



UNIVERSIDAD AUTÓNOMA DE LA CIUDAD DE MÉXICO  
COLEGIO DE CIENCIA Y TECNOLOGÍA  
PLANTEL CUAUTEPEC

## PROYECTO FINAL

Proyecto que presenta Balamm Salvador Lucio Lopez,  
Luis Alberto Varela Varela,  
Anahi Carrillo Alvarado,  
Raúl Hernández Chaires

a la profesora Elizabeth López Lozada como práctica de certificación  
de la materia de **Sistemas Distribuidos** en el ciclo 2024-II

México, CDMX, a 22 de mayo de 2024



# Índice

<b>Introducción.....</b>	<b>3</b>
<b>Objetivos generales del sistema.....</b>	<b>4</b>
Objetivos específicos:.....	4
Objetivos Diseño.....	4
<b>Capítulo 1: Marco Teórico.....</b>	<b>5</b>
a) Panorama.....	5
b) Ventajas y desventajas de los sistemas distribuidos (SD).....	6
Arquitectura de un sistema distribuido.....	7
Topología y/o Arquitectura tipo Cliente/Servidor.....	7
Topología y/o Arquitectura tipo Consumidor/Productor.....	7
Protocolo TCP/IP.....	8
c) Arquitectura de Netflix.....	9
d) Tipos de Streaming.....	11
DISTRIBUCIÓN DE AUDIO.....	11
AUDIO ANALÓGICO.....	11
AUDIO DIGITAL.....	11
Elementos en un sistema de audio distribuido.....	13
DISTRIBUCIÓN DE VIDEO.....	14
COMPONENTES.....	14
<b>Capítulo 2: Desarrollo.....</b>	<b>15</b>
HERRAMIENTAS E INSTALACIÓN.....	16
<b>ARQUITECTURA PROPUESTA.....</b>	<b>17</b>
Diagrama de Despliegue.....	17
Componentes.....	18
Casos de uso.....	19
Objetos.....	20
<b>IMPLEMENTACIÓN DEL CÓDIGO.....</b>	<b>21</b>
Clase Reproducción (Cliente).....	22
Clase Servidor.....	23
Clase Cliente.....	26
Demostración de funcionamiento.....	27
<b>CAPÍTULO 3: OBSERVACIONES Y CONCLUSIONES.....</b>	<b>28</b>



# Introducción

La tecnología ha evolucionado constantemente empezando desde la creación de la primera computadora, si hablamos del ábaco, ese que usamos en el kinder y en algunas primarias hasta la primera computadora fabricada en aquellos años hasta llegar a la actualidad donde pasamos de la evolución al hardware y adaptar ese software al hardware, y para eso llegaron los sistemas operativos desde Unix, a Linux como Windows a IOS, marcando un hito en el desarrollo de la computación, hablemos de la 1ra generación de los Sistemas Operativos la cual fue en la década de los 50, con las computadoras costosas y enormes, no existía software que controlara y gestionara las funciones, poco después apareciendo **GM-NAA I/O** el cual fue desarrollado por IBM para las computadoras mainframe las cuales interactuaban con terminal y comandos algo como la matrix facilitando la gestión de recursos del sistema, luego apareció la 2da generación la cual empezó a tener sistemas operativos con transistores que simplificaban y automatizaban las tareas, teniendo mejor control y gestión de los mismos recursos de las computadoras, llegando a la 3ra generación la cual consistía en circuitos integrados el avance tecnológico más grande en los sistemas operativos ya que ofrecían mejor y mayor rendimiento y llegando así a la 4ta generación la cual consiste en sistemas operativos con multiprocesadores siendo aquí el nacimiento de UNIX.

Conforme fue evolucionando los sistemas operativos ha habido una evolución de los sistemas distribuidos la cual es fascinante lo que ha logrado transformar en la tecnología y si forma de gestionar las tareas permite dividir procesos para un trabajo más eficiente; en 1960 a 1970 las redes de computadoras surgieron y con ella el proyecto **ARPANET** el padre de la hoy conocida internet; utilizada para compartir recursos y comunicarse entre diferentes instituciones académicas y de investigación, lo cual popularizó el concepto de **comunicación de paquetes** para una transmisión eficiente hasta que en 1983 apareció el protocolo TCP/IP y nació el internet.

