PRÁCTICA EMPRESARIAL EN CLOUD BASED S.A.S - DISEÑO E IMPLEMENTACIÓN DE UNA ARQUITECTURA WEB ESCABALBLE PARA 1DOC3.

NICOLÁS DURÁN LÓPEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2016

PRÁCTICA EMPRESARIAL EN CLOUD BASED S.A.S. - DISEÑO E IMPLEMENTACIÓN DE UNA ARQUITECTURA WEB ESCABALBLE PARA 1DOC3

NICOLÁS DURÁN LÓPEZ

TRABAJO DE GRADO PRESENTADO COMO REQUISITO PARA OPTAR AL TÍTULO DE INGENIERO DE SISTEMAS

DIRECTOR
PhD. SERGIO FERNANDO CASTILLO CASTELBLANCO

TUTOR
ING. ISRAEL ALFONSO PEDRAZA CAMPUZANO

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2016

TABLA DE CONTENIDO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE FIGURAS

Pág.

RESUMEN

TÍTULO: PRÁCTICA EMPRESARIAL EN CLOUD BASED S.A.S. - DISEÑO E IMPLEMENTACIÓN DE UNA ARQUITECTURA WEB ESCABALBLE PARA 1DOC3.[1]

AUTOR: NICOLÁS DURÁN LÓPEZ[2]

PALABRAS CLAVES: Arquitectura web escalable, desarrollo ágil.

DESCRIPCIÓN:

Cloud Based S.A.S. es una empresa dedicada al desarrollo de soluciones tecnológicas a la medida y del acompañamiento, administración e implementación de TI (Tecnologías de la Información), y se apoya en las más modernas tendencias de diseño y administración de sistemas escalables globalmente para poner a disposición de sus clientes las más altas tecnologías y ampliar su alcance a niveles que sólo el internet hoy en día les puede brindar.

Lo anterior motiva a Cloud Based S.A.S a buscar jóvenes estudiantes a quienes pueda formar y que los ayuden con los retos que les representa estar ampliando sus clientes y proyectos.

La práctica empresarial se basó en tres principios: en primer lugar el constante aprendizaje y mejoramiento de los procesos en el área de desarrollo, enfocados siempre en las tecnologías de punta y las metodologías de desarrollo y patrones de diseño que más se adaptaran a cada proyecto; en segundo lugar, aplicar todo el aprendizaje en la construcción de plataformas multicanal que fueran mantenibles y escalables tanto en su desarrollo como en su funcionamiento; por último se llevaron a cabo tareas de monitoreo y mantenimiento de sistemas previamente construidos, así como su mejoramiento y ampliación. Estos tres principios fueron aplicados en un proyecto específico de la empresa llamado 1DOC3, que tuvo y tiene aún requerimientos tecnológicos muy exigentes y permitió al estudiante durante la practica empresarial aplicar los conocimientos adquiridos durante toda su carrera en un entorno real y al mismo tiempo desarrollar habilidades en un ambiente laboral.

[1] Proyecto de grado en modalidad de práctica empresarial

[2] Facultad de Ingenierías Físico – Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: PhD. Sergio Fernando Castillo Castelblanco Tutor: Ing. Israel Alfonso Pedraza Campuzano.

ABSTRACT

TITLE: BUSINESS PRACTICE FOR CLOUD BASED S.A.S – DESIGN AND IMPLEMENTATION OF A SCALABLE WEB ARCHITECTURE FOR 1DOC3[3]

AUTHOR: NICOLÁS DURÁN LÓPEZ[4]

KEYWORDS: Scalable web architecture, agile development.

DESCRIPTION:

Cloud Based S.A.S. is an organization dedicated to the development of custom technological solutions, the administration and the implementation of IT (Information Technologies). The company bases its work in the most advanced design and administration trends about globally scalable software in order to give its clients high end technology and allowing them to expand their reach to global levels that only the Internet can give them today.

The above motivates Cloud Based S.A.S into looking for young students who they can train and for facing the challenges that being a growing company have.

The engineering practice was based on three principles: first of all, the continuous learning and improvement of the development team processes, always focusing in high end technologies and the most suitables development methodologies and design patterns for each proyect; secondly, to apply the acquired knowledge on building multichannel platforms that were maintainable and scalable in both its design and its development operations; and finally, additional tasks such as monitoring and maintenance of existing applications, as well as its improvement and its extensión. These principles were applied in a specific proyect called 1DOC3, which demand high technology solutions and allow the student to apply the acquired knowledge throughout the career in a real environment and at the same time developing working skills.

[3] Final undergraduate project.

[4] Physics Mechanical Engineering Faculty. Informatic and System Engineering School. Director: PhD. Sergio Fernando Castillo Castelblanco. Advisor: Eng. Israel Alfonso Pedraza Campuzano

INTRODUCCIÓN

El proyecto de grado en modalidad de práctica empresarial es la perfecta manera para el estudiante de fortalecer los conocimientos adquiridos mediante la aplicación de estos en empresas altamente competitivas y aportándole a estas conocimientos y técnicas que de otra manera le tomaría a la industria años en aplicar e incluso conocer. El hecho de lidiar con ambientes de producción y alta exigencia tecnológica permite familiarizarse con la forma de trabajo del mundo real y ayuda al estudiante a adentrarse en el mundo laboral, adquirir habilidades de trabajo en equipo aún bajo presión y el manejo de las relaciones con los clientes y personas claves en su futuro laboral.

Hoy en día la creación de aplicaciones y de negocios en torno a ellas crece como nunca antes, la creciente penetración de teléfonos inteligentes y de redes de alta velocidad han permitido la masificación de este tipo de tecnología y de la misma manera han venido apareciendo mejores herramientas para su desarrollo, nuevas metodologías y arquitecturas capaces de hacer llegar a millones de personas estas tecnologías.

Cloud Based S.A.S. se ha caracterizado siempre por poner a disposición de sus clientes los beneficios de la tecnología, por lo que siempre se ha preocupado por mantener sus metodologías y sus desarrollos al más alto nivel permitiéndole competir en el mercado nacional y mundial, para lograrlo ha requerido talento humano con alta capacidad técnica, de rápido aprendizaje, con un perfil profesional competitivo y buenas bases de conocimiento, características que ha encontrado en los estudiantes de la Universidad Industrial de Santander.

A continuación se expone una recopilación de lo que fueron seis meses de práctica empresarial, se resumen las actividades realizadas y experiencias adquiridas por el estudiante en práctica, así como las recomendaciones y conclusiones que dejó la misma.

1. DESCRIPCION DE LA PRÁCTICA

En el presente capítulo se describen los aspectos generales de Cloud Based S.A.S, organización donde se llevó a cabo la práctica empresarial, tales como la razón social, reseña histórica y campo de acción, así como los detalles del proyecto en el que se enfocó la práctica, sus alcances y los objetivos del mismo.

1.1 DESCRIPCIÓN DE LA EMPRESA

1.1.1 Nombre de la empresa

Razón social: Cloud Based S.A.S.
Tipo de organización: Sociedad por Acciones Simplificada
Fecha de fundación: 28 de abril de 2011
Domicilio: Km 4 Anillo Vial, Zona Franca Santander, Baichala torre 1 Oficina 703, Floridablanca, Santander
Ciudad: Bucaramanga

1.1.2 Misión

Somos una empresa de base tecnológica que está permanentemente innovando a través de sus plataformas de mensajería de texto, mensajería de voz y aplicaciones digitales tanto para el móvil como para el computador que trabaja constantemente en búsqueda de mejorar los procesos de las empresas para que sean más rentables y para que las personas reciban una comunicación más amigable y divertida.

1.1.3 Valores

Sencilla, versátil, innovadora, divertida, rentable, acogedora.

1.1.4 Estructura organizacional

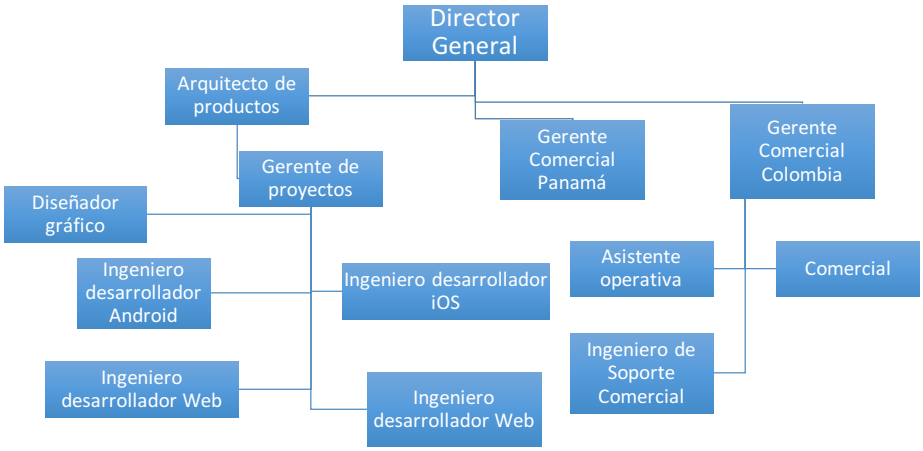

Figura 1. Estructura organizacional Cloud Based S.A.S.

1.1.5 Responsabilidades a cargo

El practicante tuvo la responsabilidad de diseñar y ejecutar todo el proyecto relacionado con el cliente 1DOC3, así como el seguimiento del mismo y la responsabilidad de cumplir los tiempos de entrega establecidos con el cliente.

1.2 DESCRIPCIÓN DEL PROYECTO

1.2.1 Planteamiento del problema

Cloud Based S.A.S. provee tecnología y desarrollo a empresas y proyectos de cualquier tamaño, dentro de sus clientes se encuentra una prometedora empresa llamada 1DOC3 que busca ser la mayor plataforma de información médica en español a nivel mundial, por lo que sus requerimientos tecnológicos son tan grandes que requiere el apoyo de empresas como Cloud Based S.A.S. para mantenerse competitiva y crecer más rápidamente.

Por ser 1DOC3 una empresa joven pero con un ritmo de crecimiento muy acelerado, sus requerimientos son muy dinámicos, sus objetivos pueden

cambiar muy rápidamente y se requiere desarrollar e implementar toda la tecnología de manera que sea escalable a millones de usuarios, que permita la fácil colaboración por parte de otros profesionales incluso en otros países y se rija bajo los más altos estándares de seguridad.

1.2.2 Justificación

Actualmente, el 72% de las personas que tiene acceso a internet formula preguntas médicas relacionadas con síntomas experimentados por ellos o sus familiares, e incluso busca validar si el diagnóstico dado por un doctor es acertado.

La información que encuentran las personas que acuden a internet para resolver sus dudas médicas no está verificada, o puede ser innecesariamente alarmante.

De lo anterior surge la necesidad de crear un portal de preguntas médicas gratis y anónimas, donde todo tipo de doctores especialistas responden preguntas directamente a los usuarios. Adicionalmente, les permite a los doctores crear su propio perfil en la plataforma con el fin de darse a conocer, crear una reputación en línea y adquirir nuevos pacientes por medio de internet, el cual hoy en día muy pocos médicos utilizan como medio para darse a conocer.

1DOC3 es hoy en día un portal web desarrollado por la empresa, con más de 60.000 registros y más de 100.000 preguntas respondidas por diferentes doctores, que crece cada día, por lo cual, es necesario extender sus funcionalidades a medida que crecen las necesidades de sus usuarios.

Durante la práctica empresarial en CLOUD BASED S.A.S. que constituye el presente proyecto de grado, se planea analizar las nuevas necesidades de la plataforma, diseñar soluciones para estas necesidades e implementarlas dentro de esta plataforma.

1.2.3 Objetivos

Objetivo General

Analizar los nuevos requerimientos de la plataforma web de orientación médica 1doc3.com y diseñar e implementar las respectivas funcionalidades requeridas.

