos se ha ido cambiando en relación con el tiempo, que ha sido uno de los aspectos que más nos ha delimitado el desarrollo del mismo. El alcance del proyecto ha quedado cubierto con todas las tareas realizadas. Además, se ha realizado la parte de la interfaz web para mostrar todo ese desarrollo de una forma más visual y el login de usuarios, clasificado en la categoría Would, se ha priorizado por delante de otras tareas de la categoría Should. En un principio se estableció en una parte muy inferior de la escala de prioridades, pero al no integrarse en "Magallanes" y ser una aplicación independiente por ahora, es necesario tener un login, aunque sea sencillo, para preservar la seguridad del usuario.

El resultado final del proyecto es incompleto, pero como se dijo en la planificación, es un proyecto de mucha más envergadura y para más tiempo de desarrollo y nosotros hemos dispuesto de un par de meses de desarrollo. Las metas estaban fijadas y el alcance del proyecto se ha cumplido, pero era realmente difícil que los quince requisitos u objetivos se llevasen a cabo. El gran objetivo está cumplido, que es el de tener una plataforma o servicio que pueda aprovisionar al cliente y manipular las instancias de AWS con ella.

6.2 Ampliaciones y trabajos futuros

Referente a las ampliaciones y trabajos futuros, lo primero es terminar lo que uno empezó, las ampliaciones es seguir con el plan de ruta establecido:

- Lo siguiente en realizar es la automatización de la instalación y despliegue de los productos DocPath en la plataforma, para que una vez que se tenga automatizada la contratación de la máquina se despliegue del producto. Ya se han hecho pasos sobre ello y consiste en el despliegue de un war con el producto mediante scripts, algunos de esos scripts estaban hechos ya, pero al no tener terminado el PBI no se puede enseñar el resultado final.
- También se podría crear una pantalla de configuración para los aspectos más técnicos y crear scripts para la autoconfiguración y aprovisamiento automático de las máquinas virtuales. Son requisitos que estaban previstos en el propio proyecto.
- Otra de las características propuestas es la de aumentar o disminuir la potencia o recursos de la máquina elegida para soportar un crecimiento o disminución del volumen de trabajo del cliente. Actualmente solo se puede parar y reanudar una máquina y cobrar según el uso que se le da, es decir, cuando está encendida. Pero en un futuro se quiere añadir esa funcionalidad de poder aumentar la potencia de la máquina en vez de contratar alguna más.
- La última característica que estaba contemplada en el proyecto actual sería la de contratar un almacenamiento secundario de la máquina, es decir, elegir si se quiere tener un almacenamiento interno añadido para guardar los documentos generados con cualquier producto de DocPath.
- Mejorar el sistema de configuraciones que se tiene ahora mismo, es decir, ahora mismo nos ceñimos a configuraciones de productos o AMIs ya creadas por parte de AWS, con una potencia, almacenamiento y rendimiento establecido y según las

necesidades del cliente se contrata un tipo u otro. La mejora sería poder elegir y personalizar completamente la instancia y personalizar todos los elementos tú mismo. Quizás esta alternativa se debería de acotar de algún modo, pero si por lo menos, sería muy eficiente tener alguna que otra configuración prefijada para todo este tipo de circunstancias y no solo ceñirnos a las que se ofertan en AWS y así tener una amplia variedad de instancias para todos los públicos.

6.3 Valoración personal

La realización de este proyecto ha sido un reto, ya que es la primera experiencia laboral que he tenido y es la primera vez que realmente he tenido la oportunidad de aprender lo que es este sector profesional que he estudiado. He tenido que adaptarme a un entorno empresarial y profesional desconocido totalmente, donde he aprendido a utilizar herramientas y a aplicar diferentes marcos de trabajo, técnicas y metodologías que antes no había oído hablar. Ha sido la primera vez que he tenido contacto con profesionales y he afrontado una necesidad real de un negocio.

A nivel formativo, ha sido una experiencia muy positiva, que creo que todo el mundo debería realizar, ya que realmente en las prácticas o trabajos no se tiene la necesidad de "buscarte la vida" sobre algunos conceptos que no se saben y no vienen en el guion, todo esto te enseña a trabajar sobre la necesidad de colaborar, trabajar en grupo y saber sobreponerse a los problemas generados.

Hay muchos conceptos teóricos que se ven de forma teórica, que estaban casi olvidados y se ha tenido la necesidad de darles una vuelta y volver a recordarlos.

Mas allá de los resultados de este proyecto, me siento lo primero muy agradecido por poder tener la oportunidad de esta beca de formación, por el esfuerzo realizado personalmente y por parte de mis tutores y por la oportunidad que me ha brindado DocPath de poder formar parte de su trabajo. Es la primera experiencia real con la que me he dado cuenta del perfil de trabajo que se puede desarrollar y el día a día de esta profesión.

Capítulo 7

Bibliografía

[1] Luis Javier Peris. « *DocPath Cloud Platform e Inception*.» Miguelturra. 2016

[2] DocPath® Document Solutions S.L. *Business Pro Suite*. Miguelturra: 2009.

[3] Michael J. Kavis: *Architecting the Cloud: Design Decisions for Cloud Computing Service Models.*

[4] A. Bahga y V. Madisetti, *Cloud Computing: A Hands-On Approach*. CreateSpace Independent Publishing Platform, 2013.

[5] Dai Clegg, *Case Method Fast-Track: A RAD Approach*. Wokingham: Addison-Wesley, 1994. (MoSCoW)

[6] Dimes, T. And M.Jiménez: *Conceptos Básicos de Scrum: Desarrollo de Software Agile y Manejo de Proyectos Agile.* Babelcube Incorporated, 2015.

[7] Stellman, A. and J. Greene: Learning Agile: Understanding Scrum, XP, Lean, and Kanban. O'Reilly Media, 1st edition, 2014

[8] Guia sencilla de Github: rogerdudler.github.io/git-guide/index.es.html

[9] Ryan Smith: *Scrum Guide: Agile Project Management Guide for Scrum Master and Software Development Team* . CreateSpace Independent Publishing Platform, 2016

[10] K. Beck, *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, 2000.

[11] Mark Lines and Scott W.Ambler: *Introduction to Disciplined Agile Delivery: A Small Agile Team's Journey from Scrum to Continuous Delivery.* CreateSpace Independent Publishing Platform, 2015

[12] Monte, J.: *Implantar scrum con éxito*. Editorial UOC, 2016

[13] Linwood, J. & D. Minter: *Beginning Hibernate*. Apress, 2010.

[14] The Apache Software Foundation: *Apache Tomcat 7 User Guide*. Fultus Corporation, 2011

[15] Macario Polo. *Apuntes y documentación de prácticas de Tecnología y Sistemas* Web para la parte de la web. UCLM 2016

[16] Flanagan, D.: *JavaScript: The Definitive Guide: The Definitive Guide.* O'Reilly

[17] Pterneas, V.: *Getting Started with HTML5 WebSocket Programming*. 2013.

[18] L. V. Lancker, *HTML5 y CSS3: Domine los estándares de las aplicaciones Web [2ª edición]*. Ediciones ENI, 2013.

[19] Butcher, M.: *Drupal 6 JavaScript and jQuery*. Packt Publishing, 2009

[20] M.A. Alvarez *Manual de jQuery online de desarrolloweb.com*: www.cav.jovenclub.cu/comunidad/datos/descargas/jQuery.pdf

[21] Juan Diego Gauchat: *El gran libro de HTML5, CSS3 y Javascript*. Marcombo

[22] Libro Web de bootstrap: librosweb.es/libro/bootstrap_3/

[23] Beck, Kent: *Test-Driven Development*. By example, Addison-Wesley, 2003.

[24] Petar Tachiev, Felipe Leme, Vicent Massol: *Junit in action*.

ANEXO A : Modelos del proyecto

En este anexo se muestran todos los bocetos de los modelos de dominio, procesos e interfaz de usuario del proyecto. Tan solo son bocetos, y puede diferir mucho el resultado final de todo lo anterior expuesto aquí. Se podría decir, que la finalidad para la que se han hecho estos diagramas es la de comprender mejor la interacción del usuario con la aplicación y para que el desarrollador y el *Product Owner* acuerden un flujo de utilización de la plataforma.

MODELOS DE PROCESOS:

- **PBI 1: Registrar todas las acciones**

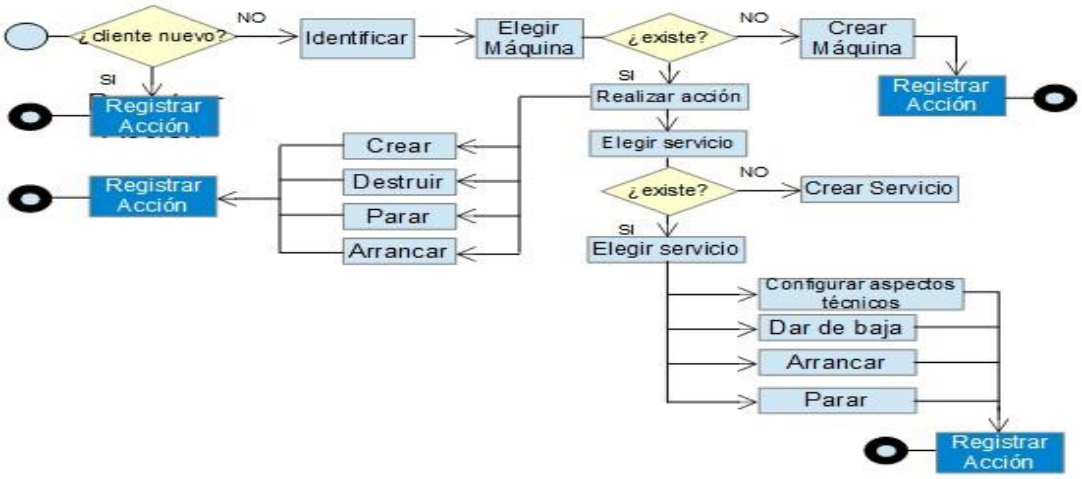

Figura 34:Diagrama de procesos: PBI 1

- **PBI 2: Analizar los elementos facturables**

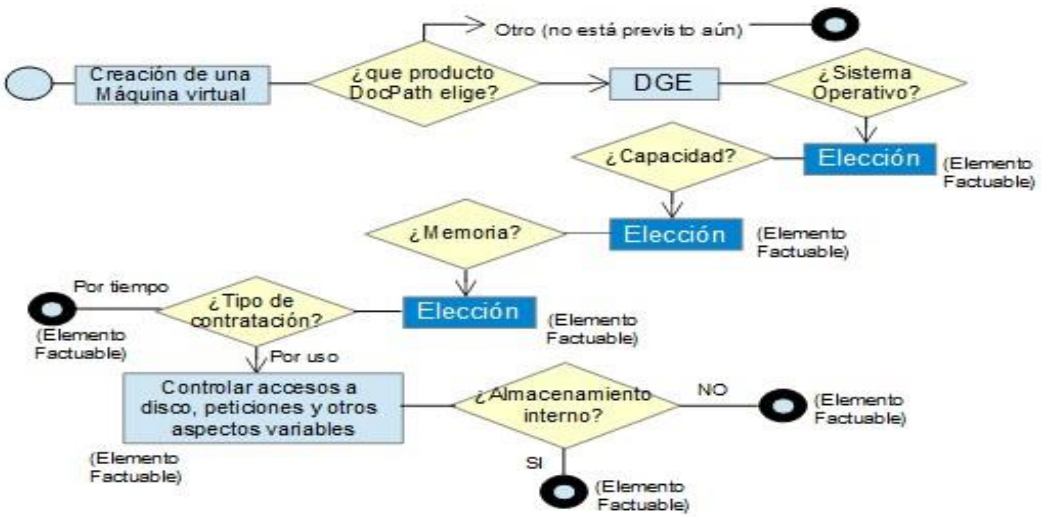

Figura 35:Diagrama de procesos: PBI 2

Respecto al diagrama o modelo anterior, hay que destacar que todo lo que se pide en la creación de la máquina virtual o mejor dicho AMI en Amazon EC2, son elementos facturables. En caso de la contratación hay dos posibilidades: por tiempo y por uso, si es por tiempo es aplicar una tarifa, pero en el otro caso hay que tener en cuenta muchos más parámetros como la cantidad de accesos a disco. Quizás la mejor opción es tener productos personalizados que permitan una tarifa, al estilo de los teléfonos móviles, pero eso no sería contratación por uso.