En la llegada de la cloud computing empezaron a aparecer plataformas como Amazon Web Services (AWS), Microsoft Azure y Google Cloud que ofrecen servicios como almacenamiento, procesamiento, análisis de datos, entre otros, alojando todos los datos en servidores contenidos alrededor del mundo donde puedes tener la información accesible a través de algún servicio web.

Los sistemas distribuidos son un tipo de sistema que se basa en una colección independiente de dispositivos teniendo características únicas como la compatibilidad, tolerancia a fallas, y tiene una capa de software que proporciona la abstracción y servicios comunes para facilitar la comunicación y la coordinación con el middleware y las API's.



# Objetivos generales del sistema

- **Ampliar la audiencia:** Aumentar el número de usuarios que utilizan la plataforma
- **Mejorar la satisfacción del usuario:** Proporciona una experiencia visual fluida y agradable
- **Incrementar la retención de usuarios:** Fomentar que los usuarios regresen y utilicen la plataforma de manera recurrente

## Objetivos específicos:

- **Optimizar la velocidad de carga:** Reducir el tiempo de espera en los usuarios para ver el contenido
- **Personalizar recomendaciones:** Utilizar ciertos algoritmos que favorezcan mejor contenido relevante
- **Implementar funciones sociales:** Permitir a los usuarios compartir su lista de reproducción y compartir sus opiniones
- **Garantizar la seguridad de los datos:** Proteger la privacidad de los usuarios y sus datos personales

## Objetivos Diseño

Criterio de diseño	Definición
<b>Latencia (Tiempo de latencia)</b>	<ul style="list-style-type: none"><li>• Las peticiones que realizan los usuarios deben ser respondidas en menos de 5 segundos.</li><li>• En este caso el Productor genera un Evento para que el Consumidor pueda emitir una Respuesta y encontrarla rápidamente en la lista de Eventos</li></ul>
<b>Concurrencia</b>	<ul style="list-style-type: none"><li>• Manejar múltiples solicitudes simultáneamente sin que cause conflicto o inconsistencias con otras solicitudes</li></ul>
<b>Robustez</b>	<ul style="list-style-type: none"><li>• Evalúa qué tan bien responde el sistema a condiciones de operación inesperadas.</li><li>• Un sistema robusto es capaz de manejar situaciones inesperadas o errores sin colapsar.</li></ul>



<b>Seguridad</b>	<ul style="list-style-type: none"> <li>• Evalúa qué tan bien el sistema protege contra lesiones o daños físicos,</li> <li>• también que tan fácil se recupera de fallas como lo son el hardware, software y la comunicación (broker)</li> <li>• Incluye medidas como el control de acceso, la protección contra incendios y la seguridad en las instalaciones</li> </ul>
<b>Costo de desarrollo</b>	<ul style="list-style-type: none"> <li>• Para el procesamiento, y desarrollo en kubernetes se estima un Total: ≈ \$10,600/mes</li> </ul>
<b>Escalabilidad (Extensibilidad)</b>	<ul style="list-style-type: none"> <li>• Asegurar que el sistema pueda manejar un aumento en la carga de trabajo sin degradar el rendimiento.</li> <li>• Distribuir la carga de manera equitativa entre los nodos para evitar cuellos de botella.</li> </ul>
<b>Redundancia</b>	<ul style="list-style-type: none"> <li>• Almacenar copias redundantes de datos o servicios en diferentes ubicaciones.</li> <li>• Si un nodo falla, otro puede tomar su lugar sin interrupciones.</li> </ul>

## Capítulo 1: Marco Teórico

### a) Panorama

#### ¿Qué son los sistemas distribuidos?

Para Tanenbaum & Maarten Van, los sistemas distribuidos son una colección de computadoras independientes que dan al usuario la impresión de constituir un único sistema coherente ( 2008).

De acuerdo a Martin, las computadoras que están conectadas mediante una red pueden estar separadas espacialmente por cualquier distancia(2004).