Objetivos Específicos.

- Analizar y diseñar las funcionalidades requeridas actualmente, como son:
    - Perfil destacado para los doctores con membresía en la plataforma.
    - Implementar el protocolo SSL en la plataforma.
- Analizar los nuevos requerimientos de la plataforma 1doc3, basados en las necesidades actuales de los usuarios.
- Diseñar las nuevas funcionalidades que surjan de los anteriores requerimientos, de forma que se adapten al estado actual y sean escalables a futuras ampliaciones.
- Implementar las soluciones diseñadas en la plataforma 1doc3.
- Documentar y socializar cada una de las soluciones implementadas en la plataforma 1doc3.
- Monitorear la carga, el almacenamiento y el uso de red de cada componente del sistema.

2. MARCO TEÓRICO

La plataforma 1DOC3 es una aplicación web alojada en la nube que permite a usuarios de todo el mundo enviar preguntas de salud a doctores profesionales quienes, a través de la misma plataforma, responden las preguntas.

Los usuarios en 1DOC3 o requieren registrarse para navegar, pero si para enviar una pregunta, lo que implica manejo de sesiones y autenticación de usuarios. Todas las preguntas son almacenadas en la base de datos para que futuros usuarios puedan acceder a este contenido.

Figura 2. Ejemplo de una pregunta respondida en la plataforma 1DOC3

La plataforma de 1DOC3 está alojada en Amazon Web Services, a continuación se presenta una introducción a los servicios utilizados y su arquitectura inicial.

2.1 ARQUITECTURA DE HARDWARE INICIAL

La plataforma 1DOC3 ha tenido un crecimiento muy acelerado, la arquitectura de hardware y de software inicial era básica y permitió llevar su negocio hasta el nivel actual.

Toda la plataforma fue construida sobre Amazon Web Services, por lo que a continuación se presentan los servicios que fueron utilizados para construir la primera versión de la plataforma.

2.1.1 Amazon Web Services

Amazon Web Services[5] (AWS) es una plataforma de servicios en la nube que ofrece potencia de cómputo, almacenamiento de basas de datos, entrega de contenido y otras funcionalidades para ayudar a las empresas a escalar y crecer.

Amazon es el líder mundial a la fecha en este tipo de servicios, poniendo a disposición de sus usuarios servidores virtuales en la nube para múltiples casos de uso, entre ellos el hosting de aplicaciones web, que es uno de los usos que se le dará durante esta práctica.

AWS ofrece soluciones basadas en la nube para múltiples requerimientos de un negocio. Ejecutar una solución en la nube de AWS ayuda a poner en marcha y desarrollar aplicaciones escalables en un menor tiempo. Con la nube de AWS, es fácil disponer de una amplia gama de servicios, socios y opciones de soporte para poder centrarse en que el proyecto a desarrollar sea todo un éxito.

Los servicios de AWS sobre los que se construyó inicialmente 1DOC3 fueron:

[5] ¿Qué es AWS?, https://aws.amazon.com/es/what-is-aws/

- **Amazon EC2:** Es un servicio de alojamiento web en la nube, que permite crear servidores virtuales con todos los permisos de acceso, normalmente usados para alojar aplicaciones web.
- **Amazon RDS:** Es un servicio de alojamiento de bases de datos completamente administrado. Provee un servidor virtual optimizado para basas de datos y se pueden crear bases de datos de los motores más comunes como MySQL, PostgreSQL, Oracle, etc. En este servicio, AWS se encarga de llevar a cabo las instalaciones y actualizaciones de software de base de datos y backups automatizados.
- **Amazon S3:** Es un servicio de almacenamiento escalable, permite almacenar un número ilimitado de archivos en la nube, permitiendo decidir si se quieren dejar disponibles en la web a través de una URL. Es un servicio completamente administrado que no requiere creación de servidores ni destinar ninguna cantidad de almacenamiento, AWS se encarga de toda la infraestructura detrás del servicio.
- **Amazon Route 53:** Es un servicio web DNS (Sistema de nombres de dominio) escalable y de alta disponibilidad en la nube. Está diseñado para ofrecer a los desarrolladores y las empresas un método de confianza y rentable de redirigir a los usuarios finales a las aplicaciones en Internet convirtiendo nombres legibles para las personas como www.ejemplo.com en direcciones IP numéricas como 192.0.2.1 que utilizan los equipos para conectarse entre ellos.

Estos servicios funcionan dentro de la nube de AWS, distribuida en varios lugares alrededor del mundo, pero se comunican entre si dentro de una red virtual privada configurable para permitir el acceso sólo desde donde los usuarios deseen.

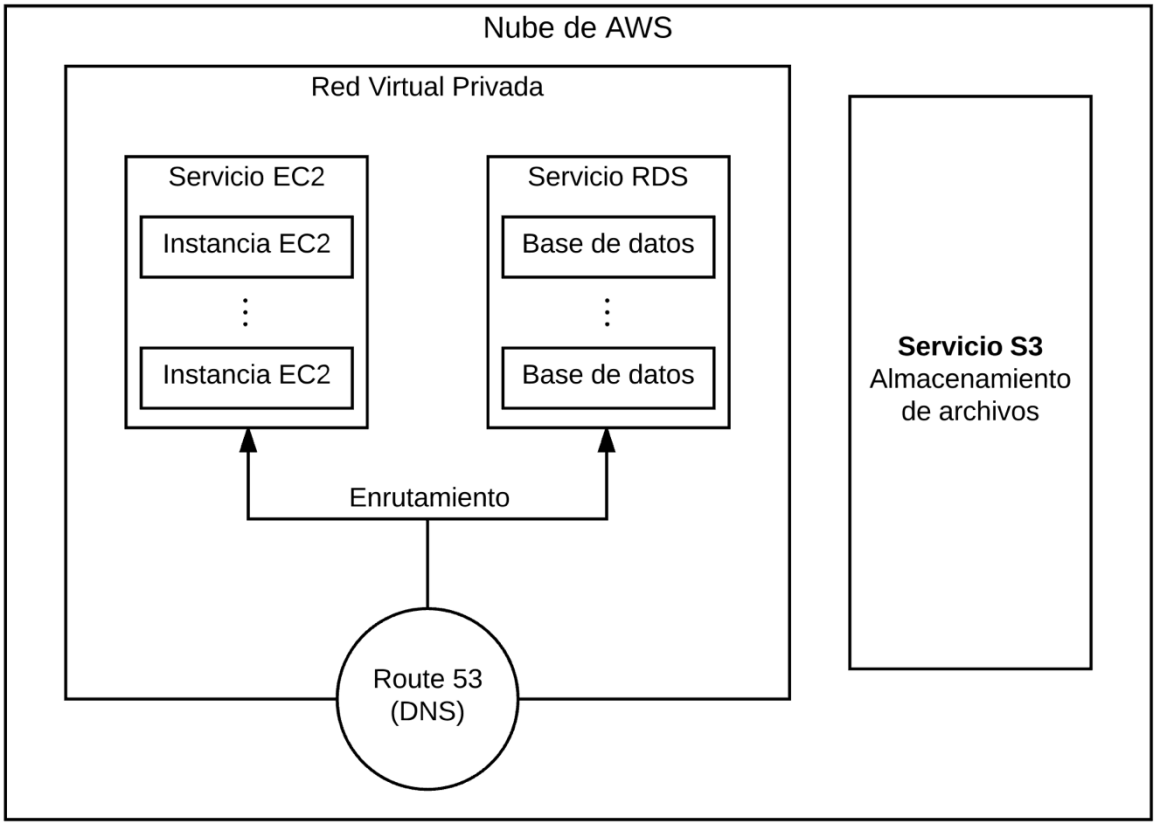

Figura 3. Ejemplo aproximado de la arquitectura interna de AWS

A continuación se profundiza más en las capacidades de cada uno de estos servicios:

Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) provee capacidad de cómputo escalable dentro de AWS. Usando Amazon EC2 elimina la necesidad de invertir en hardware físico y permite desarrollar y desplegar aplicaciones más rápidamente. Amaxon EC2 permite lanzar servidores virtuales según sea necesario teniendo el control sobre las configuraciones de seguridad, red y almacenamiento de cada uno de estos servidores. Amazon EC2 permite escalar según los requerimientos de las aplicaciones.

Las características más importantes de Amazon EC2 son las siguientes:

- Ambientes virtuales de computación, conocidos como *instancias.*
- Plantillas preconfiguradas para las instancias, conocidas como Amazon Machine Images (AMIs), que empaquetan toda la información de un servidor personalizado (incluyendo sistema operativo y software adicional).
- Múltiples configuraciones de CPU, memoria, almacenamiento y capacidad de red para las instancias, conocidas como *tipos de instancias.*
- Acceso seguro a las instancias usando parejas de llaves (AWS guarda las claves públicas y los usuarios de AWS la llave privada en un lugar seguro).
- Volúmenes de almacenamiento para datos temporales que se borran cuando se detenga o termine una instancia, conocidos como *volúmenes de almacenamiento de las instancias.*
- Volúmenes de almacenamiento persistente usando Amazon Elastic Block Store (Amazon EBS), conocidos como *volúmenes EBS de Amazon.*
- Múltiples ubicaciones físicas para los recursos como instancias, y volúmenes EBS, conocidas como *regiones* y *zonas de disponibilidad.*
- Un cortafuegos que permite especificar protocolos, puertos y rangos de IP que están autorizados a acceder a las diferentes instancias, conocidos como *grupos de seguridad.*
- Direcciones IP estáticas para computación dinámica en la nube, conocidas como *direcciones IP elásticas.*
- Metadatos, conocidos como *tags*, que se pueden crear y asignar a los recursos de Amazon EC2.
- Redes virtuales en la nube que permiten aislar recursos de la nube de AWS, conectar unas redes con otras, permitir o no el acceso a internet o conectarlas a una red privada local, conocidas como *nube privada virtual (VPC's* por sus siglas en inglés)

Amazon Relational Database Service (Amazon RDS)

Amazon Relational Database Service (Amazon RDS) es un servicio web que hace más fácil instalar, operar y escalar una base de datos relacional en la nube. Provee un solución costo-efectiva y con capacidad ajustable para las bases de datos relacionales más difundidas de la industria y se encarga de llevar a cabo las tareas más comunes de administración de una base de datos.

Amazon RDS se encarga de muchas de las tareas más difíciles y tediosas de la administración de una base de datos relacional:

- A diferencia de invertir en un servidor donde se recibe CPU, memoria, almacenamiento y Operaciones de Entrada/Salida por segundo (IOPS por sus siglas en inglés), Amazon RDS divide cada una de estas características para permitir escalar cada una de ellas independientemente de las demás. Por ejemplo, si se necesitara más CPU y menos IOPS, o más almacenamiento, fácilmente se puede aumentar lo que se necesite.
- Amazon RDS administra los backups periódicos, los parches de seguridad y actualizaciones, detección automática de fallos y recuperación.
- Para entregar una experiencia plenamente administrada, Amazon RDS no provee acceso a la consola de las instancias de Bases de Datos, y restringe el acceso a ciertos procedimientos del sistema y tablas que requieren privilegios avanzados.
- Se pueden llevar a cabo backups cuando se necesiten o crear una réplica exacta de una base de datos basado en alguno de los backups. Estos backups también pueden ser usados para restaurar o recuperar una base de datos de manera confiable y eficiente.
- Se puede conseguir una alta disponibilidad con una instancia primaria y una secundaria sincronizada donde recurrir en caso de que la instancia principal falle. También es posible crear réplicas de lectura de bases de datos como MySQL, MariaDB o PostgreSQL para aumentar la escalabilidad de las lecturas.