- **PBI 3: Automatización de contratación de la máquina**

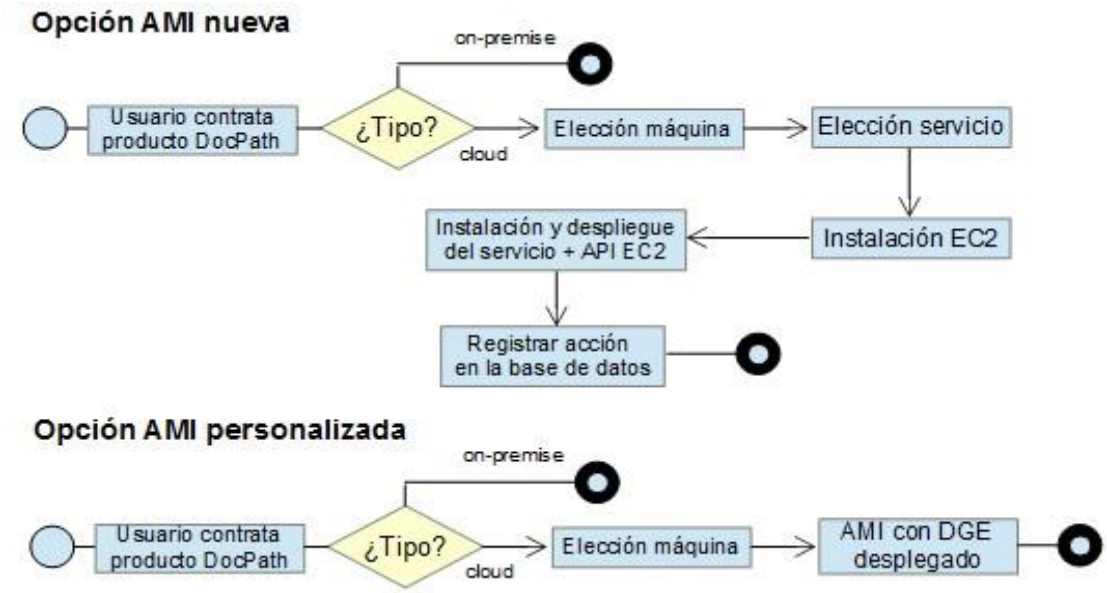

Figura 36::Diagrama de procesos: PBI 3

Hay que destacar algunos aspectos relacionados con el diagrama anterior, hay dos posibilidades reales de hacer la automatización de la contratación de la imagen y varían en un aspecto importante. Existe la posibilidad de crear la contratación de la máquina, realmente la contratación de la AMI que se desplegará en una instancia de EC2, y esa AMI puede ser una nueva o una creada y personalizada antes. A modo personal, creo que la mejor opción es la contratación de una AMI personalizada que tenga todo actualizado e instalado desde un principio y con ello nos ahorraríamos una serie de pasos, aunque también existe la opción de crear esa AMI nueva, desplegarla en una instancia de EC2 y en ella instalar todo el servicio correspondiente. Al menos habría que crear un script que cree esa AMI y albergarla en una instancia. En la opción de la AMI personalizada también habría que registrar la operación.

- **PBI 4: *Script* crear/destruir máquina**

En estos diagramas o modelos, obviamos la parte de identificar al cliente y el último paso de registrar todas las acciones en la base de datos, pero hay que tener en cuenta que todo ello también se hace, queda explícito.

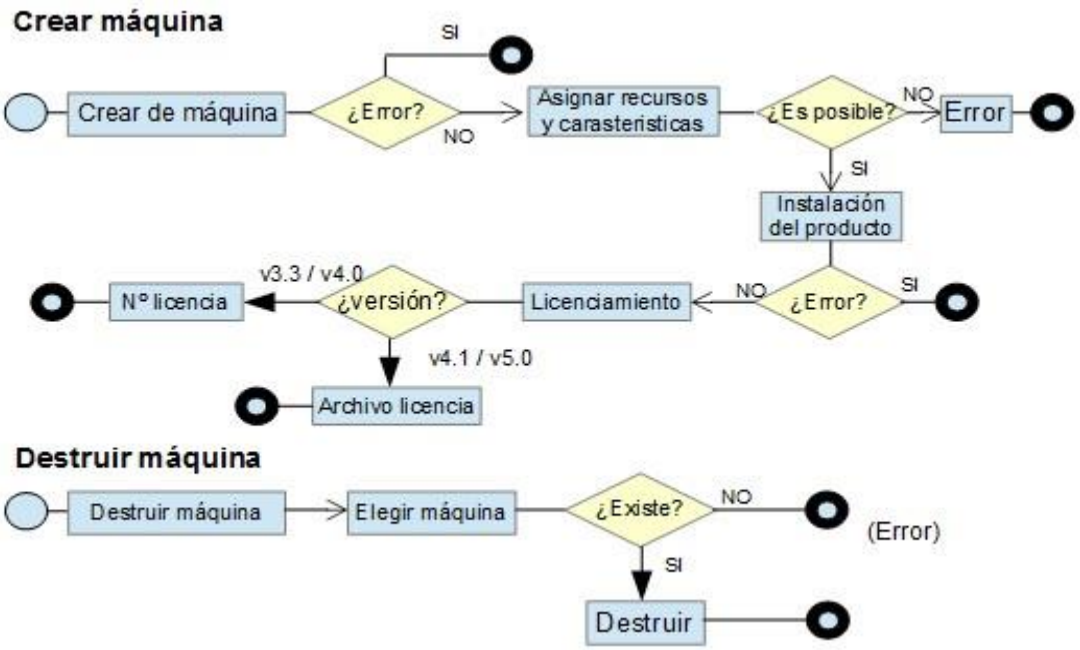

Figura 37:Diagrama de procesos: PBI 4

- **PBI 5: *Scripts* parar/arrancar máquina**

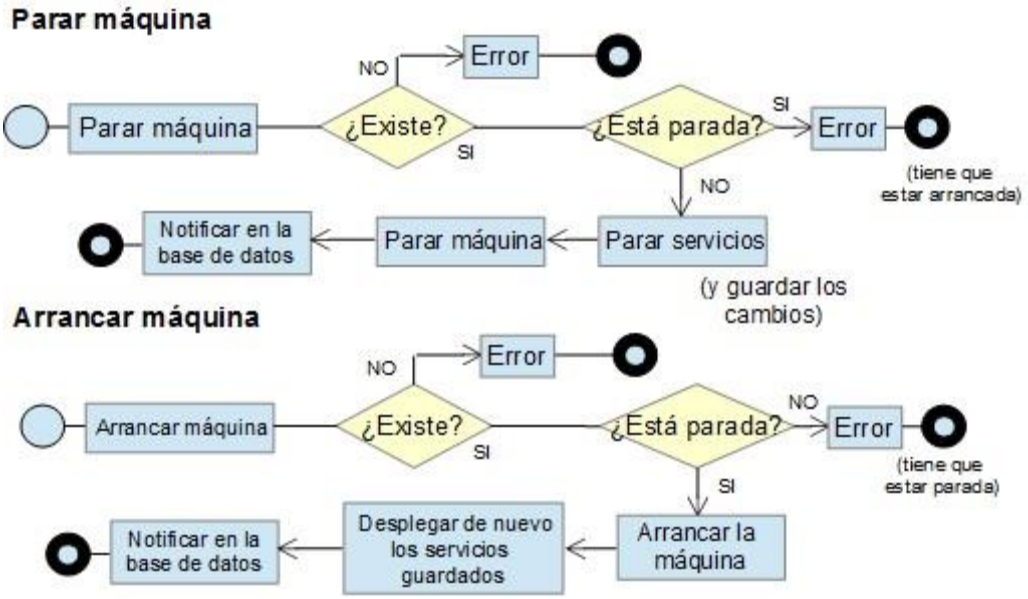

Figura 38:Diagrama de procesos: PBI 5

- **PBI 6: Dar de baja servicios**

Se esperan 48 horas para permitir reactivar el servicio, para evitar problemas derivados de errores humanos y poder reactivarlo sin tener que recuperar nada, es decir, la primera vez que se solicita, se avisa del borrado y si se confirma la baja del servicio hay que esperar 48 horas para su borrado, ya que en ese periodo de tiempo el cliente puede arrepentirse de ello y querer dar marcha atrás, por lo que cuando pase ese tiempo el servicio será borrado y se destruirá la máquina posteriormente. La parte de destrucción de la máquina está en un apartado anterior.

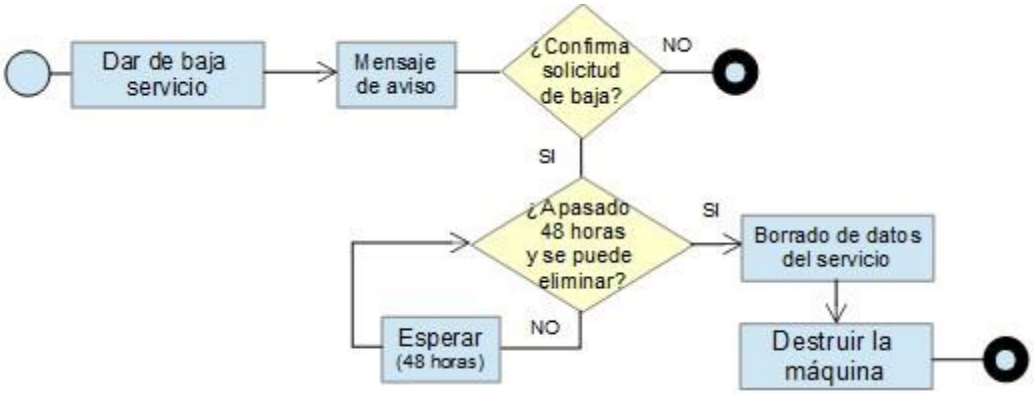

Figura 39:Diagrama de procesos: PBI 6

- **PBI 7: Arrancar/parar servicios**

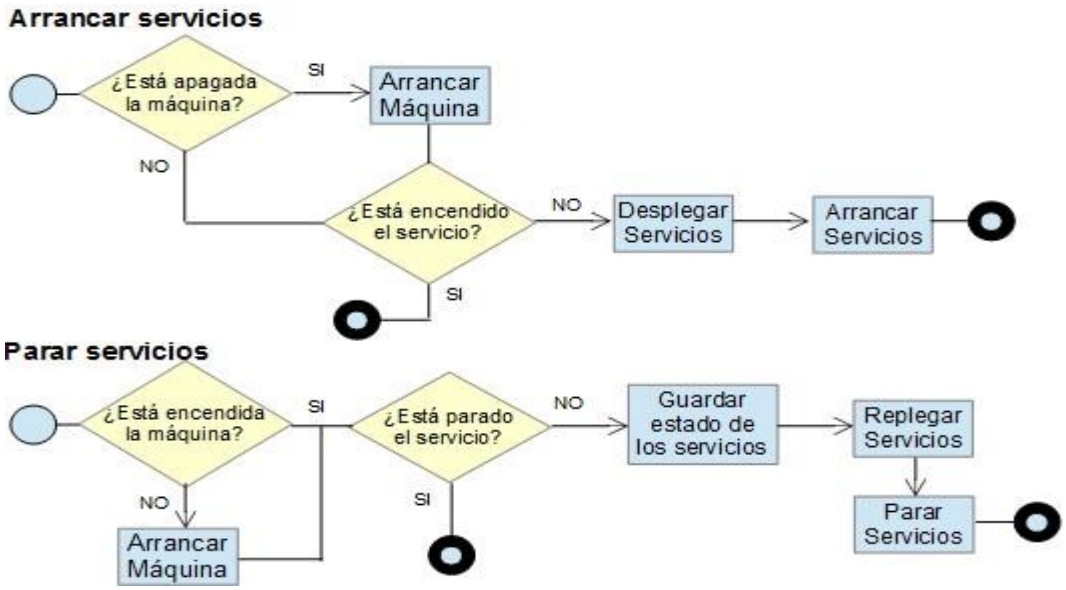

Figura 40:Diagrama de procesos: PBI 7

Es provisional, ya que la funcionalidad de arrancar/parar servicios no está muy clara aún y se necesita hablar con el personal indicado. La parte de Arrancar la máquina está implementada en diagramas anteriores y se implementa porque para arrancar o parar servicios se necesita que la máquina esté encendida.