Por lo que vemos en estas dos definiciones por los autores se puede decir que un sistema distribuido es un conjunto de programas informáticos que utilizan recursos computacionales en varios nodos de cálculo distintos para lograr un objetivo compartido común. La finalidad de los sistemas distribuidos es eliminar los cuellos de botella o los puntos de error centrales de un sistema.

## b) Ventajas y desventajas de los sistemas distribuidos (SD)

Ventajas y desventajas de los sistemas distribuidos<sup>1</sup>

Ventajas con respecto a los sistemas centralizados	Ventajas con respecto a las computadoras aisladas	Desventajas
<b>Economía:</b> Microprocesadores ofrecen una mejor relación precio/rendimiento. <b>Velocidad:</b> Puede tener mayor poder de cómputo	<b>Datos compartidos:</b> Permite que los usuarios tengan acceso a una BD o archivo común. <b>Dispositivos compartidos:</b> Permite compartir un recurso costoso entre distintos usuarios, como plotters o impresoras láser.	<b>Software:</b> Gran parte del software para SD está aún en desarrollo. <b>Redes:</b> Los problemas de transmisión en las redes de comunicación todavía son frecuentes en la transferencia de grandes volúmenes de datos (por ejemplo, multimedia).
<b>Distribución inherente:</b> Implica que puede emplear aplicaciones instaladas en computadoras remotas. <b>Confiabilidad:</b> Consistente, aun si una computadora del sistema deja de funcionar.	<b>Comunicación:</b> Brinda la posibilidad de comunicación de usuario a usuario (telnet, correo electrónico, etc.). <b>Confiabilidad:</b> Facilita la repartición de la carga de trabajo entre las distintas computadoras con base en su funciones y capacidades, brindando una mayor flexibilidad y confiabilidad al sistema.	<b>Seguridad:</b> Se necesitan mejores esquemas de protección para mejorar el acceso a información confidencial o secreta. <b>Tolerancia a fallas:</b> Las fallas operativas y de componentes aún son frecuentes.
<b>Escalado Horizontal:</b> Cada que se requiera mayor poder de cómputo; solo se pueden adicionar los incrementos de cómputo requeridos.	<b>Escalado Vertical:</b> Cada que se requiere mejorar el rendimiento solo se puede agregar recursos temporalmente de almacenamiento como RAM, almacenamiento o CPU	

Tabla 1: Comparativa de las ventajas y desventajas de los sistemas distribuidos centralizados y aislados

---

<sup>1</sup> tabla comparativa de las ventajas y desventajas de los sistemas distribuidos centralizados y aislados

## Arquitectura de un sistema distribuido

### Topología y/o Arquitectura tipo Cliente/Servidor

El modelo cliente-servidor, también conocido como “principio cliente-servidor”, es un modelo de comunicación que permite la distribución de tareas dentro de una red de ordenadores.

Un servidor es un hardware que proporciona los recursos necesarios para otros ordenadores o programas, pero un servidor también puede ser un programa informático que se comunica con los clientes. Un servidor acepta las peticiones del cliente, las procesa y proporciona la respuesta solicitada. También existen diferentes tipos de clientes. Un ordenador o un programa informático se comunica con el servidor, envía solicitudes y recibe respuestas del servidor. En cuanto al modelo cliente-servidor, representa la interacción entre el servidor y el cliente.