- Se pueden productos de bases de datos ampliamente conocidos como son: MySQL, MariaDB, PostgresSQL, Oracle, Microsoft SQL Server y el nuevo motor, compatible con MySQL Amazon Aurora DB.
- AWS permite el manejo de acceso de usuarios a la base de datos con su servicio IAM (Identity and Access Management) y permite proteger también las bases de datos con los grupos de seguridad y las VPC's.

Amazon Simple Storage Service (Amazon S3)

Amazon Simple Storage Service (Amazon S3) es el almacenamiento para el Internet. Está diseñado para hacer fácil para los desarrolladores la computación a grandes escalas.

Amazon S3 tiene una sencilla interfaz de servicios web que se pueden utilizar para almacenar y acceder cualquier cantidad de datos, en cualquier momento y desde cualquier lugar de la web. Le da a los desarrolladores acceso a toda la infraestructura de almacenamiento de datos altamente escalable, confiable, rápido y de bajo costo que utiliza Amazon para correr su propia red global de sitios web. Este servicio busca maximizar los beneficios de la gran escala y ponerlos a disposición de los desarrolladores.

Amazon S3 fue construido intencionalmente con características mínimas que se concentran en la simplicidad y la robustez. Las principales ventajas de Amazon S3 son las siguientes:

- Crear *Buckets*: Crear y nombrar buckets que almacenan datos. Los buckets son el contenedor fundamental de almacenamiento de datos de Amazon S3.
- Almacenar datos en buckets: Almacenar una infinita cantidad de datos en un bucket. Subir cualquier cantidad de objetos a un bucket de Amazon S3, cada objeto puede contener hasta 5 TB de datos y todos son almacenados y solicitados usando un identificador único creado por el desarrollador.

- Descargar datos: Descargar los datos o permitir a los demás hacerlo. Descargar cualquier cantidad de datos en cualquier momento o permitirle a cualquier persona hacerlo.
- Permisos: Autorizar o negar el acceso a terceros que quieran subir o descargar datos dentro de los buckets de Amazon S3.
- Interfaces estándares: Utilizar interfaces estándares como REST y SOAP, diseñadas para trabajar con cualquier herramienta de desarrollo en Internet.

Amazon Route 53

Amazon Route 53 tiene tres funciones principales:

- **Registro de nombres de dominios:** Amazon Route 53 permite registrar nombres de dominio como 1doc3.com
- **Sistema de Nombres de Dominio (DNS, por sus siglas en inglés):** Amazon Route 53 traduce nombres de dominio amigables como www.1doc3.com a direcciones IP como 192.0.2.1. Responde a queries DNS utilizando una red global de servidores DNS autoritarios que reduce la latencia.
- **Chequeos de salud:** Amazon Route 53 envía solicitudes automáticamente a través de Internet a las aplicaciones para verificar que sean alcanzables, disponibles y funcionales.

Se puede utilizar cualquier combinación de estas funciones. Por ejemplo, se puede utilizar Amazon Route 53 para registrar su nombre de dominio y ser su servicio DNS o se puede utilizar Amazon Route 53 como servicio de DNS para un dominio registrado con otro proveedor de dominios.

2.2 ANÁLISIS DE ARQUITECTURA DE HARDWARE INICIAL DE 1DOC3

La plataforma 1DOC3 contaba con una arquitectura básica de hardware como se puede ver en la *figura 3* donde los usuarios accedían a través de Internet

directamente a una instancia de EC2 asociada a una IP elástica y esta misma asociada a un nombre de dominio a través de Route 53.

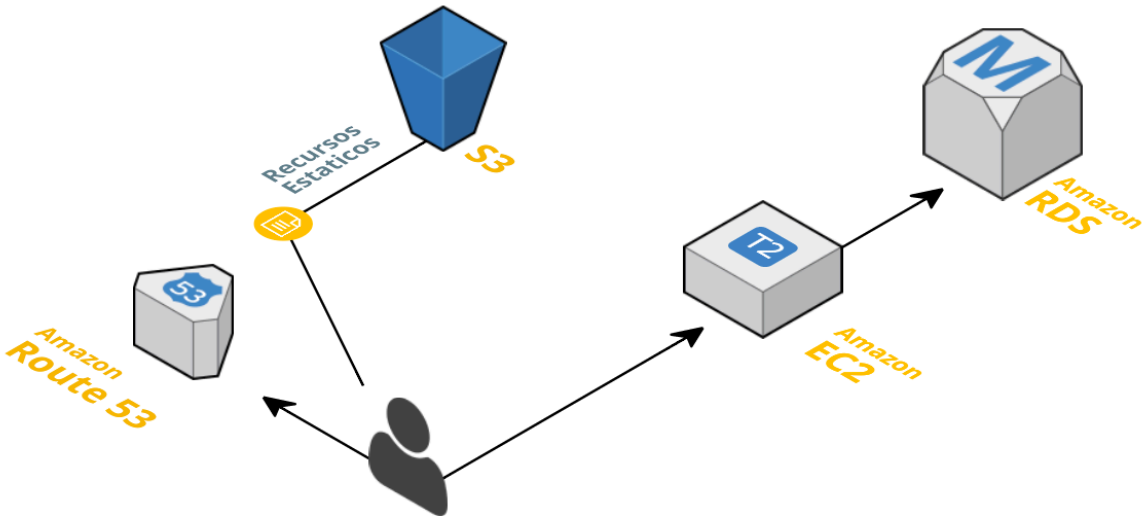

Figura 3. Diagrama de arquitectura inicial de 1DOC3.

La instancia de EC2 contaba con un servidor HTTP Apache que procesaba las solicitudes en conjunto con una aplicación PHP que llevaba a cabo la lógica de la aplicación y se conectaba con una base de datos relacional MySQL alojada en Amazon RDS para su funcionamiento. Los recursos estáticos tales como imágenes, archivos, etc. eran entregados directamente desde Amazon S3.

Luego de analizar la arquitectura anteriormente expuesta se concluyó que el problema más relevante y urgente de solucionar era **la imposibilidad de escalamiento automático ante un posible aumento sustancial de tráfico en un tiempo reducido.** Con la actual arquitectura no existía capacidad de reacción ante una situación tan común como esta debido a que:

- **No había manera de aumentar el número de servidores respondiendo a los usuarios ya que no se contaba con balanceadores de carga**, creando un cuello de botella en el procesamiento de las peticiones en la única instancia EC2.

- **La base de datos principal estaba siendo usada tanto para lectura como escritura** en todo momento y no se llevaba a cabo el uso de caché para las consultas, lo que hacía la sobrecarga de la instancia EC2 y/o la base de datos de RDS dos muy probables escenarios ante un aumento considerable de la carga en el sistema. En la *figura 4* se muestra la evidencia del aumento desproporcionado de uso de CPU en la base de datos ante un pico de tráfico provocado por un anuncio televisivo en Perú que llevó a un aumento en la latencia del tiempo de respuesta de la base de datos.

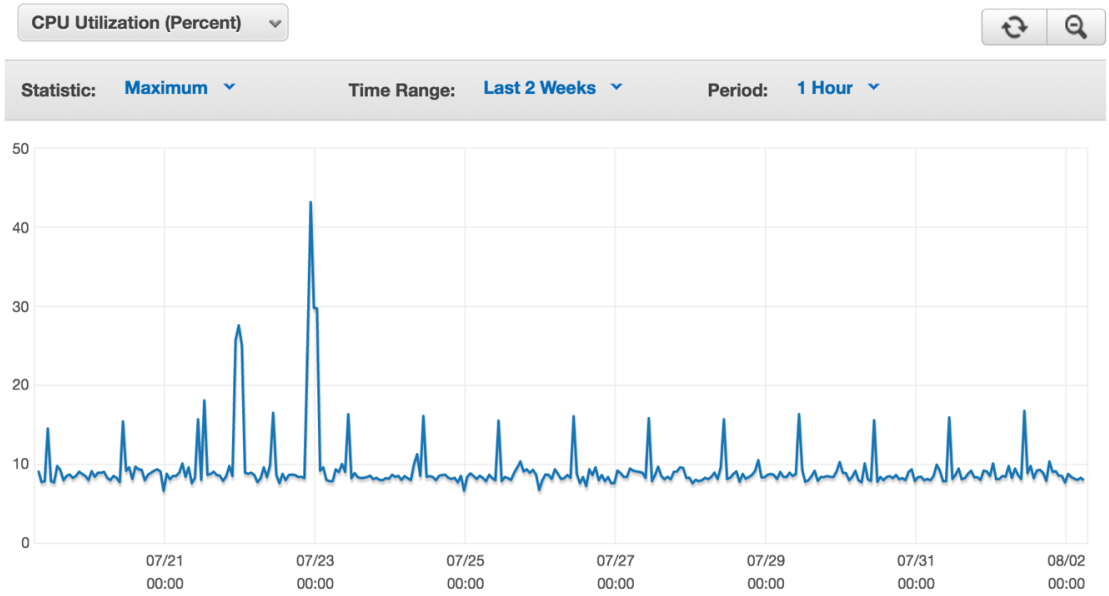

Figura 4. Gráfica de uso de CPU en la base de datos principal de 1DOC3

2.3 ARQUITECTURA DE SOFTWARE INICIAL

La plataforma 1DOC3 fue pensada desde un principio para que sobre ella trabajaran múltiples desarrolladores, por lo que se inició con las siguientes herramientas y lenguajes:

- **Lenguaje PHP:** Es un lenguaje de código abierto ampliamente difundido y especialmente adecuado para el desarrollo web. Puede ser incrustado dentro del lenguaje HTML y se destaca por su velocidad y su

comunidad. Al ser un lenguaje maduro, cuenta con todas las facilidades para llevar a cabo conexiones a bases de datos y motores de caché y posee una gran diversidad de Frameworks para un desarrollo más rápido.

- **CodeIgniter Framework**: Es un framework para desarrollo de aplicaciones que utilizan PHP como su lenguaje de programación. Su objetivo es agilizar el desarrollo poniendo a disposición de los desarrolladores una gran variedad de librerías para tareas comunes sin ser muy invasivo en la aplicación.
- **Modelo Vista Controlador de CodeIgniter:** CodeIgniter está basado en el patrón de arquitectura Modelo Vista Controlador (MVC). En la práctica, permite a las aplicaciones web tener un código mucho más legible y sencillo ya que separa la presentación (Vistas) de la lógica de la aplicación (Controladores y Modelos).

Con estas herramientas y principios fue construída la primera versión de la plataforma 1DOC3, por lo que a continuación se profundiza más en el lenguaje, framework y patrón de arquitectura de software utilizados.

2.3.1 Lenguaje PHP

PHP (acrónimo recursivo de PHP: Hypertext Preprocessor) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML.

Lo que distingue a PHP de algo del lado del cliente como JavaScript es que el código es ejecutado en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá el resultado de ejecutar el script, aunque no se sabrá el código subyacente que era. El servidor web puede ser configurado incluso para que procese todos los ficheros HTML con PHP, por lo que no hay manera de que los usuarios puedan saber qué se tiene debajo de la manga.

2.3.2 CodeIgniter Framework

CodeIgniter es un framework para desarrollo de aplicaciones que utilizan PHP como su lenguaje de programación. Su objetivo es agilizar el desarrollo poniendo a disposición de los desarrolladores una gran variedad de librerías para tareas comunes, así como una interfaz simple y una estructura lógica para acceder a ellas. CodeIgniter permite concentrarse creativamente en el desarrollo de un proyecto minimizando la cantidad de código necesaria para llevar a cabo alguna actividad.

CodeIgniter se ha caracterizado entre la comunidad mundial por ser muy ligero y poco invasivo, es decir, su funcionamiento básico requiere sólo muy pocas librerías pequeñas contrario a otros frameworks que requieren muchos recursos para ser utilizados. Por esto mismo se destaca también por su alto rendimiento y velocidad.

