

## ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

## Sistemas Distribuidos

## II-2019

## Proyecto: Arquitectura basada en servicios para aplicación Web

# **Diseño del Sistema**

# Repositorio en GITHUB:

<https://github.com/steffanyvallejo/Proyecto_SistDistribuidos>

# Servicios y Comunicación:

* AWS Fargate es un motor informático sin servidores que permite la asignación de cantidad correcta de cómputo, es decir, se paga solo los recursos que se necesitan para ejecutar los contenedores.
* Amazon Elastic Container (ECS) permite la administración y ejecución de contenedores de Docker, altamente escalables utilizando AWS Fargate con quien se lanza las tareas.
* Elastic Load Balancing permite distribuir de manera automática el trafico entre las tareas de Amazon ECS, brindando tolerancia a fallos.

# Middleware:

# Lenguaje de Programación, librerías y herramientas:

* El lenguaje de programación que se utilizó para desarrollar los servicios del *backend* es: JAVA, se tomó esta decisión por la cantidad de documentación e información disponible relacionada al tema, además de la experiencia previa con el lenguaje con lo que se mejoró el tiempo de desarrollo.
* Las librerías que se utilizaron:
  + POI: Es una librería que proporciona bibliotecas Java puras para leer y escribir archivos en formatos de Microsoft Office , como Word , PowerPoint y Excel, con esta librería se espera complementar la funcionalidad de convertir un documento .docx a PDF.
  + ImageJ: Al igual que la librería anterior, ImageJ permite manipular, procesar o convertir imágenes de diferentes formatos. Usando esta herramienta se espera brindar al usuario la posibilidad de convertir imágenes de un formato a otro.
  + JAVE2 (Java Audio Video Encoder): Es un contenedor Java en el proyecto ffmpeg. Permite transcodificar archivos de audio y video de un formato a otro. En este caso, se utiliza esta librería para convertir un archivo de audio WAV a MP3. JAVE2 admite muchos otros formatos, contenedores y operaciones.

# Funcionamiento del Sistema

El sistema funciona de tal manera que existen 3 servicios de conversión de archivos en diferentes formatos (documentos, imágenes y audio), estos servicios pertenecen al a un clúster llamado Cluster-Convert. Cada servicio posee una instancia de una tarea donde se aloja la imagen de Docker correspondiente al JAR del microservicio. Adicionalmente, se crean target Group para cada servicio, con el fin de asociarlo con un Load Balancer previamente creado. El Load Balancer permite distribuir la carga a través de un link DNS y utiliza el algoritmo Round Robin.

Ahora, los clientes para realizar deben entrar al link \*\*\*, cada botón de conversión está asociado al link de conversión con la ruta DNS más la función correspondiente.

# Pruebas

# Presupuesto

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |

# Diagrama Diseño de Arquitectura:

# Referencias:

[1] E. Salguero, "Sistemas distribuidos: paradigmas de comunicación (V)", *Medium*, 2019. [Online]. Available: [https://medium.com/@edusalguero/sistemas-distribuidos-paradigmas-de-comunicacion-fc450ff3860b](https://grpc.io/). [Accessed: 09- Dec- 2019].

[2]"gRPC", *Grpc.io*, 2019. [Online]. Available: <https://grpc.io/>. [Accessed: 09- Dec- 2019].

[3]"Protocol Buffers  |  Google Developers", *Google Developers*, 2019. [Online]. Available: [https://developers.google.com/protocol-buffers](https://grpc.io/). [Accessed: 09- Dec- 2019].

[4]"File Format APIs for Word Excel PDF Email PowerPoint Barcode Images OCR Note &amp; 3D", *Aspose.com*, 2019. [Online]. Available: [https://www.aspose.com/](https://grpc.io/). [Accessed: 09- Dec- 2019].