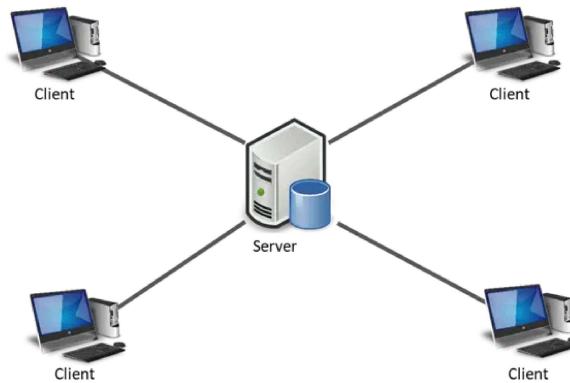


Figura 1:Arquitectura Cliente/Servidor<sup>2</sup>

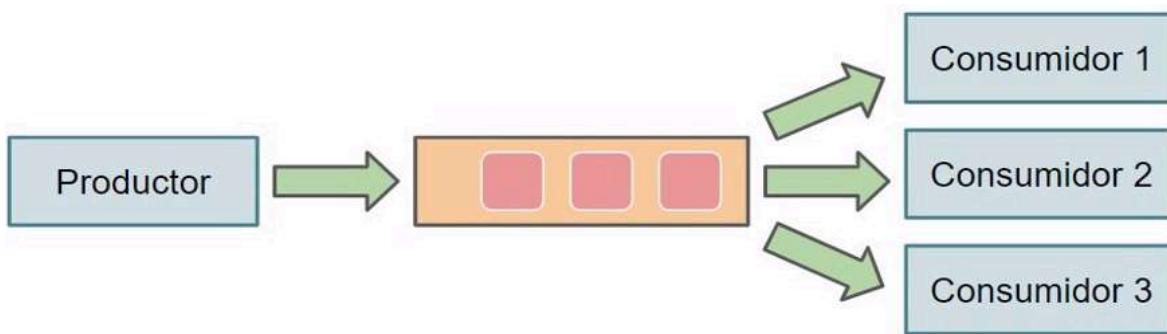
### Topología y/o Arquitectura tipo Consumidor/Productor

**Productor Consumidor** es un modelo de comunicación entre hilos donde un hilo produce valores y otro hilo donde consume esos valores. Todo esto de manera sincronizada.

Ambos hilos tratan de manejar valores de una zona común que se le suele llamar Monitor (también se le suele llamar Buffer) que se encarga de sincronizar la entrada y salida de valores.

---

<sup>2</sup> Diagrama que muestra



*Figura 2:Arquitectura Consumidor/Producer<sup>3</sup>*

## Protocolo TCP/IP

Es un conjunto de protocolos de red que se utiliza para permitir la comunicación entre dispositivos en una red, y vamos a ver como se relaciona con el concepto de “mensajes por trazas”. Este modelo se divide en **cuatro capas**, y estas son:

### 1. Capa de Aplicación:

- Es la más cercana al usuario final y permite la interacción con la aplicación.
- Es donde ocurre la interpretación de los datos de los programas de aplicación, como lo es la transferencia de archivos, correo electrónico o inicio de sesión remoto.
- En los “mensajes por trazas”, se involucra en la generación y/o recepción de mensajes de aplicación que se transmiten a través de la red.

### 2. Capa de Transporte:

- Se encarga de dividir los datos en segmentos, enumerarlos y se asegura de que lleguen a su destino de una manera ordenada y sin errores.
- Utilizado el protocolo **TCP (Protocolo de Control de Transmisión)**.
- Cuando un segmento se pierde o se daña durante la transmisión, **TCP** solicita su reenvío.
- **Se manejan los mensajes de la capa de aplicación y se establecen las conexiones de extremo a extremo.**

### 3. Capa de Internet:

- Se encarga de enrutar los paquetes de datos a través de la red.
- Se asignan las direcciones IP y se determina la mejor ruta para enviar los paquetes.
- **Los mensajes de la capa de aplicación se dividen en paquetes y se envían a través de esta capa hacia su destino.**

### 4. Capa de Interfaz de Red:

- Se relaciona con la transmisión física de los datos.
- Se manejan los detalles específicos en medio de la transmisión (Ethernet, Wi-Fi o fibra óptica).
- **Los paquetes se envían a través de esta capa hacia la red física.**

<sup>3</sup> Muestra cómo el consumidor busca en la lista de eventos el evento producido al cual le hicieron Request y luego el evento genera el Response