En la parte de parar servicios es necesario que se guarden los cambios, ya que no es igual finalizar un servicio, que sería dar de baja como le hemos llamado nosotros, y parar un servicio, que es desactivarlo, para luego reanudarlo o arrancarlo de nuevo.

- **PBI 9: Mostrar lista de servicios contratados**

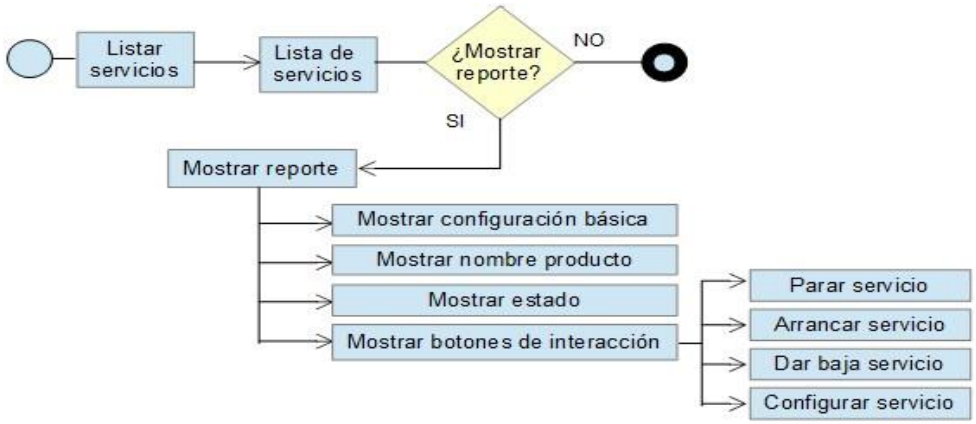

Figura 41:Diagrama de procesos: PBI 9

Todas las acciones que derivan de mostrar reporte serían nodos finales, o de cada una volver a el reporte general, es decir a "Mostrar reporte". También se puede ver la cantidad de instancias creadas y también interaccionar con ellas con las opciones de parar, arrancar y finalizar el servicio, tal y como se pide en este PBI, en la interfaz de EC2

- **PBI 10 y 11: Pantalla de configuración general/configuración automática.**

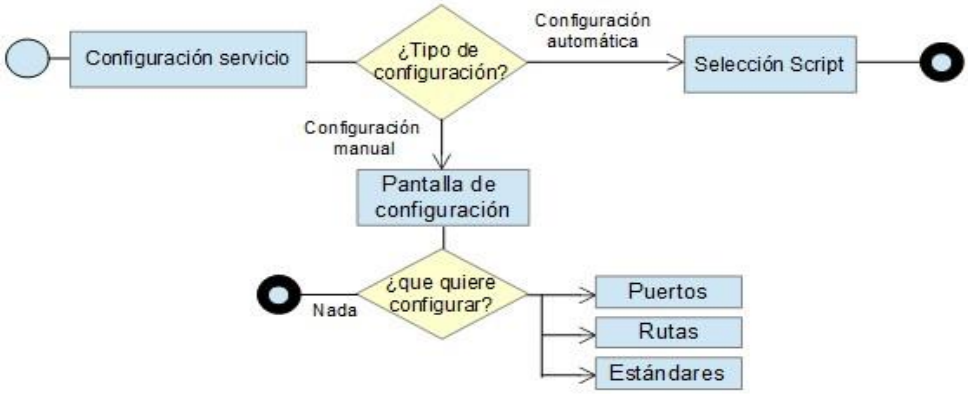

Figura 42:Diagrama de procesos: PBI 10 y 11

Este diagrama va a ser un modelo de procesos de dos PBIs conjuntos, que serán los PBI 10 y 11. El primero de ellos es crear una pantalla de configuración manual para cambiar por ejemplo los puertos de entrada y salida, rutas o estándares y el segundo de los PBIs es para hacer toda esa tarea que se hace manualmente, hacerla mediante un script de manera automática.

- **PBI 12: Downgrade/Upgrade de la máquina**

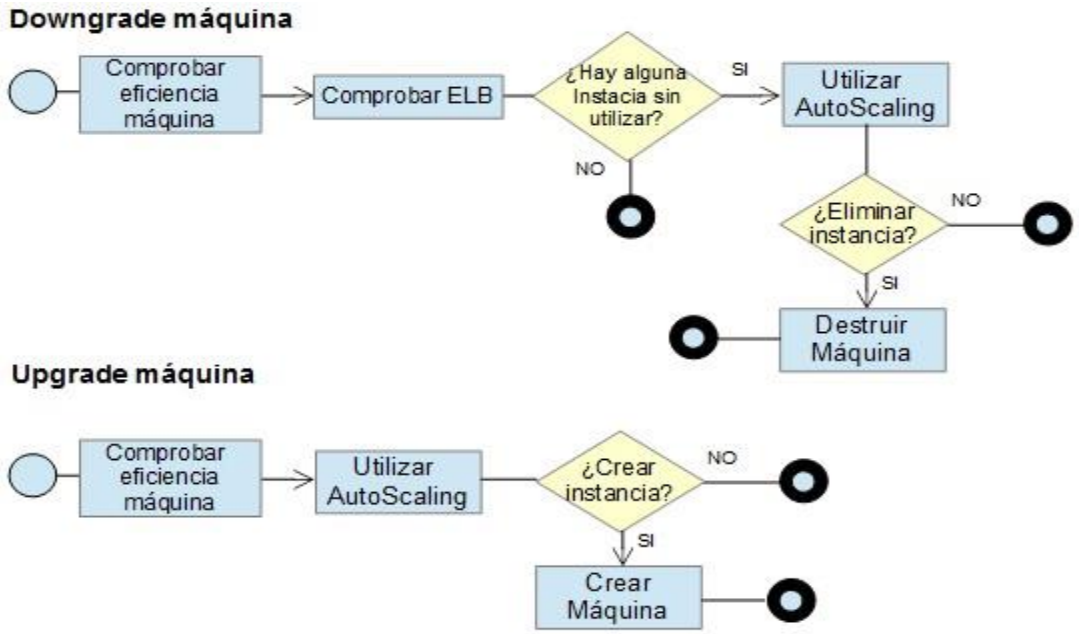

Figura 43::Diagrama de procesos: PBI 12

En estos modelos de procesos aparecen dos conceptos nuevos, ELB y *AutoScaling*. El primero de ellos es un balanceador que su función es comprobar que las instancias están operativas y de no serlo así, con la ayuda del *AutoScaling*, eliminaría las que no se utilizan o se crearían nuevas si se necesitan, para favorecer al coste por uso y no pagar de más.

- **PBI 13: Almacenamiento de la máquina**

Figura 44:Diagrama de procesos: PBI 13

Realmente es un suplemento que se añade a la hora de crear una máquina, AMI en AWS, si la respuesta es afirmativa, se contratará un espacio específico según las características que el cliente elija, asumiendo su coste particular. Todo ello estará contratado a través de otro servicio de AWS, llamado S3.

- **PBI 14: Login de usuarios**

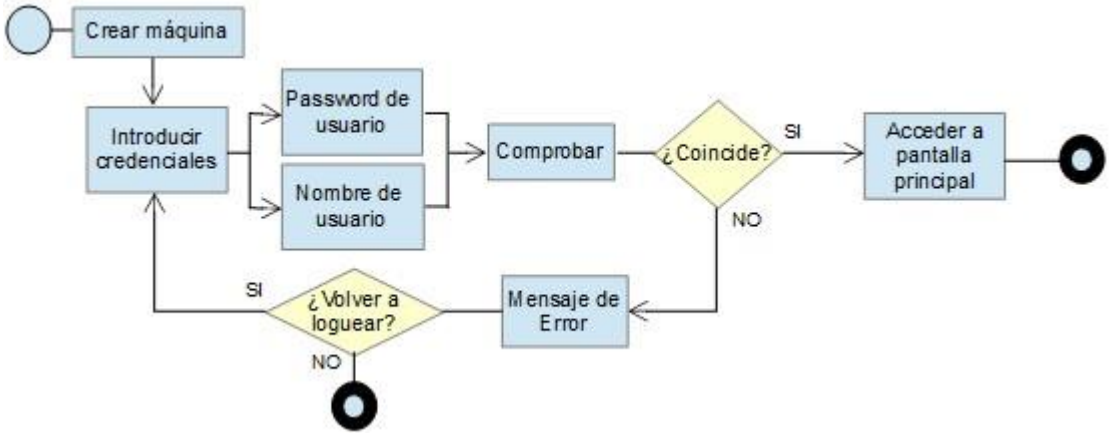

Figura 45:Diagrama de procesos: PBI 14

Esta parte es sólo un *login*, ya que el registro formará parte de los administradores del sistema y no podrá registrarse en él el propio cliente. La empresa facilitará una pareja de credenciales con un identificador o nombre y una *password*, que se deberá introducir en sus respectivos lugares y comprobar: si coincide se accederá a la pantalla principal del cliente, con las respectivas instancias de EC2 y el producto de DocPath® desplegado en ella. En caso contrario, se volverá a solicitar el acceso y para ello se tendrá que introducir nuevamente las credenciales proporcionadas con la empresa. Como en todos los formularios de acceso, se habilitará una opción de contacto por si ocurre cualquier problema con una dirección de correo o teléfono de contacto para solicitar ayuda, además de poder salir de todo y terminar.

MODELOS DE INTERFAZ DE USUARIO

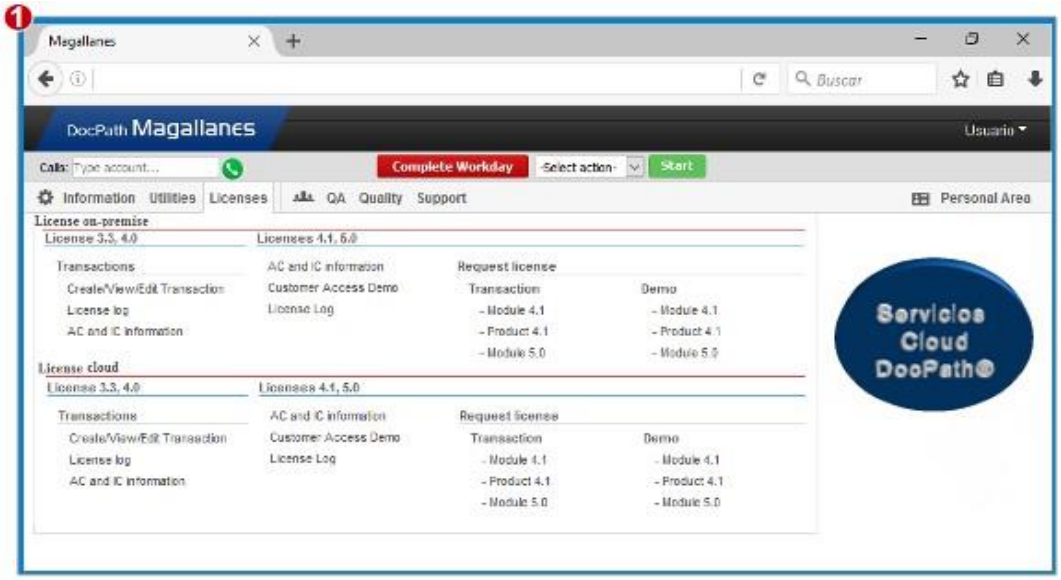

Figura 46:Interfaz de usuario- Integración del servicio con Magallanes

Esta es la apariencia que tendría Magallanes con los enlaces a los productos *cloud*, eso sería posible en el caso de tener al final todos los productos que se tienen para ejecutar localmente, tenerlos para ejecutarlos de manera *cloud*, en un principio es una aproximación, pero se puede poner de cualquier otra manera.

Además de esa manera un poco más enfocada al futuro, se le ha añadido un botón que servirá como enlace a nuestra web que será muy parecida a lo que aspecto se refiere a la principal, pero en la que sólo se podrá contratar servicios cloud. El *login* puede ser compartido, podría valer el mismo *login* de Magallanes o utilizar otro *login* distinto para acceder a los servicios *cloud* que se tengan. Posteriormente cuando se accede a nuestra plataforma web, tendría un aspecto parecido al de la segunda imagen. En ella habrá información sobre los servicios que se ofertan y un resumen de las posibles ventajas que tiene para la empresa que contrate esos servicios *cloud* en lugar de contratarlos *on-premise*.