CodeIgniter utiliza el patrón de arquitectura de software Modelo Vista Controlador (MVC) que permite la separación de la lógica de la aplicación de la presentación. Esto permite mantener un código fuente muy organizado, permitiendo el trabajo de múltiples desarrolladores simultáneamente y haciendo el código fuente más mantenible.

Flujo de una aplicación de CodeIgniter

En la *Figura 5* se ilustra el flujo de la información a través de una aplicación desarrollada con CodeIgniter:

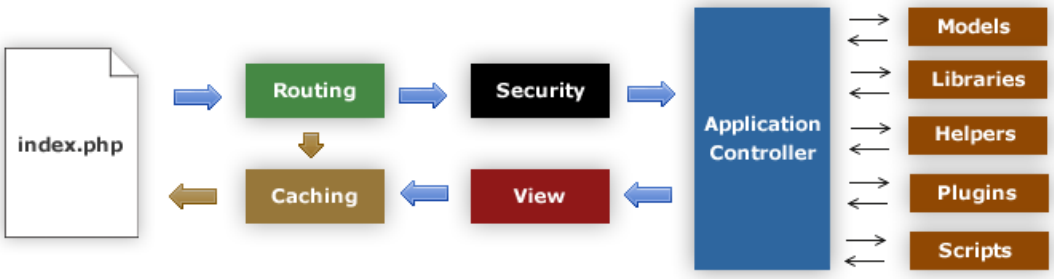

Figura 5. Flujo de una aplicación de CodeIgniter

1. El archivo index.php sirve como controlador de entrada, inicializando los recursos básicos necesarios para ejecutar CodeIgniter.
2. El enrutador o *Router* examina las peticiones HTTP para determinar qué se debe hacer con ellas.
3. Si existe algún archivo de caché, lo envía directamente al navegador, sin llevar a cabo la ejecución normal del sistema.
4. Seguridad. Antes de cargar el controlador de la aplicación, la petición HTTP y toda la información ingresada por los usuarios es filtrada por seguridad.
5. El controlador llama los modelos, librerías base, *helpers* y cualquier otro recurso necesitado para procesar cada petición específicamente.
6. Finalmente el resultado y la información es renderizada en una Vista y enviada al navegador web para ser visualizada. De estar habilitado el caché, la vista primero es guardada en caché para responder a las siguientes peticiones similares.

2.3.3 Modelo Vista Controlador de CodeIgniter

CodeIgniter está basado en el patrón de arquitectura Modelo Vista Controlador (MVC). MVC es un patrón de arquitectura de software que separa la lógica de la aplicación de su presentación. En la práctica, permite a las aplicaciones web tener un código mucho más legible y sencillo ya que separa la presentación (Vistas) de la lógica de la aplicación (Controladores y Modelos).

- El **Modelo** representa las estructuras de datos de la aplicación. Normalmente las clases en los modelos contienen funciones que ayudan consultar, insertar y actualizar información de la base de datos.
- La **Vista** la información que es presentada al usuario. Normalmente una vista contiene toda la presentación (HTML) de la aplicación web, pero en CodeIgniter, una vista también puede ser un fragmento de página como una cabecera o un pie de página.

- El **Controlador** sirve de intermediario entre el Modelo, la Vista y cualquier otro recurso necesario para procesar una petición HTTP y generar la página web.

CodeIgniter tiene un enfoque con mucha libertad hacia el patrón MVC ya que los modelos no son requeridos. De no necesitar esta separación, o si no se desea utilizar los modelos ya sea por que se crea más complejo o difícil de mantener, se puede ignorar su utilización y construir la aplicación utilizando únicamente Controladores y Vistas. CodeIgniter también permite incorporar scripts propios o incluso añadir librerías bases desarrolladas por uno, dando flexibilidad para trabajar de la manera en que uno encuentre adecuada.

2.4 ANÁLISIS DE ARQUITECTURA DE SOFTWARE INICIAL DE 1DOC3

La arquitectura de software inicial si bien utilizaba patrones de arquitectura de software óptimos y adecuados para la aplicación, se contaba con ciertas limitaciones a nivel de software que dificultaban la escalabilidad de la aplicación y al mismo tiempo 1DOC3 necesitaba darle a sus usuarios funcionalidades adicionales para mejorar su experiencia y seguridad. A continuación se describen los problemas encontrados:

- **Almacenamiento de archivos en servidor local.** Una de las funcionalidades de la aplicación consiste en permitir a los doctores registrados cargar una foto de perfil para que los usuarios los reconozcan a través de la aplicación; a pesar de que esta característica era plenamente funcional, el hecho de almacenar los archivos en el servidor hacía a la aplicación perder su capacidad de ser replicada, es decir, poder ser ejecutada en diferentes servidores simultáneamente sin depender de el servidor local.
- **Uso de sesiones basadas en archivos locales.** El manejo de sesiones de los usuarios, si bien se hacía de manera segura y funcional, se había optado por manejarlo mediante archivos locales que guardaban la información de las sesiones de los usuarios y se

almacenaban en el servidor, lo que, nuevamente, limitaba la replicabilidad de la aplicación.

- **Ausencia de caché para las consultas en la base de datos y la no utilización de réplicas de lectura.** No se llevaba a cabo ninguna de estas prácticas que ayudarían a mejorar el rendimiento de toda la aplicación. Si bien esto requería una adaptación del hardware de la aplicación, se debía hacer de la mano del software para manejar el caché y las réplicas de lectura de manera óptima.
- **Falta de implementación del protocolo SSL en las peticiones.** No se estaba llevando a cabo la encriptación extremo a extremo en la comunicación entre los clientes y el servidor, lo que daba lugar a vulnerabilidades de seguridad para los usuarios y, siendo 1DOC3 una aplicación con información de salud que puede ser sensible, era de alta prioridad implementar esta funcionalidad.
- **Necesidad de crear un perfil destacado para los doctores que lo adquirieran.** Al ser 1DOC3 una aplicación y empresa tan joven, sus requerimientos funcionales cambiaban muy rápido, por lo que esta funcionalidad fue descartada por la empresa ya que decidieron enfocar su negocio a empresas del sector salud y no en dar la posibilidad de tener perfiles destacados a los usuarios.

3. DISEÑO Y DESARROLLO DE NUEVAS FUNCIONALIDADES

La nueva arquitectura propuesta para 1DOC3 requiere el uso de servicios adicionales de AWS para ser llevada a cabo, como son:

- **Amazon Elastic Load Balancing (Amazon ELB):** Es un servicio de balanceo de carga administrado por AWS, permite balancear la carga de una aplicación a través de un conjunto de instancias EC2 para permitir escalar las aplicaciones más fácilmente.
- **Amazon Auto Scalling:** Es una configuración asociada a un grupo de instancias de EC2 que permite aumentar o disminuir automáticamente la cantidad de servidores virtuales EC2 asociadas a un balanceador de carga para aumentar la capacidad de respuesta de la aplicación.
- **Amazon ElastiCache:** Es un servicio de almacenamiento en memoria administrado por AWS, permite crear clústeres de nodos de caché de los motores más comunes de caché como son Redis y Memcache.

A continuación se da información más detallada de cada uno de estos servicios utilizados para la nueva arquitectura de 1DOC3.

Amazon Elastic Load Balancing (Amazon ELB)

Amazon Elastic Load Balancing (Amazon ELB) distribuye automáticamente el tráfico entrante a través de múltiples instancias EC2. AWS permite crear un balanceador de carga y registrar instancias en múltiples zonas de disponibilidad a este balanceador. El balanceador de carga pone a disposición de los clientes un único punto de acceso. Esto permite incrementar la disponibilidad de la aplicación desarrollada. Puedes agregar y remover instancias de EC2 de los balanceadores de carga según sea necesario sin interrumpir el flujo normal de la información. Si una instancia de EC2 falla, Amazon ELB redistribuye el tráfico automáticamente hacia las instancias restantes. Si una instancia de EC2 se recupera tras un fallo, Amazon ELB

restable el tráfico a esa instancia. Amazon ELB también sirve como primera línea de defensa ante ataques a las redes privadas. Amazon ELB permite despreocuparse del trabajo de encriptación y desencriptación ya que los balanceadores se encargan de esto y las instancias de EC2 se pueden enfocar en su trabajo principal.

Amazon ELB provee las siguientes características:

- Se pueden utilizar los sistemas operativos y tipos de instancias soportados por Amazon EC2. Se puede configurar las instancias de EC2 para aceptar tráfico únicamente proveniente del balanceador de carga.

- Se puede configurar el balanceador de carga de manera que acepte tráfico de los siguientes protocolos: HTTP, HTTPS (HTTP seguro), TCP y SSL (TCP seguro).

- Amazon ELB permite configurar los balanceadores de carga para que distribuyan el tráfico entre instancias en múltiples zonas de disponibilidad, minimizando el riesgo de sobrecarga en una única instancia. Si una zona de disponibilidad completa deja de estar disponible, el balanceador envía el tráfico a instancias en otras zonas de disponibilidad.

- No hay un límite de conexiones que un balanceador de carga puede intentar crear con instancias de EC2. El número de conexiones crece a medida que crecen las peticiones concurrentes a el balanceador de carga.

- Amazon ELB puede ser configurado para llevar a cabo chequeos de salud de las instancias de EC2 registradas a un balanceador para así enviar tráfico únicamente a instancias plenamente funcionales.

- Es posible utilizar encriptación de extremo a extremo en redes que utilizan conexiones seguras (HTTPS/SSL).

- Se pueden crear balanceadores de carga públicos en Internet para balancear tráfico proveniente de clientes en cualquier parte del mundo y enrutar el tráfico a las instancias de EC2, o crear balanceadores de carga internos que permitan balancear tráfico de clientes dentro de una VPC y distribuirlo entre instancias de EC2 en redes privadas.

- AWS permite monitorear los balanceadores de carga a través de CloudWatch y analizar los logs de acceso.

- Se pueden asociar los balanceadores de carga públicos en Internet con nombres de dominio. Gracias a que el balanceador recibe todas las peticiones de los clientes, no se necesita crear y administrar los nombres de dominio públicos para las instancias de EC2. De esta manera el nombre de dominio está únicamente asociado al balanceador de carga permitiendo mayor flexibilidad en las instancias de EC2 añadiendo mayor capacidad (o eliminando) de ser necesario sin necesidad de cambios en la configuración en cada cambio.

Amazon Auto Scalling

Amazon Auto Scalling ayuda a mantener el número correcto de instancias de EC2 disponibles para manejar la carga de una aplicación. Permite crear colecciones de instancias de EC2 llamados *grupos de Auto-escalamiento*. Se puede configurar el número mínimo y máximo de instancias en cada grupo de auto-escalamiento y configurar Auto Scalling con políticas de escalamiento para que inicie o termine instancias según la aplicación lo requiera.

Amazon ElastiCache

ElastiCache es un servicio web que facilita la instalación, administración y escalamiento de un ambiente distribuido de caché en memoria en la nube. Provee una solución costo-efectiva de caché de alto rendimiento y escalabilidad, eliminando la complejidad de desplegar y administrar un ambiente de caché distribuido.

Con ElastiCache se pueden desplegar ambientes de caché distribuídos, sin necesidad de preocuparse por provisionar el hardware o instalar el software. Se puede escoger entre Memcached o Redis como motores de caché y dejar que ElastiCache lleve a cabo tareas de administración como actualizaciones de software y parches de seguridad.

ElastiCache tiene múltiples características para aumentar su confiabilidad para ambientes de producción críticos:

- Detección y recuperación automática de fallos de los nodos de caché.
- Apoyo automático en réplicas de lecturas ante fallos del clúster principal.
- Creación flexible de nodos en diferentes zonas de disponibilidad.
- Integración con múltiples servicios de AWS como Amazon EC2, CloudWatch, CloudTrail y Amazon SNS para proveer una solución de caché en memoria distribuida de alto rendimiento, administrada y segura.