### c) Arquitectura de Netflix

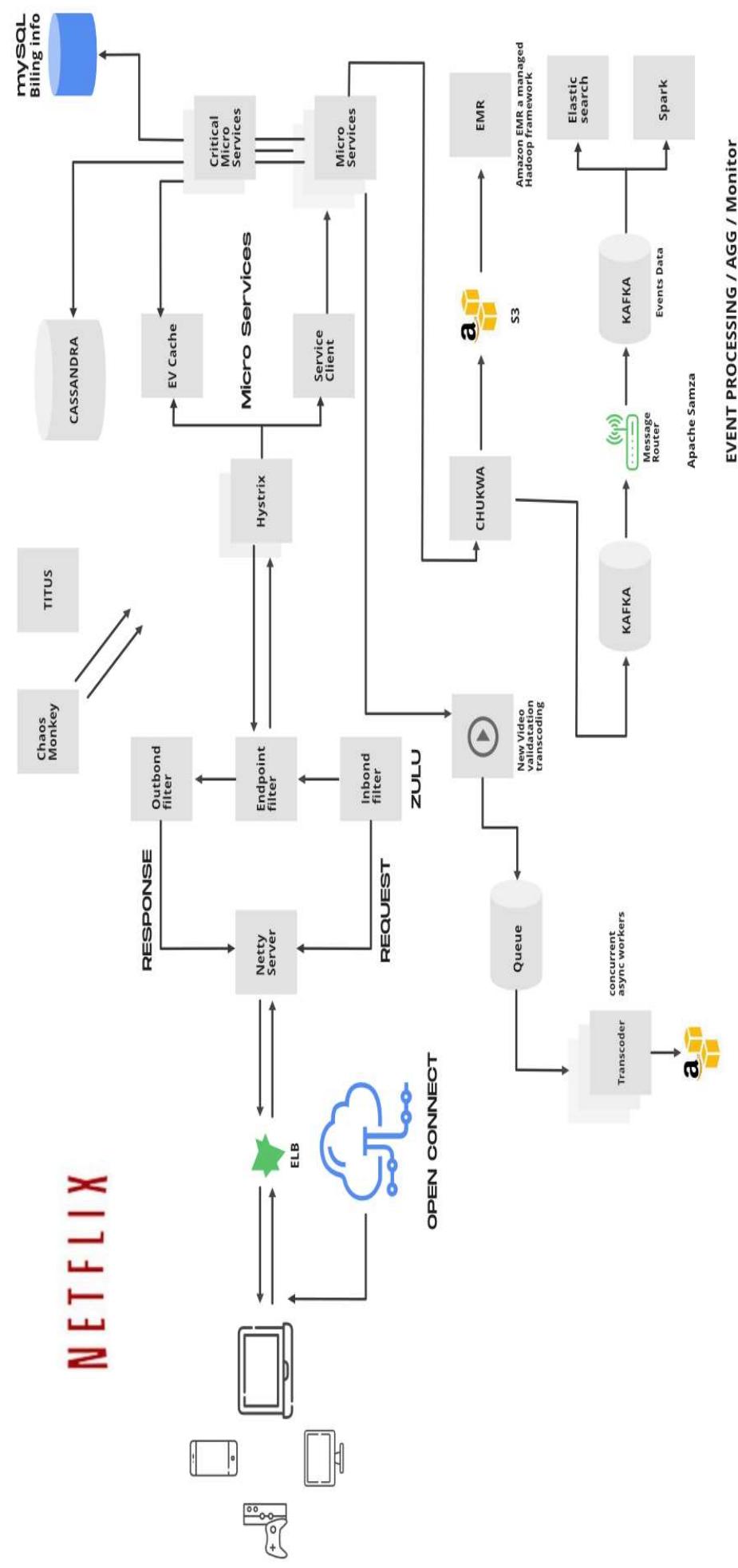


Figura 3:Arquitectura de alto nivel de Netflix<sup>4</sup>

---

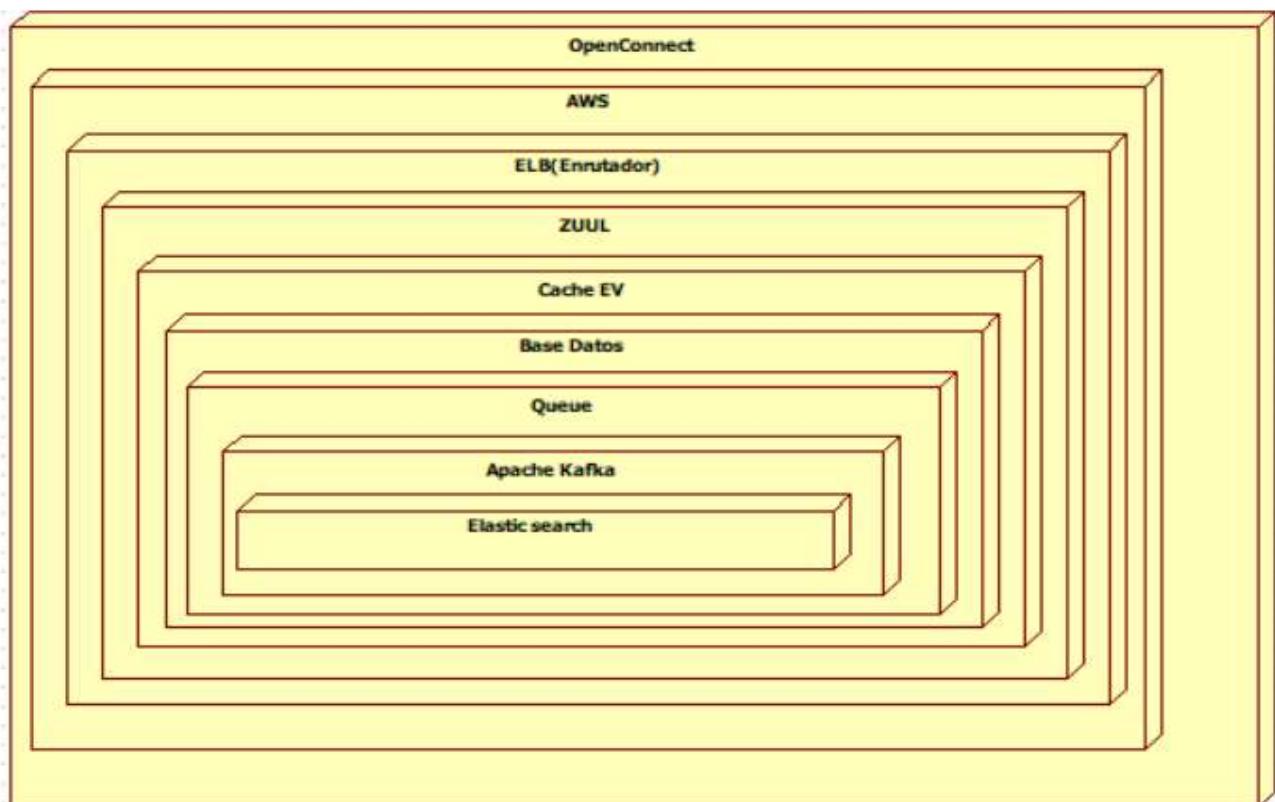
<sup>4</sup> Es una visualización general de cómo se compone la arquitectura de Netflix a un nivel abstracto.



Como se puede observar en la imagen, Netflix maneja dos brokers uno funciona como procesador y almacenamiento y el otro como conector, estas juntas con la columna vertebral que sostiene esa arquitectura y estas son: AWS y Open Connect.

La aplicación tiene 3 principales componentes:

- **Cliente:** Es cualquier dispositivo que utiliza el usuario
- **Netflix CND:** es una red de servidores distribuidos en diferentes ubicaciones geográficamente. **Open Connect (OC):** Es la propia CND (la red de entrega de contenido). Maneja y administra toda la transmisión de video desde la distribución, la reproducción.
- **Backend(Base de Datos):** maneja todo lo que no tiene que ver con la transmisión, procesamiento y distribución de los servidores. **Cloud Amazon Services:** se encarga de todos los eventos de los servicios.



*Figura 4: Diagrama de despliegue Netflix<sup>5</sup>*

---

<sup>5</sup> Interpretación de cómo se despliega el sistema de Netflix, diagrama hecho por nosotros



## d) Tipos de Streaming

### DISTRIBUCIÓN DE AUDIO

Para (Ribelles Garcia , 2011), la distribución de imagen en casi cualquier aspecto va asociada a la distribución de audio, como el audio tiene características diferentes a la imagen su tratamiento también es diferente en general más sencillo. El autor también dice que hay dos diferentes tipos de audio: el analógico y digital.