En la página principal además de la información, habrá un cuadro donde los usuarios podrán identificarse, como hemos dicho antes, se puede tener ambas posibilidades, en esta página se tendría la otra posibilidad de logearse. Justo debajo del cuadro de logeo existe un botón o link en el que pulsando en él te redirigirá a otra ventana que será la ventana de registro que veremos a continuación.

Figura 47:Interfaz de usuario- Página principal de web CloudPlatform

El tercer modelo de interfaz es un diseño de un posible registro. En el sistema no se podrá registrar, sólo se podrá loguear en él. Las tareas de registro se hacen por medio de los administradores que otorgarán un usuario y una password para acceder a la visualización de los servicios y configuración de los mismos. Como se puede observar es un diseño muy rudimentario de la interfaz de registro, ya que le faltan campos y hay lugares vacíos pero que se rellenarían con información o campos necesarios para este apartado

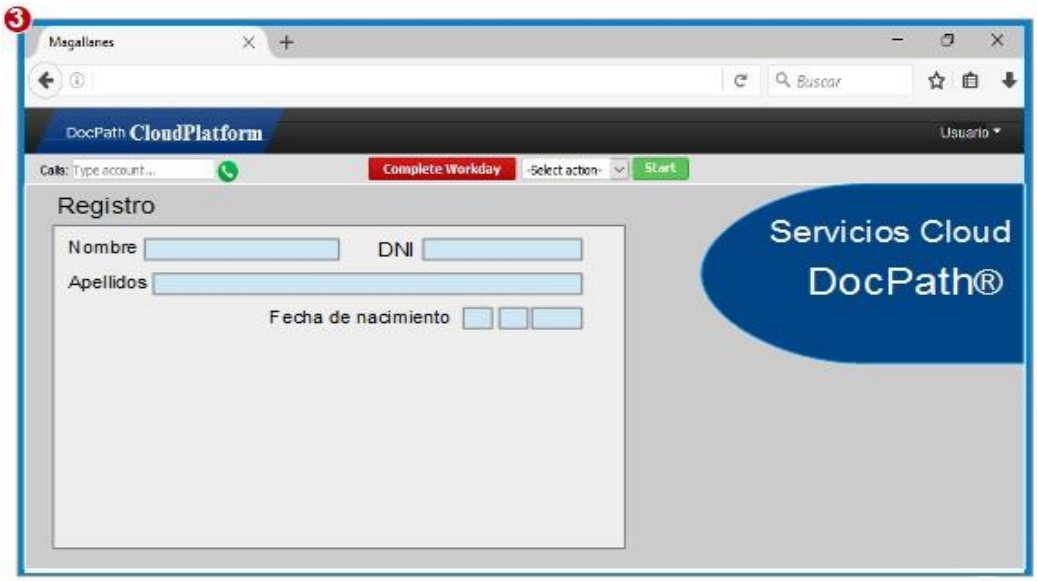

Figura 48:Interfaz de registro

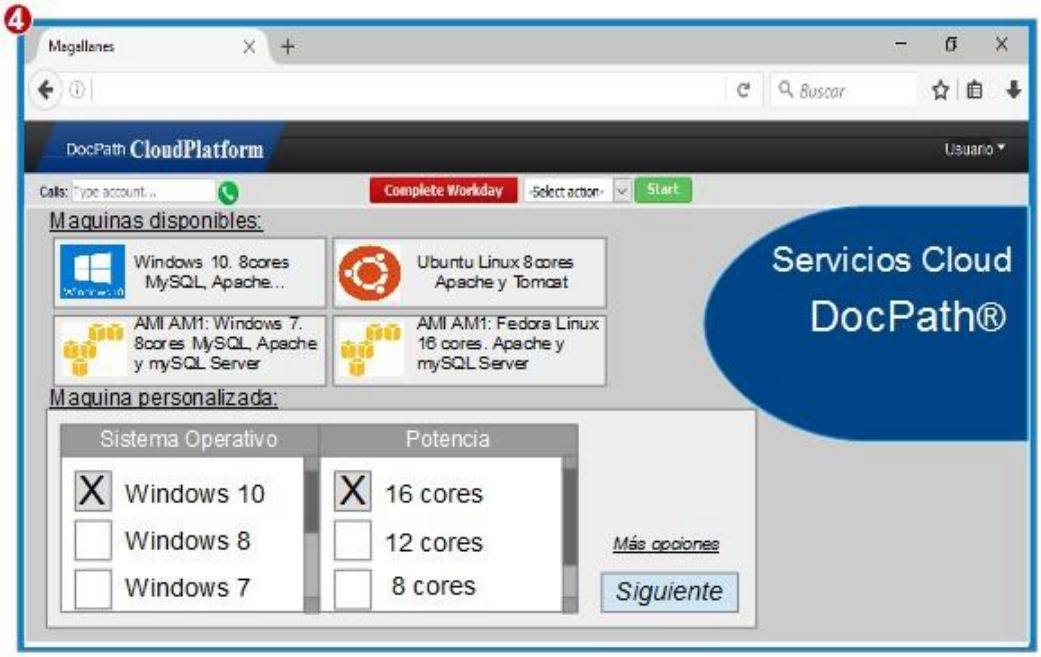

Figura 49:Interfaz de creación de instancia

En la cuarta de las interfaces representaría la selección de la máquina virtual en la que se desplegará el producto que posteriormente se indique. Se podrá elegir entre una AMI que ya está configurada como las primeras o se podrá crear una nueva AMI personalizada en la que instalar cualquier servicio o despliegue de un producto. Aquí en esta cuarta interfaz también hay sitios vacíos, pero lo importante estaría plasmado en ese boceto de la interfaz. En la cuarta de las interfaces representaría la selección de la máquina virtual en la que se desplegará el producto que posteriormente se indique. Se podrá elegir entre una AMI que ya está configurada como las primeras o se podrá crear una nueva AMI personalizada en la que instalar cualquier servicio o despliegue de un producto. Aquí en esta cuarta interfaz también hay sitios vacíos, pero lo importante estaría plasmado en ese boceto de la interfaz.

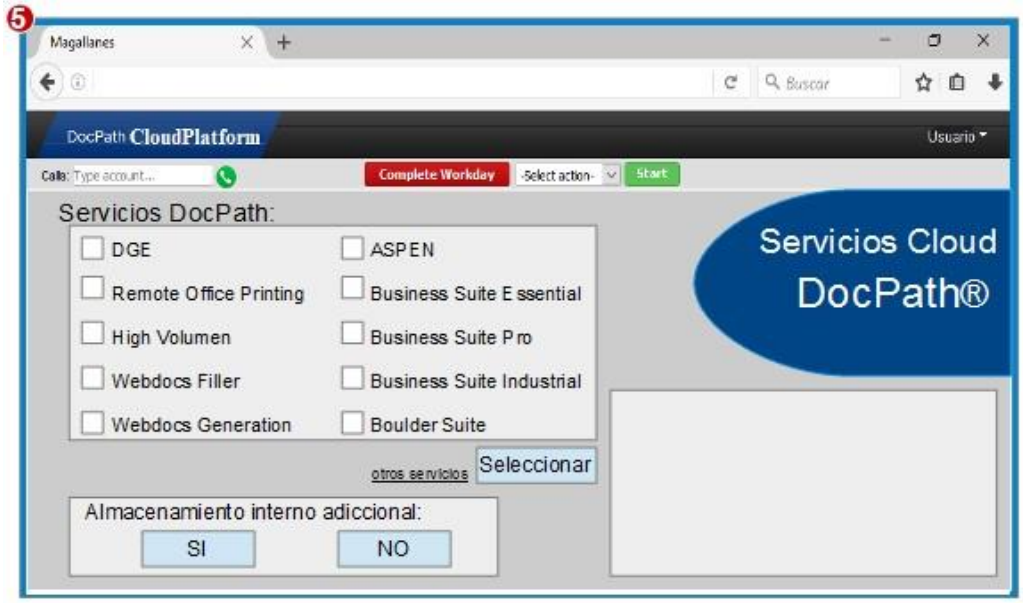

Figura 50:Interfaz de usuario-Selección de servicio

La quinta interfaz sería para la elección del servicio DocPath a desplegar y como en las anteriores quedan huecos libres para poder insertar algún otro campo

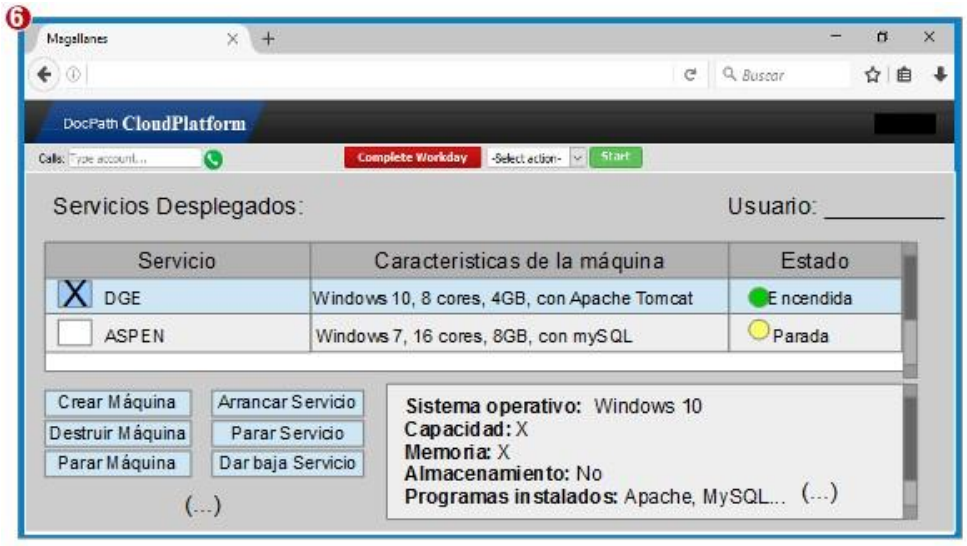

Figura 51:Interfaz de usuario-Lista de servicios y operaciones

La sexta y última de las interfaces es la más utilizada y representativa para los usuarios y sería una interfaz parecida a la que tiene el propio AWS en su servicio de EC2 en la que controlaría las instancias. En nuestro caso es lo mismo, pero además de controlar las instancias que cada usuario o cliente tiene, se podría ver muchas más cosas como: servicio que está desplegado, características de la máquina y el estado que tendría. Además, si se selecciona cualquiera de las instancias con su servicio desplegado se puede ver un reporte con la información específica de ese servicio. Hay que añadir que en la parte inferior

izquierda hay unos cuantos botones que sirven para aplicarle cualquier acción a la máquina o instancia cloud seleccionada. Estas acciones corresponden a cada funcionalidad o PBI que se busca en el alcance inicial, como son: crear, destruir, parar o configurar máquina y arrancar, parar, dar de baja servicio.

Como conclusión, hay que decir que todas estas interfaces corresponderían al diseño de la WebApp que se pide en el PBI 8 y que a priori no es un requisito muy prioritario y se podría evaluar analizar y estudiar de manera más detenida más avanzado el proyecto. Tan solo es un boceto y puede variar mucho el resultado final de todo lo anterior. Pero se podría decir que se han hecho estos diagramas, con la finalidad de comprender mejor la interacción del usuario con la aplicación y para que el desarrollador y el *Product Owner* acuerden un flujo de utilización de la plataforma.

ANEXO B: Descripción del servicio a desplegar

En este anexo se encuentra la descripción del producto elegido y la parte del servicio que se va a desplegar como parte cloud. En este anexo se explicarán las partes que lo constan y al final del mismo se dirá la parte que se despliega y las razones que nos ha llevado a ello.Para saber el producto a desplegar en la máquina, tenemos que definir más detalladamente uno de los productos de DocPath®, ese producto es ***Business Suite Pro***.