3.1 NUEVOS REQUERIMIENTOS DE LA PLATAFORMA

Debido a las necesidades y falencias que se encontraron en la arquitectura tanto de hardware como de software inicial de 1DOC3, se analizaron las diferentes opciones disponibles y se presentaron al cliente las siguientes propuestas, que, vale la pena mencionar, fueron aceptadas e implementadas durante la práctica empresarial.

Con base en análisis de requerimientos de hardware y la detección de los posibles cuellos de botella en el flujo de la aplicación, se propuso la arquitectura expuesta en la *figura 6.*

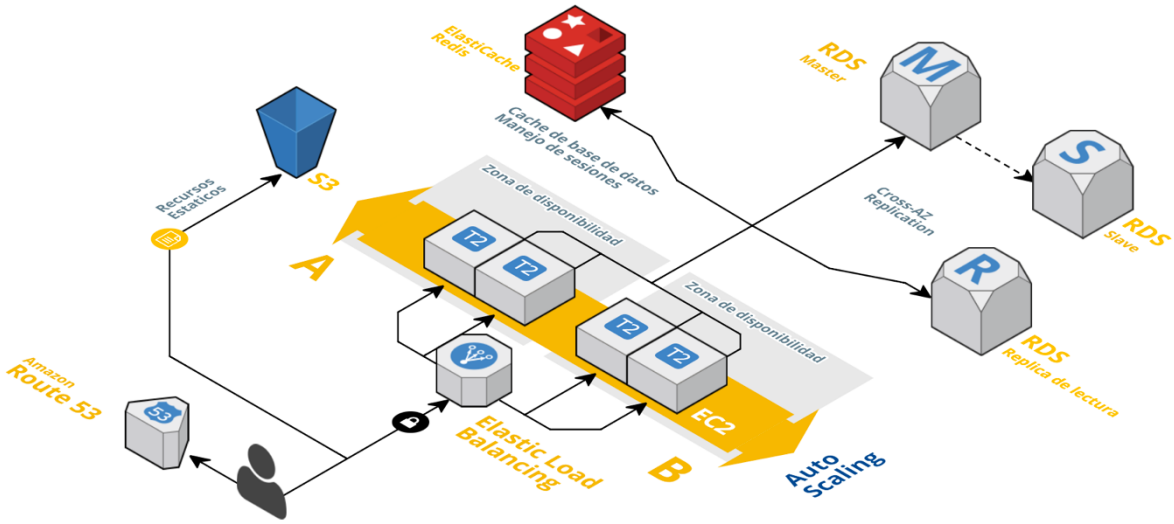

Figura 6. Propuesta de arquitectura de hardware para 1DOC3

Esta arquitectura consta de cuatro nuevos componentes de hardware cada uno con una finalidad específica:

1. **Balanceador de carga.** Se propuso un balanceador de carga de Amazon ELB para solucionar dos problemas: el primero la **falta de implementación del protocolo SSL en las peticiones**, que, gracias a Amazon ELB puede ser llevada a cabo entre el usuario final y el balanceador de carga sin necesidad de modificar la aplicación ni de agregarle ese procesamiento a las instancias de EC2 que se dedicarían únicamente al procesamiento de la lógica de la aplicación y el segundo, solucionar el problema de que **no había manera de aumentar el número de servidores respondiendo a los usuarios ya que no se contaba con balanceadores de carga**.

2. **Grupo de auto-escalamiento.** Se propuso un grupo de auto-escalamiento de Amazon Auto Scalling para también solucionar el problema de que **no había manera de aumentar el número de servidores respondiendo a los usuarios ya que no se contaba con balanceadores de carga** y evitar la intervención manual a la hora de aumentar o disminuir el número de instancias de EC2 que procesarán las solicitudes enviadas por el balanceador de carga. Este grupo de auto-escalamiento contaría con mínimo dos instancias de EC2 en diferentes zonas de disponibilidad para darle a 1DOC3 una alta disponibilidad incluso en caso de comprometerse una zona de disponibilidad y un máximo no determinado para escalar según sea necesario.

3. **Réplica de lectura de la base de datos.** Se propone añadir una réplica de lectura de la base de datos MySQL porque **la base de datos principal estaba siendo usada tanto para lectura como escritura.** Con esta réplica se busca distribuir la carga de la base de datos, ejecutando únicamente las escrituras en la base de datos principal y enviando todas las lecturas a una réplica de lectura. Adicionalmente se propone crear una réplica síncrona de la base de datos principal en otra zona de disponibilidad en caso de fallo de la base de datos principal, para lograr así una arquitectura de base de datos mucho más robusta y de alta disponibilidad.

4. **Clúster de caché en memoria de Redis.** Se propone crear un clúster de caché en memoria con el motor Redis administrado por el servicio Amazon ElastiCache debido a la **ausencia de caché para las consultas en la base de datos y la necesidad de elminar el uso de sesiones basadas en archivos locales** con el manejo de sesiones en caché y el caché de las lecturas más comunes en la base de datos y de las consultas que por la naturaleza de la aplicación pueden ser ejecutadas miles de veces en muy poco tiempo, lo que el caché en

Redis haría que la base de datos sólo deba ejecutar esas consultas una vez y las demás sean entregadas por el caché en memoria de Redis que es mucho más eficiente por su misma naturaleza.

3.2 DESARROLLO DE LA SOLUCIÓN PARTE HARDWARE

3.2.1 Creación y configuración del balanceador de carga

La creación del balanceador de carga se llevó a cabo a través de la consola de administración de AWS, que permite su fácil configuración y visualización del progreso.

El balanceador de carga fue configurado para recibir las peticiones a través del protocolo HTTPS (HTTPS seguro), manteniendo la comunicación con las instancias dentro de la VPC configurada a través del puerto 80, es decir el protocolo HTTP para evitar la adaptación de la aplicación a este protocolo y el procesamiento que requiere la encriptación en las instancias de EC2.

Amazon ELB permite asociar un certificado SSL con un balanceador de carga de varias maneras:

1. A través de su servicio AWS Certificate Manager (ACM)
2. A través de su servicio AWS Identity and Access Management (IAM), o
3. Cargando un certificado SSL ya existente a IAM

AWS Certificate Manager (ACM) permite cargar un certificado ya existente a su servicio, y, al 1DOC3 ya tener un certificado SSL válido, se optó por cargar este dentro de ACM y asociarlo al balanceador de carga.

Para garantizar la disponibilidad se configuró un chequeo de salud cada 30 segundos de las instancias de EC2 asociadas al balanceador de carga, al

recibir dos chequeos fallidos por parte de alguna instancia de EC2, el balanceador de carga dejará de enviar tráfico a esa instancia hasta recibir 10 chequeos consecutivos saludables, lo que representa que ha vuelto a la normalidad.

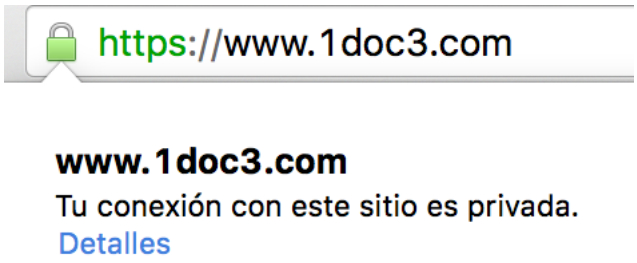

Figura 7. Conexión segura de usuarios a 1DOC3.

Como resultado de esta implementación se logró crear conexiones seguras para los usuarios de 1DOC3 como se muestra en la *figura 7.*

Configuración	Valor
Protocolo de recepción	HTTPS
Puerto recepción	443
Protocolo comunicación con instancias	HTTP
Puerto comunicación con instancias	80
Frecuencia de chequeo de salud	30 segundos
Chequeos fallidos consecutivos para inhabilitar una instancia	2
Chequeos exitosos para habilitar una instancia	10

Tabla 1. Configuración aplicada al balanceador de carga.

3.2.1 Creación y configuración del grupo de auto-escalamiento

Inicialmente se creó una imagen exacta de la instancia EC2 en la que funcionaba 1DOC3, ya que esta imagen tenía el sistema operativo, extensiones, librerías y archivos utilizados por la aplicación de 1DOC3. Esta imagen lo que nos permite es poderla tener de referencia a la hora de crear nuevas instancias en el grupo de auto-escalamiento.

Configuración	Valor
Sistema operativo	Linux
Distribución	Amazon linux
Servidor HTTP	Apache
Otras	PHP 5.6, ImageMagick, Git

Tabla 2. Configuración de la imagen creada

Una vez creada esta imagen se crea una configuración de creación escogiendo esta imagen como punto de partida para la creación de nuevas instancias futuras.

El siguiente paso fue crear y configurar el grupo de auto-escalamiento con un tamaño inicial de 2 instancias, ambas dentro de una red privada y cada una en una subred diferente en diferentes zonas de disponibilidad, así como también configurar el grupo para que las instancias reciban el tráfico del previamente creado balanceador de carga. Los detalles de la configuración se pueden observar en la *tabla 3.* Igualmente se configuraron las alarmas que llevan a cabo el auto-escalamiento, se decidió tomar como métrica decisiva el uso de CPU por encima del 90% en las instancias para escalar hacia arriba y el uso de CPU por debajo del 5% para escalar hacia abajo.

Configuración	Valor
Cantidad de instancias mínima	2
Zonas de disponibilidad	2
Instancias mínimas por zona de disponibilidad	1
Métrica para escalar hacia arriba	Uso de CPU
Valor máximo para escalar hacia arriba	90%
Métrica para escalar hacia abajo	Uso de CPU
Valor mínimo para escalar hacia abajo	5%

Tabla 3. Configuración del grupo de auto-escalamiento

Lo anterior basado en el comportamiento pasado de las instancias en inconvenientes pasados. Las otras opciones como el uso de red de entrada o

de salida no eran precisas a la hora de saber si la aplicación necesitaba escalar o no.

Finalmente configuraron los tags que se utilizaron en las instancias de este grupo que se optó por crear ambas instancias iniciales desde cero en paralelo al sistema actual para llevar a cabo pruebas antes de cambiar el sistema de producción del anterior al nuevo.

3.2.2 Creación de una réplica de lectura de la base de datos

La creación y configuración de la réplica de lectura en AWS fue bastante sencilla pues Amazon RDS lleva a cabo la administración de este proceso, la complejidad de este punto estuvo relacionada con la adaptación del software, por lo que se mirará más en detalle en la implementación de software este punto.

La creación se llevó a cabo a través de la consola de administración de AWS, que permite crear réplicas de lecturas de bases de datos en producción sin interrumpir su disponibilidad, permitiendo configurar desde el tamaño, hasta el tipo de almacenamiento utilizado. En la *tabla 4* se muestra la configuración utilizada para este caso.

Configuración	**Valor**
Motor de base de datos	MySQL 5.6.19b
Puerto	3360
CPU virtuales	2
Memoria	4 GB
Almacenamiento	100GB

Tabla 4. Configuración de la réplica de lectura de la base de datos

La modificación de la base de datos existente de manera que tenga una réplica sincrónica en otra zona de disponibilidad a la cual acceder en caso de fallo de esta se hizo de manera similar a través de la consola de

administración de AWS. Este proceso de modificación requirió el reinicio de la instancia de base de datos, al ser inevitable el tiempo de interrupción del servicio, se dio aviso a los usuarios activos de 1DOC3 de la interrupción y se llevó a cabo durante las horas de menor carga en la plataforma. Este proceso no duró más de 15 minutos en ser realizado, lo que es un tiempo aceptable teniendo en cuenta que la gran característica que se adicionó a la plataforma.