#### AUDIO ANALÓGICO

Convertido el audio en una señal eléctrica, ésta puede alimentar directamente el equipo, el altavoz o los cascos, por ejemplo, donde se vaya a reproducir. Salvando la potencia necesaria, este sistema de audio abarca desde micrófonos hasta sistemas de bafles de conciertos con éxito desde hace décadas, sobreviviendo en la era digital a través de las salidas RCA de los equipos CD, DVD, descodificadores de cable y satélite, o de las salidas de auriculares de 3,5 mm de todo reproductor MP3. Sin embargo, el ruido, las interferencias, la calidad de los conectores y del cable hacen mella en su resultado final; además, no incluye señalización que facilite su sincronía con el vídeo que pueda acompañar, por lo que se circunscribe actualmente al entorno de consumo.

#### AUDIO DIGITAL

El sonido siempre será analógico, pero la transmisión y almacenamiento se han digitalizado, por lo que así lo han protegido del ruido y lo han hecho inmune a las interferencias, a la vez que lo compatibiliza con los sistemas digitales. Además, del CD al MP3, el audio digital ha pasado de ser no comprimido a utilizar técnicas de compresión que veremos más adelante. Necesita conectores Canon XLS, aunque el formato AES también es el utilizado para empaquetar el audio digital en la señal de vídeo SDI, HD-SDI o 3G, transmitiendo todo el conjunto por un único cable coaxial.

Por lo anterior mencionado podemos decir que la distribución de audio es un sistema de sonorización a través del que se distribuye el audio que entra por una determinada fuente de sonido para emitir en diversas zonas.

Las señales de audio distribuidas de esta manera son principalmente música ambiental:

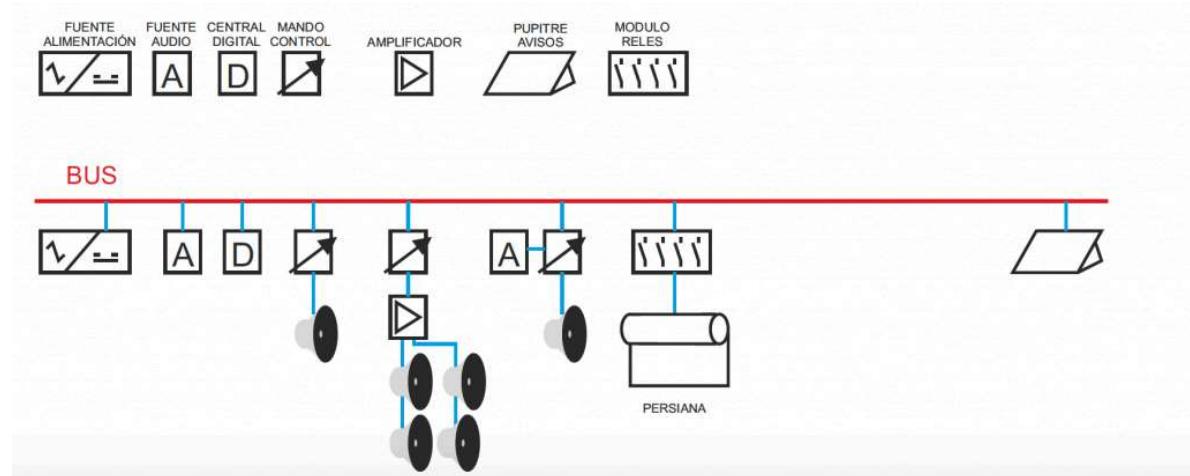
- Las instalaciones de audio distribuido se caracterizan por tener todos sus elementos conectados a un BUS, donde van todas las señales de audio, la alimentación, y las señales de control.

- Las fuentes musicales, que pueden estar en cualquier sitio, se conectan a este bus, produciendo la señal de audio que luego recogen los mandos.
- Los mandos también se conectan al BUS. En ellos está el amplificador de potencia, de 1 ó 2W. Si hace falta más potencia se le conecta un amplificador que se alimenta de la red a 220V.

Un sistema de audio distribuido también ofrece dos funciones interesantes:

- La intercomunicación entre mandos para dotar de comunicación interna a distintas salas o zonas, así como la posibilidad de escuchar el sonido que se produce en una habitación desde otra, como sucede con un “BABY PHONE”.
- La domótica y la gestión de avisos y alarmas. En el BUS hay unos hilos dedicados a las comunicaciones digitales, que se integran con el control de la domótica para la gestión de persianas y luces, así como con la gestión de alarmas (de intrusión, gas, inundación y humos).