Business Suite Pro, es una solución que permite la generación de documentos. La solución consta básicamente de tres partes, un diseñador de plantillas, un controlador y un generador de las mismas, para ello definiremos dichas partes:

B.1 Diseñador

La parte de diseño está constituida por un programa concreto en el que se podrá realizar todo el diseño del formulario o documento concreto, ese programa se llama ***Areca Builder***. Al diseñador se le pasan como entradas la información del cliente, que puede venir en diferentes formatos como XML, base de datos, archivo de texto plano... y uno o varios formularios predefinidos y a partir de ahí el diseñador tendrá que hacer una serie de acciones para llegar a tener un diseño final del documento para facilitar la tarea del generador. Hay que distinguir en primer lugar dos tipos de partes dentro del diseño de un documento: partes estáticas y partes dinámicos.

- Partes estáticas: son partes que no cambian y que tal y como se diseñen saldrán en la impresión del documento, por ejemplo: el logotipo e información de la entidad, los índices de cualquier tabla, y textos del tipo: “nombre: “, “apellidos: “... todas esas partes que no cambian serían partes estáticas en un documento.
- Partes dinámicas: son los datos del cliente que podrían cambiar de uno a otro, son los datos personales como el nombre, DNI, fecha de nacimiento, los datos de negocio o gastos que se expresan en la factura o una oferta personalizada....

  Después de esa aclaración veremos parte por parte la composición interna del diseñador, ayudado por un ejemplo de diseño de un documento:
- ***Design***: parte de diseño de la página o páginas que constará el documento. En ella se pueden añadir: figuras geométricas, campos de texto estáticos, campos de texto dinámicos, imágenes tanto dinámicas como estáticas, códigos de barras, gráficos de

todos los tipos que se formarán con la información del cliente. En resumen, puedes "diseñar" la apariencia del documento y colocar los diferentes campos que lo dividirán. Los campos o partes se crean a través de *DocSegment*, que son pequeños bloques de contenido que siempre van unidos, por ejemplo, un bloque de contenido podría ser para todo lo relacionado con la información del cliente, aunque sea una parte dinámica también hay que ubicarla. Esto es de utilidad porque en un documento se pueden repetir muchas veces alguno de esos campos, por eso en vez de crear cada parte otra vez, se puede crear un *DocSegment* igual que los que has definido.

La imagen siguiente, muestra un ejemplo de la interfaz descrita anteriormente. Ahí se puede ver que hay campos acotados que tendrían información dinámica, pero que hay que modelar para facilitar la tarea al generador.

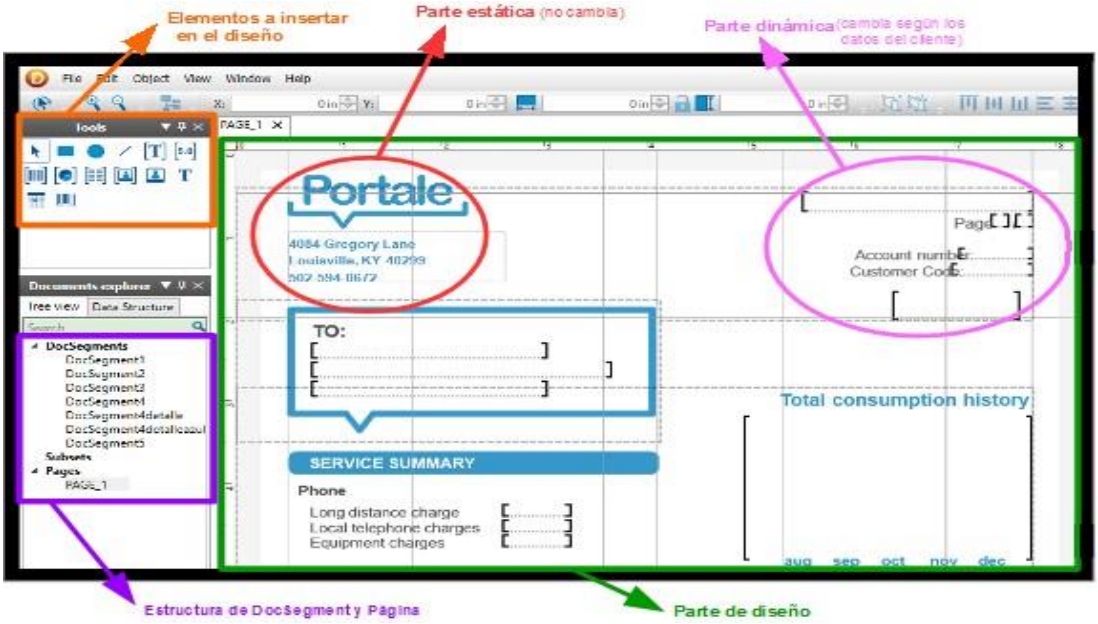

Figura 52:Design Builder

- ***Dynamic Struct*:** parte donde se define la parte dinámica del documento, se expresa el comportamiento que debería tener esa parte cuando reciba la información de los clientes y expresarla de la manera que indique en esa parte.

Figura 53:Dynamic Struct Builder

Como ejemplo, en la imagen de la izquierda, se puede observar dos segmentos de información, dos *DocSegment.* En concreto en la segunda de ellas, tiene dos símbolos que aplican un comportamiento a la parte dinámica indicada. En este ejemplo, el segundo de los *DocSegment* corresponde a una tabla, concretamente a la creación de

filas, y los símbolos que se indican, corresponden con el símbolo de repetición, que sirve, en este caso para volver a crear una línea si es necesario y el siguiente símbolo es para colorear según el comportamiento de la información, concretamente sirve para colorear las filas pares de un color específico.

- ***Pagination***: parte donde se define el comportamiento y seguimiento de las páginas. En el ejemplo de la derecha, tan sólo hay una página, por lo que el recorrido es trivial, pero esa es la forma de visualizar y vincular el recorrido de una página a otra.

Figura 54:Pagination Builder

- ***Job Flow****:* lugar donde se controla el flujo de trabajo que tiene que realizar para la realización del documento, es decir, se detallan las entradas, procesos y salidas que tiene la creación de ese documento. En la parte inferior se puede ver un esquema muy sencillo del ejemplo que estamos viendo. En la parte del *Input* se expresan los tipos de entradas que puede tener este proceso, en concreto en este sólo está XML, pero puede haber otros como *tag mode*, texto plano, base de datos... En la parte de *Proceses* se detallan los tipos de procesos que se realizan con esos datos, en concreto nosotros el proceso que realizamos se llama "*DocGeneration*", que se refiere a la generación del documento. Por último, en la última de las partes del diagrama, se indica los formatos y las salidas. Hay que hacer un inciso en esto, ya que no es lo mismo. Dentro de los formatos estarían: AFP, PCL, PDF, PostScript y ZPL. Por parte de las salidas estarían: impresión, archivo de disco, correo o fax, API Output, HTML5 y Archivo OD, aunque esto es para la solución *Business Suite Pro*.

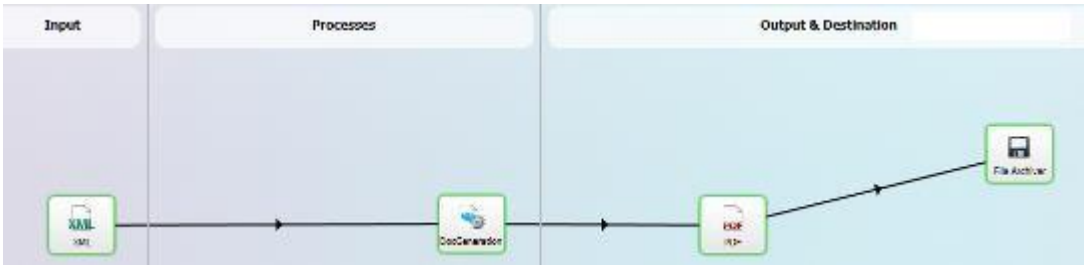

Figura 55:Diagrama de Job Flow Builder

- ***Definition***: parte en la que se describe el lugar donde tiene que ir cada elemento dinámico. La imagen siguiente, detalla la interfaz de *definition. E*n ella podemos ver las etiquetas del XML y a la derecha hay un diálogo en la que relacionar las etiquetas

del XML con los objetos que tiene la página 1, esa relación es necesaria para ubicar el contenido dinámico en su sitio indicado.

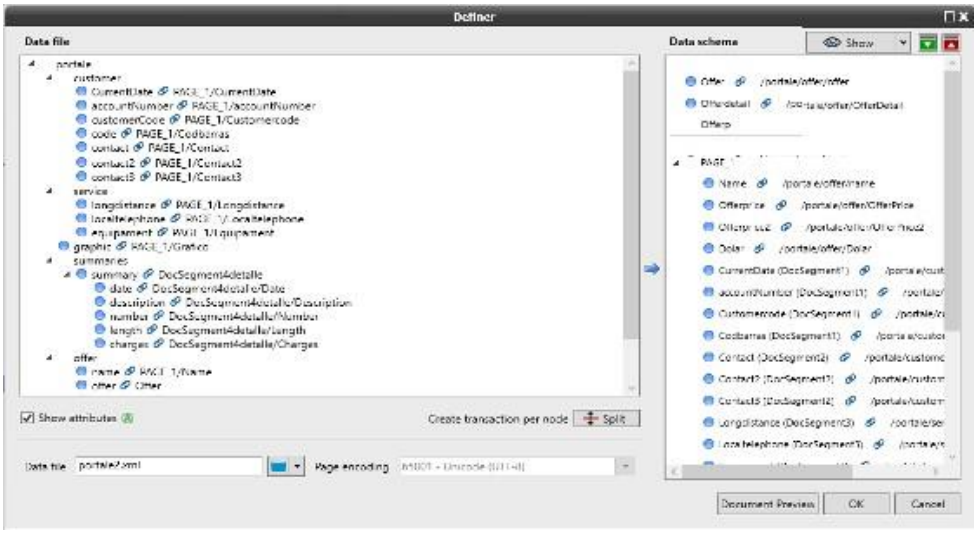

Figura 56:Funcionamiento parte dinámica

Todo ello, cuando se tiene un diseño del documento que se quiere, el Builder crea el documento, en concreto crea lo que sería el diseño con una extensión .IPF, que, junto con los datos del cliente, en este caso el XML, son las entradas que tendría el motor de generación de la solución, que es otra de las partes del producto.

B.2 Motor de generación (DGE)

El motor de generación es una serie de procesos que se instalan en el equipo que se lanzan y llaman unos a otros, siguiendo un orden. Son totalmente independientes y tienen una secuencia o proceso de funcionamiento. Las entradas a estos procesos son la salida del diseñador, es decir un .IPF y los mismos datos que se le pasan al diseñador para crear esa plantilla, los datos no varían, ya que es la única información que proporciona el cliente. El funcionamiento del **DGE** se puede resumir en los siguientes pasos:

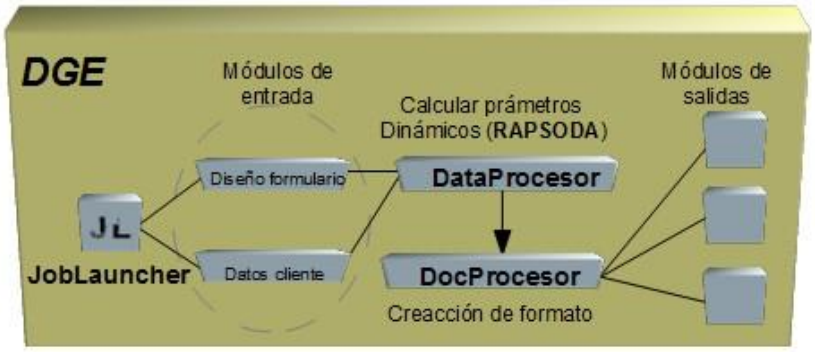

Figura 57:Funcionamiento del DGE

- **Job Launcher**: es la parte encargada de lanzar los procesos, elige el diseño del formulario y los datos del cliente y se los proporciona a los módulos correspondientes para seguir el proceso, concretamente *DataProcesor.*
- **Módulos de entrada**: los módulos de entrada son los .IDF que son creados con el diseñador y los datos en el formato que se establezcan: XML, base de datos, texto plano...
- ***DataProcesor:*** en este proceso se calcula toda la parte de los datos dinámicos. Realmente hay más procesos intermedios y no todo se realiza en *DataProcesor*, un proceso intermedio es *Rapsoda*.
- ***DocProcesor***: es el proceso encargado de crear el formato, es decir, en él se crea la solución final lista para ser impresa, guardada o enviada. Crea formatos por ejemplo como: AFP, PCL, PDF, ZPL...
- **Módulos de salida**: Son los módulos o procesos tales como:
    - ***PrintProcesor***: Módulo para imprimir los documentos.
    - ***FileArchiver***: Módulo para guardar en disco los documentos.
    - ***ODArchiver***: Módulo de *Output Dynamic*.
    - ***MailFax Procesor***: Módulo para enviar vía correo o fax los documentos.