3.2.3 Creación de clúster de caché en memoria

La creación de este clúster de caché en memoria, al ser ElastiCache un servicio administrado por AWS, también es llevado a cabo a través de la consola de administración y AWS se encarga de todo su despliegue y mantenimiento.

Configuración	Valor
Motor de caché	Redis
Versión	2.8.23
Número de nodos de caché	2
Puerto	6379
Memoria	1.55GB

Tabla 5. Configuración de clúster de caché en memoria

En la *tabla 5* se muestra la configuración de el clúster creado en la consola de administración y AWS facilita todas las demás tareas.

El resultado de este proceso da como resultado una dirección de red de la red privada a través del cual se pueden conectar las instancias dentro de la red privada de 1DOC3 al servicio de caché en memoria, tanto como para escribir como para leer. Nuevamente, la mayor complejidad de este despliegue es a nivel de software, por lo que se hablará más en detalle en la implementación de software.

3.3 DESARROLLO DE LA NUEVA SOLUCIÓN PARTE SOFTWARE

Si bien la arquitectura de software se mantuvo con el patrón MVC, se llevaron a cabo múltiples cambios y desarrollaron nuevas funcionalidades basadas en el análisis de requerimientos tanto de hardware como de software.

Con base en el análisis de requerimientos, se propusieron los siguientes cambios y adaptaciones al software:

1. **Desarrollo de subida de archivos escalable.** Se propuso almacenar los archivos e imágenes subidas por los usuarios/doctores en Amazon S3 con el fin de permitir la replicabilidad que **almacenamiento de archivos en servidor local** eliminaba de la aplicación.

2. **Desarrollo de sesiones con Redis.** Adicionalmente a la creación del clúster de caché, se adaptó la aplicación para eliminar el **uso de sesiones basadas en archivos locales** y permitir la replicabilidad de la aplicación**.**

3. **Desarrollo del uso de caché de Redis para optimizar las consultas a la base de datos.** También adicionalmente a la creación del clúster de caché, se adapta la aplicación para que se utilice este para consultas a la base de datos que no varían mucho a través del tiempo. Lo anterior para solucionar la **ausencia de caché para las consultas en la base de datos.**

4. **Desarrollo del uso de réplicas de lectura de base de datos.** Para hacer uso de la réplica de lectura de la base de datos creada anteriormente, se requiere una adaptación del software de la aplicación con el fin de reducir la carga a la base de datos principal y distribuir la carga de lecturas en la réplica.

3.3.1 Desarrollo de subida de archivos escalable

Con el fin de eliminar la dependencia local de los servidores de aplicación de la funcionalidad de subir archivos de imagen, se propone el siguiente flujo de aplicación para esta funcionalidad:

1. Subida inicial de archivos al servidor de aplicación en una carpeta temporal.
2. Llevar a cabo el procesamiento de las imágenes en el servidor de aplicación (optimización de tamaño, creación de copias en tamaños utilizados en el resto de la aplicación, etc.).
3. Enviar la o las imágenes resultantes al almacenamiento escalable Amazon S3.
4. Crear una referencia en la base de datos hacia la imagen guardada y sus respectivos tamaños.

De esta manera lo archivos de imagen son alojados en un almacenamiento externo, de alta disponibilidad y sin dependencia de ningún servidor de aplicación, lo que nos permite replicar la aplicación y servirla desde múltiples instancias EC2 a la vez.

Para esta funcionalidad se utilizó el kit de desarrollo de software (SDK por sus siglas en inglés) para PHP de Amazon, que permite acceder a librerías que abstraen la interacción con los servicios de AWS, entre ellos Amazon S3.

Toda la lógica de la recepción de las imágenes se mantuvo debido a sus buenos resultados y su validación en un ambiente de producción. Lo que se adicionó fue un nuevo helper de CodeIgniter con funciones que abstraen toda la interacción entre el servidor y Amazon S3 a la hora de subir los archivos, por lo que luego del flujo normal de la aplicación que guardaba localmente los archivos, lo único que se necesitaba adicionar en el código fuente era la carga de dicho helper y el llamado a la función para enviar las imágenes a Amazon S3.

```php
<?php
require APPPATH.'third_party/aws/aws-autoloader.php';
use Aws\S3\S3Client;

if(!function_exists('subirArchivo'))
{
    function subirArchivo($bucket,$nombre,$archivo){

        $client = S3Client::factory(
                                array(
                                  'version' => '2006-03-01',
                                  'region' => 'us-east-1'
                                )
                            );

        $result = $client->putObject(array(
            'Bucket'     => $bucket,
            'Key'        => $nombre,
            'SourceFile' => $archivo,
            'ACL'          => 'public-read',
            'StorageClass' => 'REDUCED_REDUNDANCY'
        ));

        return $result;
    }
}
```

Figura 8. Fragmento de código de envío de archivos a Amazon S3

En la *Figura 8* se observa el comienzo del helper que fue creado para la subida de archivos a Amazon S3, se reciben como parámetros el nombre con el que se quiere guardar en Amazon S3 el archivo, el bucket en el cual se va a guardar y la referencia al archivo local en el servidor. Esto permite que luego del procesamiento de las imágenes, subirlas sólo requiere, como se muestra en la *figura 9*, cargar el helper y llamar la función.

```php
// lógica de procesamiento de imagen
$this->load->helper('s3');
subirArchivo($bucket,$nombre,$archivo);
```

Figura 9. Fragmento de código para subir un archivo a Amazon S3 desde cualquier modelo o controlador

Finalmente para acceder a estas imágenes posteriormente, se tenía la referencia a ellas en la base de datos, por lo que cualquier servidor podía o

utilizar estas imágenes o bien traer el link público en Amazon S3 para mostrarla en el navegador.

3.3.2 Desarrollo de sesiones con Redis

Las sesiones utilizadas en CodeIgniter por defecto son almacenadas en archivos locales en el servidor de aplicación, lo que no permite compartir la sesión del usuario a través de múltiples instancias de EC2, es decir, no permite a la aplicación ser replicable.

Para esto se tenían dos opciones, el almacenamiento de las sesiones en base de datos, o el almacenamiento en Redis, cada una con sus ventajas y desventajas, pero se optó por Redis por los siguientes motivos:

1. Mayor rendimiento. Redis, por ser almacenamiento en memoria, es mucho más eficiente, por lo que su rendimiento es mayor comparado con el almacenamiento en una base de datos.
2. No se debía añadir más carga a la base de datos actual de 1DOC3 ya que en ese momento ya era un problema el rendimiento de la base de datos, por lo que hubiera sido necesario crear una base de datos dedicada sólo a sesiones y como se planeaba implementar un clúster de caché, por lo que este mismo se podría utilizar para las sesiones y disminuir los costos.

Afortunadamente CodeIgniter abstrae la lógica de las sesiones de su uso en la aplicación, por lo que el cambio en el código fuente no era muy invasivo, sólo requería de la creación del clúster de caché y su configuración (que también es abstraída por CodeIgniter) ya que CodeIgniter posee también un driver para conectarse a Redis, ya sea para uso de sesiones o simple caché.

3.3.3 Desarrollo del uso de caché de Redis para optimizar las consultas a la base de datos

Si bien las consultas a la base de datos de 1DOC3 eran optimas para traer la información necesaria y se utilizaban muy bien las ventajas de los índices de MySQL para optimizar el rendimiento, se estaban llevando a cabo queries innecesarios.
Por ejemplo, la consulta más frecuente a la base de datos de 1DOC3 era traer la información de una pregunta con su respectiva respuesta y doctor que respondió. Este query necesitaba acceder a dos tablas y, si bien era óptimo, era un query que en la escala de 1DOC3 generaba una carga alta a la base de datos. Este query no sólo era ejecutado cada vez que se necesitaba mostrar una pregunta, sino que cada vez que se necesitaba mostrar una misma pregunta, se accedía nuevamente a la base de datos, por lo que había aquí una gran posibilidad de optimización.

Una vez un doctor responde una pregunta en 1DOC3, esta queda accesible públicamente para que los demás usuarios puedan aprovechar esta respuesta, por lo que muchas veces, una misma pregunta era accedida por muchos usuarios por cuestiones de tendencias, redes sociales, etc. y la información de una pregunta, luego de ser respondida, cambia muy poco a través del tiempo, por lo que guardar el resultado de este query en caché era la opción que encajaba perfecta en este problema.

Al CodeIgniter contar con un driver para Redis, hizo su implementación muy sencilla. De nuevo, se optó por crear un helper que llevara a cabo toda la lógica del almacenamiento del caché y la comprobación como se muestra en la *figura 10.*

```php
if(!function_exists('getPreguntaCache'))
{
    function getPreguntaCache($id){
        $CI = get_instance();
        $CI->load->driver('cache',array('adapter'=>'redis'));

        $row = $CI->cache->redis->get('question_'.$id);
        if($row !== FALSE){
            return json_decode($row);
        }else{
            return false;
        }
    }
}

if(!function_exists('setPreguntaCache'))
{
    function setPreguntaCache($id,$row){
        $CI = get_instance();
        $CI->load->driver('cache',array('adapter'=>'redis'));

        $CI->cache->redis->save('question_'.$id , json_encode($row) , 60*60*24 );
    }
}
if(!function_exists('clearPreguntaCache'))
{
    function clearPreguntaCache($id){
        $CI = get_instance();
        $CI->load->driver('cache',array('adapter'=>'redis'));

        $CI->cache->redis->delete('question_'.$id);
    }
}
```

Figura 10. Fragmento de código con las funciones creadas para almacenar, consultar y limpiar el caché.

Gracias a este helper, el cambio en el modelo que trae la información de la pregunta es muy pequeño, de una lógica simple y fácil de comprender como se muestra en la *figura 11.*

```php
$row = getPreguntaCache($id);
if($row == FALSE){
    $this->db->where('id', $id);
    $query = $this->db->get('preguntas', 1, 0);
    if($query->num_rows()==0){
        show_404();
    }
    $row = $query->row();
    setPreguntaCache($id,$row);
}
```

Figura 11. Fragmento de código que optimiza las consultas a la base de datos basado en caché.

Esto permitió reducir las consultas a la base de datos considerablemente e incluso fue tomado como práctica estándar para las consultas de mayor frecuencia a la base de datos.

3.3.4 Desarrollo del uso de réplicas de lectura de base de datos

Luego de haber sido creada la réplica de lectura de la base de datos, se utiliza la capacidad de conectarse a múltiples bases de datos que tiene CodeIgniter para, en la capa de aplicación, consultar la réplica de lectura cuando no sea necesario llevar a cabo alguna escritura.

Esta funcionalidad fue implementada progresivamente, comenzando por los queries de lectura más frecuentes para generar un mayor impacto. Gracias a que CodeIgniter también abstrae todos los queries a la base de datos, el cambio requerido en código es mínimo como se muestra en la *figura 12.*

```
// conexión a la réplica de lectura
$db_replica = $this->load->database('read_replica',true);

// query a base de datos principal
$result = $this->db->where('id',$id)->get('preguntas',1,0);

// query a la réplica de lectura
$result = $db_replica->where('id',$id)->get('preguntas',1,0);
```

Figura 12. Fragmento de código comparación conexión a base de datos principal y réplica de lectura.

Esto permitió distribuir la carga de la base de datos entre al menos dos instancias de Amazon RDS y disminuir la probabilidad de un retardo en el tiempo de respuesta en la base de datos ante un posible crecimiento acelerado.

4. OTRAS FUNCIONES

Durante la práctica se llevaron a cabo otras funciones que toda empresa desarrolla de software requiere para mantener su funcionamiento y permitir a todos sus desarrolladores continuar con el trabajo realizado por sus colegas independientemente de que se encuentren aún o no en la organización.