En resumen, el diseñador devuelve un .IPF que se pasa como entrada junto con los datos del cliente al motor de generación y crea un documento final, impresión o envío. Hay que distinguir entre .IDF y .IPF. Los primeros son plantillas que utiliza el diseñador y cuando estas están diseñadas concretamente, especificando el tipo de entrada, los datos enlazados con la entrada concreta y demás pasarían a formar un .IPF que junto con los datos proporcionados por el cliente en formato XML, texto o cualquier manera se le pasan al DGE para que genere los documentos específicos.

Existe una solución más específica llamada ***OutPut Dynamics***, que es una solución que soporta un mayor volumen de documentos y formularios, permite utilizar varios .IPF diferentes y automatiza la ejecución de procesos de generación de documentos y archivos y guardarlos como plantillas para su posterior uso.

B.3 Controller

Es el encargado de gestionar y comunicar los procesos del diseñador con la parte del motor de generación. Actúa como su nombre indica de controlador. Tiene un aspecto programático a destacar que nos repercute en el proyecto y es que está programado en *C,* mientras que toda la parte de generación, así como todos los módulos que lo componen con sus respectivos *jar*, están programados en *Java,* como por ejemplo *DocProcesor* o *Rapsoda.*

B.4 Servicio a desplegar

Anteriormente se ha detallado las tres partes que componen el producto *Business Suite Pro*, pero todo el producto no es necesario desplegar para tener un funcionamiento óptimo del mismo. Solo se desplegará lo necesario.

La parte de diseño seguiría estando modelada de la misma manera, el cliente tendría la parte de diseño instalada en su ordenador para hacer con ella todos los diseños que necesite. Entre ellas se incluye el *Builder* y todos los procesos que lo relacionan como *DataProcessor* o *DPCalc.* Además de la parte de diseño, el controller seguirá actuando de la misma manera.

La parte que cambia será todo lo relacionado con el motor de generación. Todos los procesos incluidos e instalados cuando se instala el DGE serán los procesos que se instalen en el *cloud.* Algunos ejemplos son: *DocProcessor, FileArchiver, JobLauncher* y la parte del *Controller* que le afecta al DGE.

El funcionamiento entre el *Controller* y el DGE será a través de peticiones *REST*, es decir, se desplegará el servicio del motor de generación a través del despliegue de un war y se instalará en la instancia correspondiente con la ayuda de la API de AWS en concreto la de EC2 para hacer la contratación automática de la AMI, desplegar el servicio y realizar todas las acciones, como: crear/destruir instancias, dar de baja servicios, etc.

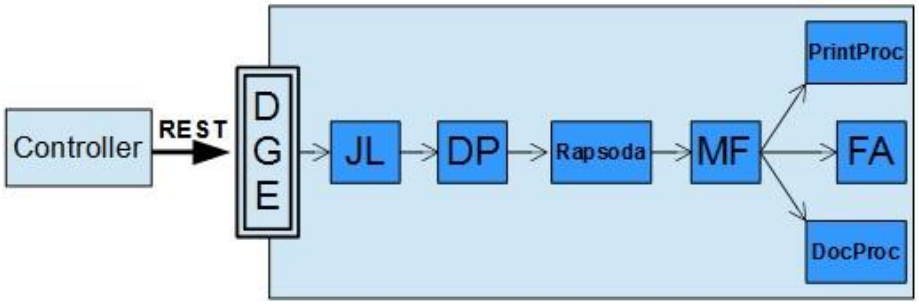

Figura 58:Producto a desplegar

La imagen anterior es un esquema y muestra todo el proceso que estaría en el *cloud*, a través de una petición *REST* se comunicará el *Controller* con el motor de generación y se comunicaría a partir de ahí con todos los procesos que estarán instalados en la AMI de *cloud.* Todos los procesos, son procesos independientes y se llamarán los unos a los otros cuando lo necesiten, aunque eso no es objeto de análisis, nuestro objetivo es el propio despliegue y no entraremos en detalle, tan sólo saber que estarán instalados como archivos ,*jar* y que todos los módulos tendrán una estructura de directorios con tres carpetas:

- ***Bin*:** para los archivos ejecutables como los jar.
- ***Configuration*:** para los archivos de configuración.
- ***Log:*** para los archivos de retroalimentación y salidas de los programas.

Se tiene que estudiar la manera de instalar esa parte en la AMI con la API de EC2 y realizar todas las acciones que se piden como requisitos. También hay que destacar que toda la parte que se tiene que desplegar, es decir el DGE está programado en *Java*, mientras que la parte del *Controller* está programado en C.

ANEXO C: Elección de proveedor

En este anexo se muestra el estudio hecho en la fase previa al desarrollo, en la fase de Inception. En él se analizan las tres grandes alternativas para contratar el espacio cloud, es decir, con que proveedor o plataforma cloud vamos a trabajar en este proyecto. Las alternativas son Google Cloud, Amazon Web Services y Microsoft Azure.

En la parte inferior se definen y comparan las características que nos proporcionan cada uno y posteriormente elegiremos el más propicio para el proyecto, argumentando los motivos de nuestra decisión.

C.1 Análisis de las alternativas

En este apartado se estudian las tres posibles alternativas propuestas para el desarrollo de la plataforma del proyecto. En primer lugar, hay que diferenciar que los tres son servicios muy similares, pero se diferencian en el enfoque, concretamente, unos son servicios IaaS y otros PaaS. IaaS significa infraestructura como servicio, gestiona los datos, proporciona los servidores y mantiene toda la infraestructura en la que se despliegan las aplicaciones o servicios de los desarrolladores, mientras que PaaS, proporciona solo una plataforma y un entorno que permiten a los desarrolladores crear aplicaciones y servicios que funcionen a través de internet. Nosotros buscamos en principio un servicio IaaS, por lo que en un primer momento que sea un servicio PaaS es motivo para descartar cualquier opción.

Mientras que Azure y Google Cloud son PaaS, AWS está más enfocado a un IaaS. En nuestro caso realmente lo que se busca es un IaaS. Hay que tener en cuenta todo esto.

Amazon Web Services (AWS[26])

Se define como una colección de servicios de computación en la nube que en su conjunto crean una plataforma de computación *cloud*, usada en *Dropbox, Foursquare* o *Hootsuit.* Es el pionero en este tipo de servicios y compite directamente con los servicios que proporciona *Microsoft Azure* y *Google Cloud Platform* entre muchos otros y entre sus clientes potenciales están *Netflix, Reddit, Pinterest* que se suman a los antes nombrados como *Dropbox*.

[26] https://aws.amazon.com/es/

En 2006 comenzó a proporcionar servicios de infraestructura de TI para empresas en forma de servicios web, conocido actualmente como *Cloud Computing* y ahora es la plataforma líder proporcionando servicios cloud. El beneficio principal es la oportunidad de remplazar importantes gastos anticipados en infraestructura para las empresas por costos reducidos para escalar su negocio, de manera que se puede tener un servicio desplegado con poco tiempo de antelación, sin adquirir servidores e infraestructura y sin previa planificación. Ventajas:

- Bajo costo: precios según el uso, sin gastos anticipados ni compromisos a largo plazo.
- Agilidad y elasticidad instantánea: proporciona una infraestructura que permite innovar, experimentar e iterar con rapidez. Puede aumentar su escala de negocio al instante.
- Accesibilidad y flexibilidad: es independiente del lenguaje y del sistema operativo.
- Fiabilidad: es segura, duradera y cuenta con certificaciones reconocidas por el sector como: ISO 27001, PCI DSS...

Amazon tiene múltiples soluciones para muchos ámbitos, nos centraremos en las soluciones de hospedaje de aplicaciones, que son las que más se ajustan a nuestra demanda.

PRODUCTOS:

- ***Amazon EC2*** *(Elastic Compute Cloud)*: permite a los usuarios alquilar máquinas virtuales en las que poder ejecutar sus propias aplicaciones. Proporciona un tamaño modificable en la nube, pagando en función a esa capacidad. Características:
    - Reduce el tiempo necesario para arrancar nuevas instancias del servidor, lo que permite escalar rápidamente la capacidad según las necesidades. Todas las instancias las "cuelga" de un directorio raíz y las clasifica desde el mismo.
    - Ofrece servicios totalmente flexibles. Puede elegir una instancia, con diferentes sistemas operativos, memoria, CPU o almacenamiento de la instancia.
    - Está pensado para integrarse con otros servicios de AWS y solo tendrá que pagar por la capacidad que contrate y utilice.
    - Otorga una disponibilidad del 99,95% en todas las regiones de *Amazon EC2.*
    - Proporciona las herramientas necesarias para crear aplicaciones seguras, ubicadas en una V*irtual Private Cloud (VPC)* con un rango de IP específico.

- ***Amazon S3*** *(Simple Storage Service)*: proporciona un servicio web que permite almacenar y recuperar la cantidad de datos que desee, en cualquier momento y desde cualquier parte.
    - Almacena grandes cantidades a un precio muy reducido. Únicamente tendrá que pagar por lo que necesite, sin compromisos, ni cuotas iniciales. Es escalable, ahorrará tiempo de planificación y maximizará la agilidad de su empresa.
    - Ofrece una durabilidad del 99,99999% y sus datos se almacenan de forma redundante en diversas instalaciones.
    - Es una tecnología segura y fiable, ya que permite transferir sus datos a través de SSL y cifrados de forma automática una vez que se han cargado.

- ***Amazon RDS*** *(Relational Database Service)*: facilita la configuración, escalado y funcionamiento de una base de datos relacional en el *cloud.* Ofrece una base de datos económica y re-dimensionable. Permite elegir motor entre: *Amazon Aurora, Oracle, Microsoft SQL Server, PostgreSQL, MySQL* y *MariaDB*. Beneficios que otorgan:
    - No es necesario instalar o mantener software de base de datos, utiliza la consola de administración de AWS o sencillas llamadas al API para acceder a ella.
    - Es muy asequible económicamente. Al igual que EC2 y S3, se paga por uso.

- ***Amazon DevPay***: es un servicio de administración y facturación online que le permite vender aplicaciones o servicios SaaS que están integrados en AWS.

Microsoft Azure[27]

En sus inicios fue llamada Windows Azure y está alojada en los Data Centers de Microsoft. Fue anunciada en 2008 en su versión beta, siendo un producto totalmente comercial el 1 de enero de 2010. Es el servicio encargado del alojamiento de las aplicaciones y el almacenamiento no relacional, sus aplicaciones pueden estar desarrolladas en .NET, PHP, C++ o incluso en Java. Características principales:

- Mayor productividad. Las herramientas y servicios integrados facilitan la administración de aplicaciones empresariales.

[27] https://azure.microsoft.com/es-es/

- Admite la mayor selección de sistemas operativos, lenguajes de programación, marcos, herramientas, bases de datos.... Además, ofrece ampliar su TI existente.
- Los datos se encuentran almacenados cerca de los usuarios para agilizar el acceso a ellos a través de una red de entrega de contenido.
- Permite escalar de acuerdo con sus necesidades, pago por uso.
- Protege sus datos de acuerdo con el estándar de privacidad ISO 27018.
- Es la plataforma desplegada en más países. Está desplegada en más países que AWS y *Google Cloud* juntos, es un precursor en este aspecto.

COMPONENTES:

- ***Windows Azure Compute***: es una plataforma aún en crecimiento y sirve para hospedar y administrar aplicaciones en los centros de datos de Microsoft. Consta de tres tipos de roles en ella: web, trabajo y máquina virtual. Los roles web, se usan para hospedar aplicaciones web *front-end*. Los roles de trabajo se utilizan para gestionar el procesamiento perteneciente a un rol web, mientras que el rol de máquina virtual gestiona todo lo relacionado con ella...
    - Proporciona un excelente entorno para hospedar aplicaciones, orientado para aplicaciones .NET, aunque también se incluye PHP y Java.
    - Se encarga automáticamente del equilibrio de carga y comunicación de errores, además está diseñado para una alta disponibilidad.
    - Crea aplicaciones escalables, que puedan reducir el alcance cuando sea necesario.
    - El tamaño de las instancias es flexible y se amolda a cada cliente.

- ***Windows Azure Storage***: es un servicio que facilita administrar y gestionar datos. Permite que los blobs, tablas y colas estén accesibles a aplicaciones o instancias de aplicaciones de manera simultánea. Los blobs son tipos de datos almacenados en las bases de datos, que sirven para almacenar datos de gran tamaño que cambian de manera dinámica. Por hacer un símil con AWS, sería muy parecido al producto *Amazon S3*.

- ***Microsoft SQL Azure***: es un servicio de base de datos en la nube, basado en las tecnologías de *SQL Server*. Con este servicio, los desarrolladores no tienen que instalar, configurar ni administrar ningún software. Ventajas:

- Pueden crear y administrar tablas, además de ejecutar fácilmente consultas complejas y uniones entre varias tablas, junto con funciones como facturación en tiempo real y generación de análisis históricos. Enfocada en aplicaciones web y empresariales personalizadas que aprovechen las funciones de la base de datos.
- Alta disponibilidad y tolerancia a errores integrada en el producto.
- Programación a través de .NET y compatibilidad con PHP y JDBC.
- Proporciona flexibilidad a la hora de ampliar la base de datos.

Google Cloud[28]

Es una plataforma igual que las dos anteriores, aunque algo menos utilizada, que proyecta sus puntos fuertes en que tiene una infraestructura para el futuro: una plataforma segura, rentable, de alto rendimiento y en constante evolución. Ofrece datos y análisis muy eficaces, que obtiene con mucha rapidez a través de su búsqueda en el Big Data.

- ***Compute Engine***: Permite crear y alojar aplicaciones web en los mismos sistemas escalables con los que funcionan las aplicaciones de Google. Ofrece procesos de desarrollo e implementación rápidos y con una administración sencilla, sin necesidad de preocuparse por el hardware o actualizaciones. Con App Engine no es necesario mantener ningún servidor. Basta con cargar su aplicación y ya se encontrará lista para servir al usuario. Escala automáticamente los recursos utilizados. Características:
  - Ofrece tipos de máquinas predefinidos y personalizados.
  - Permite distribuir las solicitudes entrantes en grupos de instancias y regiones.
  - Compatible con Windows y Linux. Incluye Debian, CentOS, Ubuntu...
  - Proporcionan un almacenamiento de disco persistente y un rendimiento uniforme. Crean todo tipo de discos de formato HDD o SSD hasta 64 TB.
  - Procesamiento por lotes. No es necesario realizar reservas y los precios son fijos, solo hay que seleccionar y gestionar el consumo.
  - Facturación por minuto. El cargo mínimo es de 10 minutos y después irá por minuto.
  - Posee un almacenamiento escalable, API muy completa y lenguaje parecido a SQL.

[28] https://cloud.google.com/products/

- ***Container Engine***: Es la plataforma sobre *App Engine* para manejar de manera distribuida en la nube contenedores *Docker*. La filosofía *Docker* es visualizar lo mínimo necesario para ejecutar una aplicación, sin la necesidad de tener un sistema operativo completo como host, sino que puede ser compartido, mínimo y eficiente. Características:
  - El contenedor del motor es compatible con el formato contenedor común acoplable, ya que son contenedores Docker y éste se puede ajustar automáticamente en función de la utilización de los recursos (CPU, memoria), que es completamente escalable.
  - Se establece una red híbrida que reserva un rango de direcciones IP para el clúster de contenedores, aspecto que la hace más segura.

- ***Cloud Storage***: Es una herramienta de almacenamiento masivo de datos en la nube replicado en varios lugares del mundo para dar disponibilidad y redundancia de información. Se trata de una infraestructura como servicio (IaaS), comparable a *Amazon S3*. Características:
  - Es interoperable con otras herramientas de almacenamiento en la nube y bibliotecas que funcionan con servicios como *Amazon S3*.
  - Proporciona una consistencia de lectura tras escritura para las operaciones de carga
  - Está diseñado para tener un 99,999999999% de disponibilidad. Almacena múltiples copias de forma redundante a través de múltiples sitios, con sumas de comprobación automática para garantizar la integridad de los datos.
  - Es casi infinitamente escalable. Ya sea que esté apoyando una pequeña aplicación o la construcción de un sistema de grandes dimensiones.

C.3 Estudio comparativo AWS / Azure / Google

Para hacer un estudio comparativo tendríamos que evaluar productos análogos, lo más parecidos posibles y que se ajusten a las necesidades del proyecto, aunque, además, se elijan otros productos adicionales que mejoren el producto. Aspectos comparativos:

- **Año de nacimiento**: Año de creación del servicio.
- **Escalabilidad automática** (*AutoScaling*): Posibilidad de incrementar o reducir de manera automática la cantidad de recursos asignados a un sistema o aplicación.
- **Estabilidad y disponibilidad del servicio**: Es una medida que nos indica el tiempo que está disponible una aplicación o plataforma, respecto a la duración deseada. También se

puede medir negativamente como la cantidad de veces que falla, pero normalmente se habla de ella de manera positiva.

- **Tipos de facturación**: Forma y periodo de facturación mínima que del servicio.
- **Precio aproximado**: Precio aproximado del servicio con unas características similares.
- **Imágenes para acelerar el aprovisionamiento**: Tipo de máquinas virtuales que ya disponen de un sistema operativo o frameworks previamente instalados y preconfigurados, para que sea más rápido y sencillo comenzar a trabajar sobre ellas.
- **Soporte para Windows**: Evalúa la capacidad de implementar sistemas que operen bajo Windows y definir sus versiones.
- **Soporte para Linux**: Evalúa la capacidad de implementar sistemas que operen bajo Windows y definir sus versiones.
- **Soporte para lenguajes**: Define los lenguajes soportados por las plataformas.
- **Disponibilidad geográfica**: Cantidad de centros de datos y puntos de distribución.
- **Backups**: Cantidad de copias y zonas en las que se distribuyen.
- **Soporte para almacenamiento de datos**: Define los medios físicos que ofrecen las plataformas analizadas para la persistencia de datos.
- **Soporte para colas**: Evalúa los soportes para colas facilitados por las plataformas. Las colas son también llamadas estructuras FIFO y se caracterizan por la inserción de elementos por un extremo y la extracción por el otro.
- **Soporte para servidores Web**: Define el servidor web que ofrece cada proveedor.
- **Seguridad**: Cantidad de certificaciones de seguridad de la plataforma.
- **Migración de servidor**: Posibilidad de migrar el servidor.

Características	AWS (*Amazon EC2*)	Microsoft Azure (*Azure Compute*)	Google Cloud Platform (*Google App Engine*)
Año de nacimiento	2006	2010	2011
Escalabilidad automática	Si, con la ayuda de *CloudWatch* y *AutoScaling*	Autoscaling Application Block y *Azure Fabric Controller*.	BigTable y GFS
Estabilidad y disponibilidad del servicio.	99.95% disponibilidad. Penalización: 99.95% - 99% → 10% < 99% →30%	99.95% disponibilidad. Penalización: 99.95% - 99% →10% < 99% → 30%	99.95% disponibilidad. Penalización: 99.95% - 99% →10% 99% -95% → 25% <95% → 50%
Tipos de facturación	Facturación por uso. Pago por horas. Consumo mínimo de 60 min	Facturación por uso. Pago por minutos. Consumo mínimo de 10 min.	Facturación por uso. Pago por minutos.
Precio aproximado	Servidor pequeño: 39€ Servidor mediano: 135€	Servidor pequeño: 43€ Servidor mediano: 180€	Servidor pequeño: 35€ Servidor mediano: 140€
Imágenes para acelerar el aprovisionamiento	Si. AMIs (*Amazon Machine Images*), hay AMIs definidas y otras, se pueden crear.	Si. Tiene una gran biblioteca de imágenes y puede crear imágenes propias	No
Soporte para Windows	Windows Server 2003, 2008, 2008 R2 y 2012.	Windows Server 2012 y 2008	No
Soporte para Linux	- SUSE Linux Enterprise - Red Hat Linux	- openSUSE 12.3 - SUSE Linux Enterprise - Ubuntu Server 13.04 - OpenLogic CentOS	Si. Pero las aplicaciones corren en un *sandbox*
Soporte para lenguajes	- C++ y C# - Java - Ruby y Perl - Python	- .NET - Java - Node.js - Python	- Java - Python
Disponibilidad geográfica	11 centros de datos 37 puntos de distribución	20 centros de datos 32 puntos de distribución	4 centros de datos 160 puntos de distribución
Backups	Realiza 3 copias en la misma zona geográfica	Realiza 3 copias en la misma zona geográfica	Realiza copias en todos sus centros de datos alrededor del mundo.
Soporte para almacenamiento de datos	- Amazon RDS - Amazon SimpleDB - SQL Server Express	- SQL Relacional - Tablas NoSQL - Blob no estructurado	Solo soportar bases de datos no relacionales.
Soporte para colas	Amazon SQS	Azure Service Bus	App Engine Task Queue
Soporte para servidores Web	Apache, IIS y algún otro	IIS V7.5	Jetty Web Server
Seguridad	20 certificaciones	25 certificaciones	18 certificaciones
Migración servidor	Si (VMware e Hyper-V)	Si (Hyper-V)	No

La tabla anterior es una comparativa de características de los tres proveedores de servicios cloud más importantes. Se puede ver un salto entre Amazon y Azure con Google, al menos en las características que consideramos críticas o más importantes para nosotros:

Por parte de la **escalabilidad** automática hay una gran diferencia no mostrada en la tabla. Por parte de Azure y AWS, la escalabilidad automática se realiza según el criterio que el propio programador establece con una serie de condiciones, mientras que, por parte de Google Cloud, la escalabilidad de sus aplicaciones es automática para todos los usuarios.

Otro aspecto son las **imágenes** que proporcionan los servicios. En esta ocasión la diferencia es mayúscula. AWS provee de más de 2000 plantillas con configuraciones diferentes para personalizar tu máquina virtual. Azure también tiene esa posibilidad, pero con una oferta más reducida, mientras que Google no brinda esa posibilidad.

Un aspecto importante para la realización del proyecto es el **soporte de lenguajes**. En este aspecto, Amazon ofrece un amplio abanico que cubre los principales lenguajes. Desde aplicaciones .NET con C#, aplicaciones Java multiplataforma y lenguajes como Ruby, Perl o Python. Por parte de Azure la oferta es igual de amplia que en AWS. Pero en el caso de Google Cloud, el soporte de lenguajes se resume exclusivamente a Python y Java.

Por parte del **almacenamiento de datos**, AWS vuelve a destacar, ya que cuenta con varias opciones en función de sus necesidades puntuales. Algunos ejemplos son: *Amazon S3*, para almacenamiento masivo, *Amazon RDS*, para base de datos relacionales, *Amazon SimpleDB*, para no relacionales, además de múltiples versiones de SQL Server para tecnologías .Net y PHP. Azure en su caso, ofrece 3 tipos de almacenamiento de datos: uno para dar soporte a SQL relacional, otro para tablas NoSQL y la última consiste en *Blobs* no estructurados, que posibilita almacenar grades cantidades de texto no estructurado, vídeos o imágenes. Por parte de Google Cloud, la única solución es una base de datos no relacional conocida como: "*Big Table*" y no soporta bases de datos relacionales, lo cual es un punto negativo para esta plataforma.

Referente al soporte para **colas**, todos los proveedores de servicios cloud ofrecen un único producto para ello. *Amazon Simple Queue Service* para Amazon *Azure Service Bus* para Azure y *App Engine Task Queue* para Google. Algo muy similar ocurre por parte de los **servidores web**, pero en este caso, si hay una distinción. Por parte de Google Cloud, existe una única alternativa: Jetty Web Server, mientras que por parte de Amazon y Azure se ofrecen al menos dos alternativas para dar soporte a ese tipo de aplicaciones web.

Otro aspecto menos significativo son los **precios y facturación**. No existe una diferencia significativa entre los precios de los proveedores. Si acaso hay que hacer una distinción es en el modo de facturación. En Amazon se factura por hora y en Azure o Google es por minutos.