4.1 DOCUMENTACIÓN Y SOCIALIZACIÓN DE LAS SOLUCIONES IMPLEMENTADAS

La documentación del software desarrollado, especialmente para una organización en la que el trabajo en equipo prevalece, es la base para la continuidad de los proyectos desarrollados tanto dentro del equipo como para la satisfacción de los clientes a los que se les permite continuar los proyectos desarrollados por Cloud Based por su propia cuenta, sobre las buenas bases que se les construye.

Bajo este principio se lleva a cabo toda la documentación de los proyectos desarrollados, siguiendo un estándar claro para facilitar el mantenimiento del software tanto dentro de la empresa como afuera de ella.

Normalmente hay dos tipos de documentaciones que se elaboran a la par del desarrollo de un proyecto: la documentación del código fuente, llevado a cabo dentro del mismo código y pretende explicar el mismo, y la documentación del proyecto en general, que busca explicar su funcionamiento a grandes rasgos, protocolos de comunicación internos y con terceros, el enfoque utilizado para solucionar algunos problemas, o simplemente comentar acciones que dentro del contexto del proyecto deban ser explicadas.

4.1.1 Documentación del código fuente

Esta documentación se lleva a cabo basada en el estilo DocBlock que su objetivo es estandarizar la forma de documentar las estructuras de código más comunes, entre ellas:

- Funciones
- Constantes
- Clases
- Constantes de clase
- Interfaces
- Propiedades
- Métodos

Cada uno de los elementos anteriores, tiene asociado un bloque de comentarios que define y busca explicar el funcionamiento del mismo bajo ciertos parámetros. A cada uno de estos bloques les llamamos *DocBlocks*. Y serán asociados a una y solamente una de estas estructuras y estas serán directamente precedidas por estos DocBlocks.

Los DocBlocks están siempre encerrados en un tipo de comentario al que llamaremos *DocComment* que inicia con /** y termina con */. Cada línea o renglón dentro de estas dos declaraciones debería comenzar con un asterisco (*). Cada DocBlock precede exactamente una estructura de las anteriores mencionadas y todo el contenido de dicho DocBlock aplica para esa estructura asociada.

Por ejemplo:

```php
<?php

/**
* Esto es un DocBlock
*/
function funcionAsociada(){
}
```

Figura 13. Ejemplo de un DocBlock

Los DocBlocks están divididos en las siguientes tres partes. Cada una de estas partes es opcional, excepto que una descripción no debería existir sin un resumen.

Resumen

Provee una breve descripción de la función del elemento asociado, en ocasione llamado una descripción pequeña. Una descripción termina dentro de un DocBlock con una de las siguientes situaciones:

1. Un punto (.) seguido de un salto de línea, o
2. Dos saltos de líneas consecutivos.

Descripción

Provee más información sobre la función del bloque asociado. Ejemplos de mayor información son la descripción del algoritmo de una función, un ejemplo de uso o la descripción de cómo una clase encaja dentro de la arquitectura de la aplicación. La descripción termina en la primer etiqueta o cuando el DocBlock es cerrado.

Etiquetas y anotaciones

Estos permiten de una forma de uniforme y breve de presentar meta-información sobre el elemento asociado. Estos, por ejemplo, describen el tipo de información que es retornada por un método o una función. Cada etiqueta esta precedida por un signo arroba (@) y comienza en una nueva línea.

Un ejemplo de un DocBlock con todos los elementos es el siguiente:

```php
<?php
/**
 * Un resumen informando al usuario qué hace el elemento.
 *
 * Una descripción, que puede contener múltiples líneas para
 * entrar más en detalles de algún elemento y proveer algún contexto
 * o referencias que ayuden a comprender mejor.
 *
 * @param string $argumento Con una descripción del argumento,
 * también pueden contener múltiples líneas.
 *
 * @return void
 */
 function funcionEjemplo($argumento)
 {
 }
```

Figura 14. Ejemplo de un DockBlock con todos sus elementos.

Para facilitar este proceso y automatizarlo lo más posible, se utilizó una herramienta llamada phpDocumentor, que permite generar los DocBlocks automáticamente y ayuda al desarrollador a hacer esta tarea más fácil, generando la estructura automáticamente y dejando únicamente la tarea de adaptar la plantilla generada a la estructura por documentar.

4.1.2 Documentación general

Esta documentación que busca dar a entender el funcionamiento a grandes rasgos de la aplicación, sus protocolos, alcance y funcionalidades, no fue llevada a cabo bajo ningún estándar sino por un formato previamente utilizado por la empresa.

Para el caso del proyecto de 1DOC3 se especificaron los siguientes aspectos del proyecto en la documentación general:

Diagramas de infraestructura

El mayor cambio a la plataforma de 1DOC3 fue enfocado a su infraestructura, por lo que para futuros desarrolladores o arquitectos, se especificaron detalles como la configuración de los servidores, el diagrama de la arquitectura como se mostró en este documento, la configuración de redes internas dentro de AWS, como interactúa el cliente con el servidor, etc.

Se optó por obviar los diagramas de arquitectura debido a que ya fueron ampliamente presentados en este documento.

Comunicación cliente-servidor

Un escenario común en una aplicación web como 1DOC3 es la comunicación entre el cliente y el servidor para ejecutar acciones sin necesidad de recargar la página o simplemente para mostrar al usuario información del estado de algún componente en tiempo real.

Para esto se utilizaba tecnología AJAX ampliamente conocida para la comunicación asíncrona entre el cliente y el servidor. Para estandarizar esta comunicación se utilizó una notación también ampliamente conocida que se ha vuelto prácticamente un estándar en la comunicación web llamado JavaScript Object Notation (JSON).

Basados en estas herramientas y principios, se crearon ciertas constantes que buscaban facilitar a los desarrolladores futuros del proyecto la fácil comprensión de esta comunicación, por lo que, para el caso de la comunicación cliente-servidor, se determinó que la comunicación se iba a llevar a cabo de la siguiente manera y así se debía regir en los futuros desarrollos.

Toda comunicación del cliente hacia el servidor (que no sea la solicitud de una página completa) se debe hacer como una solicitud HTTP tipo POST con las variables necesarias. El servicio que responde esta solicitud responderá siempre con al menos una propiedad en un formato JSON:

1. **Status (obligatoria):** con dos posibles valores:
    a. **OK:** Para que el cliente valide la correcta ejecución de la solicitud y acceda a las variables adicionales de respuesta en caso de ser necesitadas.
    b. **BAD:** Para que el cliente valide que la ejecución de la solicitud no fue llevada a cabo correctamente y, según el caso, se muestre un mensaje de error o se solicite información faltante al usuario. En este caso es obligatorio responder con una segunda variable que indique el tipo de error y el mensaje a mostrar al usuario en caso de ser necesario.
2. **Msj (en caso de error):** El valor de esta propiedad será el mensaje que deba ser mostrado al usuario en caso de ser necesario.
3. **StatusCode (en caso de error):** Indica el tipo de error en la solicitud para tomar decisiones del lado del cliente, sus valor pueden ser, por ejemplo:
    a. **1:** El usuario no está autorizado para llevar a cabo esta tarea.
    b. **2:** Error en la validación de algún parámetro enviado.
    c. **3:** Falta algún dato obligatorio para llevar a cabo esta solicitud.

4.2 MONITOREO DE LA CARGA, EL ALMACENAMIENTO Y EL USO DE RED DE CADA COMPONENTE DEL SISTEMA

Otra de las funciones que se llevaban a cabo no sólo a la hora de analizar los requerimientos para tomar decisiones, sino durante todo el desarrollo del proyecto, fue el monitoreo permanente del funcionamiento y de los cambios desplegados en la plataforma.

El monitoreo se realizaba tanto a los servidores de aplicación como a las instancias de base de datos que eran los puntos clave para determinar el rendimiento de la aplicación y su disponibilidad.

Se monitoreaban principalmente dos fuentes de información:

4.2.1 Consola de administración de AWS

La consola de administración de AWS permitía conocer el estado de los servidores casi en tiempo real, lo que no sólo servía para tomar decisiones sino para analizar el comportamiento de los desarrollos desplegados en ambiente de producción y ver la reacción ante pruebas de carga o estrés al sistema.

Se hacía seguimiento principalmente a dos indicadores:

Uso de CPU

Esta métrica nos permitía determinar qué tan exigidos estaban los servidores de aplicación y nos dimos cuenta que se relacionaba directamente con el tiempo de respuesta de la aplicación, por lo que fue tomada como punto de referencia para determinar si el grupo de auto-escalamiento se debía aumentar.

A diferencia de las instancias de EC2, que en caso de ser necesario se podía reaccionar en tiempo real escalando el número de instancias, la base de datos no tenía esta capacidad de reacción y su monitoreo se hacía con el fin de tomar decisiones a largo plazo, encontrar queries ineficientes en la aplicación y hacer seguimiento a la ejecución de inserciones por lotes o procesamiento de información en horas de tráfico disminuido.

En la *figura 15* se evidencia cómo influía en la base de datos el procesamiento que se hacía todos los días a las cinco de la mañana todos los días para generar estadísticas y reportes del día anterior.

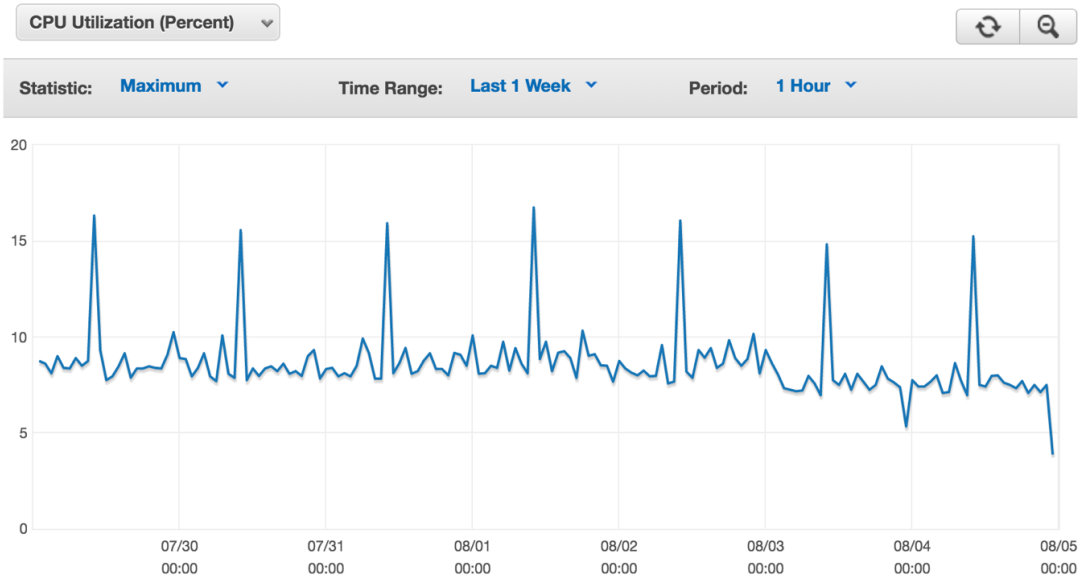

Figura 15. Ejemplo de impacto de la generación de estadísticas en el uso de CPU de la base de datos.

Esta métrica en la base de datos nos permitía tomar, a largo plazo, la decisión de aumentar la capacidad de la misma y detectar queries ineficientes.

Uso de red

Esta métrica nos permitía complementar la información del uso de CPU, ya que esta depende directamente de la cantidad de peticiones que responde un servidor. A diferencia del uso de CPU, el uso de red nos permitía saber

cuantas peticiones fue capaz de responder un servidor en determinado tiempo, y así en conjunto del indicador de uso de CPU determinar qué tan eficiente estaba siendo la ejecución de la aplicación y medir las optimizaciones llevadas a cabo.