Por último, otro de los aspectos en lo que cualquier empresa se fija es el ranking y cantidad de empresas que trabajan con cada proveedor. Amazon, en este caso, cuenta con mucha ventaja, ya que es la plataforma más contrastada y con más recorrido, aspecto que le ha otorgado y le otorga más acuerdos con grandes clientes.

En términos muy generales, AWS parece ser superior ya que ofrece una amplia gama de funcionalidades y mayor oferta. Su extensa lista de herramientas y servicios, junto con sus características empresariales, lo convierten en una propuesta muy sólida. Además, su enorme infraestructura proporciona ser más competente en cuanto a precio se refiere. Azure ha comenzado a cerrar la brecha entre los dos, y para aquellas organizaciones que han invertido en Microsoft, en términos de tecnología y habilidades para desarrolladores, Microsoft Azure es una buena propuesta. Google Cloud es una elección en continuo crecimiento, pero que aún se encuentra lejos de las dos anteriores, al menos en cuanto a prestaciones que nos interesan.

C.4 Elección de la alternativa

La elección final es AWS. En un principio se ha descartado Google Cloud ya que está mucho más limitado en lo que a prestaciones se refiere, tal y como hemos visto en la comparativa anterior. Entre AWS y Azure, se ha elegido AWS, porque nuestra empresa es primeriza en cuestiones de cloud y no tiene dependencias con cualquier otro proyecto desplegado anterior. Por profundizar un poco más, Azure está más enfocado a productos de Microsoft y a trabajar con .NET, mientras que nuestra aplicación será Java y para ello la API de AWS para Java es mejor y más propicia para ello.

En el tema de características técnicas, ambos proveedores están muy igualados, pero se inclina la balanza ligeramente para el lado de AWS, ya que tiene más alternativas para el almacenamiento y el soporte de lenguajes y poseen una oferta de productos e instancias más amplia, lo que hace que se puedan adaptar mejor al producto. Tan sólo tendría un punto negativo: el tipo de facturación, aspecto que no nos preocupa. Otro aspecto a favor de AWS, es la importancia de las empresas que trabajan con AWS. Entre dos plataformas con un coste

y rendimiento muy similar, las empresas se decantarán por el proveedor que le trasmita más seguridad y muchas veces se fijan en la cantidad e importancia de las empresas que confían en ellos y esto juega a favor de AWS. Por último, se define más en detalle la elección final: ***Amazon Web Services***[29] y algunos de sus componentes que se podrían utilizar en el proyecto***:***

AWS ofrece un conjunto de servicios, aplicaciones e infraestructura que le permite ejecutar prácticamente todo en la nube, desde aplicaciones empresariales hasta aplicaciones para dispositivos móviles. Posee un panel de control, llamado "*Management Console*" que le permite activar instantáneamente cualquier servicio de AWS. Es una interfaz que sirve para administrar la cuenta, monitorizar el gasto y administrar las credenciales.

En la consola, se podrán encontrar todos los servicios de AWS y anclarlos a la barra superior, creando así accesos directos a los servicios más utilizados. Básicamente, la consola se utiliza para administrar todo lo referente a AWS y como requisito solo requiere una cuenta de AWS y una sesión de ***Identity and Access Management*** **(IAM).**

IAM es un servicio web que ayuda a controlar de manera segura, el acceso a los recursos de AWS. Con este servicio, se puede controlar quien puede acceder a unos determinados recursos y que recursos utilizar y de qué forma, es decir, la parte de autenticación y autorización de AWS. A continuación, describiremos los servicios principales que proporciona AWS, de los cuales, utilizaremos alguno de ellos, los clasificaremos por tipo de servicio:

- *Computación**:***
    - ***Amazon Elastic Compute Cloud*** **(EC2)**[30]**:** es la parte central de AWS, es la parte de cómputo de la plataforma y permite a los usuarios alquilar máquinas virtuales para ejecutar cualquier aplicación del cliente.
        - Utiliza una variedad de AMIs con una gran cantidad de sistemas operativos y configuraciones instaladas. Permite el despliegue escalable de aplicaciones montando una AMI para crear una máquina virtual.
        - Reduce el tiempo necesario para obtener y arrancar nuevas instancias de servidor en poco tiempo, escala rápidamente las instancias.

[29] https://aws.amazon.com/es/
[30] https://aws.amazon.com/es/ec2/

        - Permite pagar sólo por la capacidad que realmente utiliza.
    - ***Amazon Lambda*[31]:** es un servicio de computación sin servidores que ejecuta el código como respuesta a eventos y administra automáticamente los recursos adyacentes. Este servicio puede ampliar o crear nuevas funcionalidades a AWS:
        - Se pueden ampliar otros productos de AWS y añadir funcionalidad a recursos como S3 o las tablas de DynamoDB y crear servicios *back-end*.
        - No se tiene que preocupar de actualizar el software, mantenerlo o administrar parches de seguridad, se encarga de manera automática.

- Almacenamiento**:**
    - **A*mazon Simple Storage Service* (S3)[32]:** es un servidor web, que aprovisiona de almacenamiento a través de interfaces de servicios web como REST, SOAP… un servicio de almacenamiento sencillo, escalable, fiable, rápido y de bajo coste.
    - ***Amazon Glacier*[33]:** servicio de almacenamiento en la nube seguro y de muy bajo coste, pero está enfocado al archivado de datos y copias de seguridad, es decir al almacenamiento a largo plazo de datos que se acceden con poca frecuencia.
    - ***Amazon Elastic Block Store (EBS)[34]:*** es un servicio orientado al almacenamiento por bloques que se conectan con las propias instancias de EC2.

- Base de datos:
    - ***Amazon Relational Database Service* (RDS)[35]:** servicio que facilita tareas de configuración, utilización y escalado de una base de datos relacional en la nube.
        - Proporciona capacidad rentable y de tamaño modificable
        - Administra las tareas de administración de la base de datos
        - Permite el acceso a las funciones de base de datos conocidas como: MySQL, MariaDB, Oracle o PostgreSQL.
    - ***Amazon DynamoDB*[36]:** servicio de base de datos NoSQL completamente administrado que proporciona un desempeño muy rápido, predecible y fácil de escalar. Evitan las cargas administrativas.

---

[31] https://aws.amazon.com/es/lambda/
[32] https://aws.amazon.com/es/s3/
[33] https://aws.amazon.com/es/glacier/
[34] https://aws.amazon.com/es/ebs/
[35] https://aws.amazon.com/es/rds/
[36] https://aws.amazon.com/es/dynamodb/

- ***Amazon ElastiCache***[37]: es un almacén de datos administrado en memoria y servicio caché de AWS. El servicio mejora el rendimiento de las aplicaciones web recuperando información de cachés administradas en memoria, en lugar de basarse solo en bases de datos más lentas basadas en disco.

- Herramientas administrativas
  - ***Amazon CloudWatch***[38]: es un servicio de monitorización de AWS. Se utiliza para el seguimiento de métricas y logs, establecer alarmas y reaccionar automáticamente a los cambios en los recursos de AWS, por ejemplo, cualquier instancia de EC2.
  - ***Amazon CloudFormation***[39]**:** ofrece la posibilidad de crear una colección de recursos de AWS relacionados para ofrecerlos de manera predecible y ordenada.

- Mensajería:
  - ***Amazon Simple Queue Service***[40] **(SQS):** es un servicio de colas de mensajes rápido, fiable, escalable y totalmente administrado, que hace que se puedan desacoplar los componentes de una aplicación en la nube.
  - ***Amazon Simple Notification Service***[41] **(SNS):** servicio de notificaciones push administrado que permite enviar mensajes individuales o distribuirlos a diferentes destinatarios. Puede enviar a Apple, Google e incluso Windows.

- Redes y distribución de contenido:
  - ***Amazon Virtual Private Cloud***[42] **(VPC):** servicio de computación que proporciona a los usuarios una nube privada virtual. Los clientes pueden acceder a EC2 a través de una red privada virtual basada en *Ipsec*.
  - ***Amazon Route 53***[43]**:** es otra de las partes de la plataforma de AWS y proporciona un sistema de DNS, nombres de dominio, escalable y potencialmente disponible. También permite encaminarlos a otras infraestructuras fuera de AWS.

---

[37] https://aws.amazon.com/es/elasticache/
[38] https://aws.amazon.com/es/cloudwatch/
[39] https://aws.amazon.com/es/cloudformation/
[40] https://aws.amazon.com/es/sqs/
[41] https://aws.amazon.com/es/sns/
[42] https://aws.amazon.com/es/vpc/
[43] https://aws.amazon.com/es/route53/

Acrónimos

AC	***A**ctivation **C**ode*
AFP	***A**dvanced **F**unction **P**resentation*
AJAX	***A**synchronous **J**avaScript **A**nd **X**ML*
API	***A**pplication **P**rogramming **I**nterface*
AMI	***A**mazon **M**achine **I**mage*
AWS	***A**mazon **W**eb **S**ervice*
BRUF	***B**ig **R**equirements **U**p **F**ront*
CCM	***C**ustomer **C**ommunications **M**anagement*
CPU	***C**entral **P**rocessing **U**nit*
CRM	***C**ustomer **R**elationship **M**anagement*
CSS	***C**ascading **S**tyle **S**heets*
CVS	***C**oncurrent **V**ersions **S**ystem*
DoD	**D**efinition **o**f **D**one
DGE	***D**ocument **G**eneration **E**ngine*
DOM	***D**ocument **O**utput **M**anagement*
DT	***D**evelopment **T**eam*
EC2	***E**lastic **C**ompute **C**loud*
EBS	***E**lastic **B**lock **S**tore*
EFS	***E**lastic **F**ile **S**tore*
ELB	***E**lastic **L**oad **B**alancing*
EPL	***E**ltron **P**rogramming **L**anguage*
ERP	***E**nterprise **R**esource **P**lanning*
FIFO	***F**irst **I**n, **F**irst **O**ut*
FORTE	**FORT**alecimiento de las competencias profesionales de los graduados para la mejora de su **E**mpleabilidad
GB	***G**iga**B**yte*
GFS	***G**oogle **F**ile **S**ystem*

GNU ***G**NU's **N**ot **U**nix*

GPU ***G**raphics **P**rocessor **U**nit*

HTML ***H**yper **T**ext **M**arkup **L**anguage*

HDD ***H**ard **D**isk* **D**rive

HTTP ***H**yper **T**ext **T**ransfer **P**rotocol*

IaaS ***I**nfrastructure **a**s **a** **S**ervice*

IAM ***I**dentity and **A**ccess **M**anagement*

IDE ***I**ntegrated **D**evelopment **E**nvironment*

IC ***I**nstalation **C**ode*

IP ***I**nternet **P**rotocol*

JDBC ***J**ava **D**ata**B**ase **C**onnectivity*

JS ***J**ava **S**cript*

MPP ***M**assively **P**arallel **P**rocessing*

PaaS ***P**latform **a**s **a** **S**ervice*

PBI ***P**roduct **B**acklog **I**tem*

PCL ***P**rinter **C**ommand **L**anguage*

PDF ***P**ortable **D**ocument **F**ormat*

PHP ***H**yper-text **P**re-**P**rocessor*

PO ***P**roduct **O**wner*

POM ***Project Objet Model***

RAM ***R**andom **A**ccess **M**emory*

RDS ***R**elational **D**atabase **S**ervice*

REST ***R**epresentational **S**tate **T**ransfer*

RFID ***R**adio **F**requency **ID**entification*

RSS ***R**eally **S**imple **S**yndication*

S3 ***S**imple **S**torage **S**ervice*

SaaS ***S**oftware **a**s **a** **S**ervice*

SCM ***S**upply **C**hain **M**anagement*

SM	***S**crum **M**aster*
SNS	***S**imple **N**otification **S**ervice*
SQL	***S**tructured **Q**uery **L**anguage*
SSD	***S**olid-**S**tate **D**rive*
SSL	***S**ecure **S**ockets **L**ayer*
TB	***T**era**B**yte*
TFG	**T**rabajo **F**in de **G**rado
TI	**T**ecnología de la **I**nformación
UML	***U**nified **M**odeling **L**anguage*
VPC	***V**irtual **P**rivate **C**loud*
XaaS	***X** **a**s **a** **S**ervice*
XD	*e**X**cel **D**ynamic*
XP	*e**X**treming **P**rogramming*
XML	*e**X**tensible **M**arkup **L**anguage*
ZPL	***Z**ebra **P**rogramming **L**anguage*