Así, por ejemplo, si conocemos que una solicitud a una URL en promedio requiere 40KB, y el servidor tuvo un uso de red de salida de 5MB por minuto esto quiere decir que fue capaz de responder 125 solicitudes por minuto. Ahora bien si el uso de CPU, por ejemplo, era de 10%, podríamos suponer que este servidor estaba en capacidad de responder 1250 solicitudes por minuto.

De lo anterior podríamos tomar decisiones de dos forma:

- Mejorar el uso de CPU por petición optimizando la misma.
- Aumentar el tamaño de los servidores para que puedan responder más solicitudes cada uno.

4.2.2 Logs de los servidores de aplicación

Aparte del monitoreo de la carga de los servidores, se supervisaban los posibles errores generados a nivel de aplicación, es decir, errores de lógica o de sintaxis que hayan sido desplegados a pesar de las pruebas de protocolo.

Si bien los servidores de producción fueron configurados para nunca mostrar errores de cara a los usuarios, si que se llevaba un registro de estos errores en caso de que ocurrieran, por lo que, cada vez que se desplegaba un cambio en la plataforma, se hacía el seguimiento a estos archivos los días siguientes a su publicación.

5. RESULTADOS

El trabajo hecho durante la práctica empresarial para este cliente fue muy bien recibido debido a que les permitió dar un gran paso en términos tecnológicos y estar preparados para el posible crecimiento exponencial de su base de usuarios.

El desarrollo en 1DOC3 bajo los principios de escalabilidad y replicabilidad de la aplicación propuestos e implementados durante la práctica les permitió a los actuales y futuros desarrolladores del sistema mantener una alta disponibilidad en la plataforma y construir sobre unas excelentes bases y principios un sistema capaz de servir a millones de personas sin necesidad de intervención humana a la hora de escalar su capacidad.

Las principales características añadidas a la plataforma fueron las siguientes:

5.1 EL ALMACENAMIENTO DE ARCHIVOS DE MANERA ESCALABLE

La plataforma existente de 1DOC3 almacenaba los archivos en los servidores de aplicación, por lo que creaba dependencia de estos mismos y evitaba a la aplicación replicarse a través de varios servidores.

Hoy en día 1DOC3 almacena los archivos en un servicio de almacenamiento externo escalable que permitió eliminar todas esas dependencias locales y lograr convertirse en una plataforma plenamente escalable. Esto significó también un cambio en la mentalidad de los desarrolladores de la plataforma pues de aquí en adelante se planeaban las nuevas funcionalidades siempre con la replicabilidad y escalabilidad en mente.

5.2 SESIONES ESCALABLES

La plataforma inicial si bien tenía un uso avanzado de sesiones, el almacenamiento de las mismas creaba también dependencias locales que, basados en las necesidades actuales de escalamiento, no debían continuar.

Esto llevo a la creación de un clúster de caché de alto rendimiento para almacenar allí los datos de sesiones de manera escalable y eliminando las dependencias mencionadas. Este clúster puso a disposición de la plataforma y los desarrolladores otra forma de almacenamiento que antes no había sido tenida en cuenta y que por su rendimiento y su tipo de almacenamiento en memoria, permitió satisfacer otras necesidades de la plataforma más adelante durante la práctica empresarial.

5.3 CACHÉ DE CONSULTAS A LA BASE DE DATOS

Al comienzo de la práctica empresarial, la plataforma 1DOC3 contaba con una base de datos con una carga casi inmanejable.

Gracias a que ya se creó un clúster de caché para almacenamiento escalable en memoria, se logró desarrollar también un sistema de caché de consultas a la base de datos haciendo uso de este clúster. Esto también significó un cambio de paradigma en la aplicación, pues los resultados en el rendimiento de la base de datos se vieron reflejados inmediatamente, disminuyendo considerablemente las consultas a la base de datos que no era necesarias llevarlas a cabo.

5.4 RÉPLICA DE LECTURA A LA BASE DE DATOS

En la plataforma inicial analizada, sólo se contaba con un servidor de base de datos el cual recibía toda la carga, lo que lo convertía en un inminente cuello de botella a la hora de aumentar los usuarios en la aplicación.

Con la creación de la réplica de lectura, la adaptación del software para consultar esta cuando no sea necesario llevar a cabo escrituras y junto con el uso de caché para las consultas, permitió aumentar ampliamente el rendimiento de la base de datos y preparar la aplicación para el crecimiento acelerado de usuarios y carga.

5.5 BALANCEAMIENTO DE CARGA Y ESCALAMIENTO AUTOMÁTICO

A diferencia de la arquitectura inicial de la plataforma, que recibía el tráfico directamente en el único servidor de aplicación, la arquitectura desarrollada durante la práctica empresarial le dio la capacidad a la plataforma de escalar automáticamente, es decir, sin intervención humana, adaptándose a la carga a medida que crecía.

5.6 OTROS RESULTADOS

Disponibilidad

La nueva arquitectura permite a 1DOC3 ser una aplicación con una alta disponibilidad, lo que no sólo se traduce en un mejor servicio para sus usuarios sino que mejora la oferta de valor para con sus mismos clientes, llegando a cumplir una promesa de una disponibilidad del 99.95% del tiempo al año.

Seguridad

El rediseño de la infraestructura no sólo supone una mejora en la disponibilidad de la plataforma, también logra una mejora sustancial en la seguridad aprovechando las bondades de AWS.

Por medio de los grupos de seguridad y su configuración, se lograron aislar de Internet tanto las instancias de EC2, a las cuales sólo se les permitió el

ingreso de solicitudes desde el balanceador de carga, como a la base de datos (y su réplica), a las cuales sólo se les permitió el acceso desde las instancias.

5.7 TABLA DE CUMPLIMIENTO DE OBJETIVOS

En la *tabla 6* se muestra como fueron cumplidos los objetivos de la práctica empresarial y en qué sección del documento se describe su desarrollo.

Objetivo	**Porcentaje de cumplimiento**	**Sección del documento**
Analizar y diseñar las funcionalidades requeridas actualmente, como son: - Perfil destacado para los doctores con membresía en la plataforma. - Implementar el protocolo SSL en la plataforma.	100%	En la sección **2.4** se explica por qué no se llevó a cabo el desarrollo del perfil destacado para doctores. En la sección **3.1.1** se describe la implementación del protocolo SSL y su resultado.
Analizar los nuevos requerimientos de la plataforma 1doc3, basados en las necesidades actuales de los usuarios.	100%	En las secciones **2.2** y **2.4** se analizan los requerimientos de software y hardware respectivamente a fin de hacer la plataforma escalable. Se obtuvo como resultado los siguientes requerimientos: **- Dejar de almacenar archivos en servidor local** **- Cambiar el uso de sesiones basadas en archivos locale** **- Utilizar caché para las consultas a la base de datos** **- Uso de réplica de lectura de la base de datos** **- Implementar un balanceador de carga y un grupo de autoescalamiento.** Para cada uno de estos requerimientos se llevaban a cabo adaptaciones tanto de hardware como de software.
Diseñar las nuevas funcionalidades que surjan de los anteriores requerimientos, de forma que se adapten al estado actual y sean escalables a futuras ampliaciones.	100%	En la sección **3.1** se muestra la propuesta del nuevo diseño de hardware que para ser ejecutado también requiere adaptaciones de software.
Implementar las soluciones diseñadas en la plataforma 1doc3	100%	En las secciones **3.2** y **3.3** se lleva a cabo el desarrollo de la solución tanto de hardware como de software respectivamente.
Documentar y socializar cada una de las soluciones implementadas en la plataforma 1doc3.	100%	En la sección **4.1** se describe la metodología e implementación de la documentación y socialización del proyecto.
Monitorear la carga, el almacenamiento y el uso de red de cada componente del sistema.	100%	En la sección **4.2** se describe como se llevó a cabo el monitoreo de la plataforma durante toda la práctica.

Tabla 6. Tabla de cumplimiento de objetivos

6. CONCLUSIONES Y RECOMENDACIONES

6.1 CONCLUSIONES

La práctica empresarial es una gran oportunidad para los estudiantes de aplicar los conocimientos construidos durante toda su carrera en un ambiente real, interactuando con los clientes y sometiendo a prueba su capacidad de planeación y ejecución.

En particular al autor de este proyecto, le permitió capacitarse y tener experiencia real en:

- La metodología de trabajo de una empresa de desarrollo de software con clientes de alto impacto.
- Uso y aplicación de tecnologías de punta para ambientes web escalables.
- Conocer herramientas para mejorar el trabajo en equipo en el desarrollo de software y llevar un mejor control de las versiones del software.
- Interacciones con clientes de la industria y aprendizaje de la experiencia de los mismos.

La práctica empresarial me ha permitido conocer a profundidad las ventajas de Amazon Web Services, que así como permite a desarrolladores de todo el mundo desplegar arquitecturas web con la más alta tecnología, me facilitó llevar a cabo una arquitectura tan robusta como la implementada en la plataforma 1DOC3

El hecho de estar en permanente interacción y contacto con un cliente de alto impacto a nivel nacional como es 1DOC3, me permitió desarrollar no sólo habilidades técnicas, sino aprender también a desenvolverme en otros tipos de ambientes profesionales como presentaciones al cliente y reuniones

comerciales que son también habilidades muy importantes para el desarrollo profesional.

6.2 RECOMENDACIONES

Se recomienda a 1DOC3 y su equipo de trabajo no perder nunca la visión que se le dio al desarrollo de la plataforma pensando siempre en la escalabililidad, ya que esto permitió reflejar la visión de negocio de 1DOC3 como empresa en su plataforma y es, personalmente, el más importante resultado de toda la práctica empresarial.

Se recomienda a la Escuela de Ingeniería de Sistemas e Informática tener en cuenta las herramientas y prácticas utilizadas dentro de esta práctica tanto para el pensum como para el área de investigación, pues son sin duda servicios como AWS los que están cambiando la industria de software a nivel mundial y como profesionales debemos estar al tanto de esto.

BIBLIOGRAFÍA

AMAZON WEB SERVICES, Documentación. [En línea]. [consultado: 22 de Agosto, 2016]. Disponible en: (https://aws.amazon.com/es/documentation/)

CODEIGNITER, Guía de usuario. [En línea]. [consultado: 22 de Agosto, 2016]. Disponible en: (http://www.codeigniter.com/user_guide/)

PATRÓN DE ARQUITECTURA MODELO VISTA CONTROLADOR. [En línea]. [consultado: 22 de Agosto, 2016] Disponible en: (http://www.lab.inf.uc3m.es/~a0080802/RAI/mvc.html)

PHPDOCUMENTOR, Guía de uso. [En línea]. [consultado: 22 de Agosto, 2016]. Disponible en: (https://phpdoc.org/docs/latest/guides/docblocks.html)

ANEXOS

Carta de aprobación de Cloud Based SAS

CloudBased S.A.S

Bucaramanga, 5 de Agosto del 2016

Señores:
ESCUELA DE INGENIERIA DE SISTEMAS E INFORMÁTICA
Universidad Industrial de Santander
Bucaramanga

La presente tiene como fin certificar que se recibió a satisfacción el informe del proyecto de grado en modalidad de práctica empresarial DISEÑO E IMPLEMENTACIÓN DE UNA ARQUITECTURA WEB ESCALABLE PARA 1DOC3 del estudiante NICOLÁS DURÁN LÓPEZ, identificado con cédula de ciudadanía 1.098'722.472 de la ciudad de Bucaramanga y que se verificó el cumplimiento de los objetivos.

Israel Alfonso Pedraza
C.C. 13.748.461
Tutor

Transversal 19 a No. 98 - 28 Oficina 701 / +(57) 1 622 56 04 / comunicaciones@guarumo.com
www.guarumo.com