#### instalación distribuida e instalación centralizada



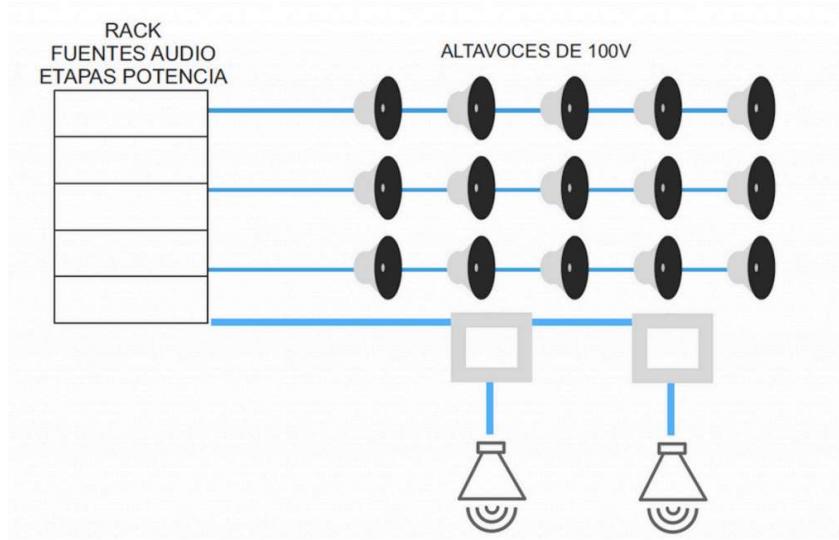
**Figura 5:Elementos conectados a un bus<sup>6</sup>**

Existen otros tipos de sistemas de instalación para sonorización, conocidos como sistemas de potencia o audio centralizado. En estos sistemas, todas las etapas de amplificación y las fuentes de música se agrupan en un solo rack. Desde este rack, se extienden los cables hacia los altavoces.

---

<sup>6</sup> En la imagen podemos ver como en los sistemas de audio distribuido todos los elementos están conectados a un BUS.

Debido a que los altavoces suelen estar ubicados lejos del rack, se eleva la tensión del audio a 100V para reducir la corriente que circula por los cables y evitar la pérdida de señal. Si se quiere ajustar el volumen de cada altavoz de forma independiente, es necesario instalar un atenuador en cada ubicación donde se desee controlar el volumen.



**Figura 6: instalación sencilla de audio con control y potencia centralizada, a línea de 100 V**

#### Elementos en un sistema de audio distribuido

**Los elementos esenciales que conforman una solución de audio distribuido son los siguientes:**

- Fuentes de audio: incluyen entradas RCA, reproductores de MP3, receptores FM y conexiones BLUETOOTH.
- Central de control digital: esta unidad gestiona el reloj, las alarmas, las funciones de vigilancia o monitorización, la programación de eventos semanales, y más.
- Mandos de control: son los dispositivos o paneles instalados en la pared que permiten gestionar las funciones del sistema, como ajustar el volumen, cambiar de canal, programar alarmas, etc. Existe una amplia gama de mandos, desde los más básicos de 1 canal hasta los más avanzados de 3 canales con receptor FM, BLUETOOTH, y capacidades para gestionar alarmas y sistemas domóticos.
- Amplificadores: se utilizan para aumentar la potencia del audio en las áreas donde sea necesario.
- Altavoces y bafles: estos componentes son esenciales para la salida del sonido.



- Módulos auxiliares: permiten el control de tareas de sistemas domóticos, integración de alarmas, control a través de Smartphone, entre otras funciones.

## DISTRIBUCIÓN DE VIDEO

Un tipo de sistema masivo cada vez más popular, pero que tal vez sea el menos restringido, es el sistema construido alrededor de una red casera. Estos sistemas generalmente consisten en una o más computadoras personales, aunque lo más importante es que integran aparatos electrónicos de consumo típicos como televisores, equipos de audio y video, dispositivos para juegos electrónicos, teléfonos (inteligentes), PDA, y otros aparatos personales en un solo sistema. Además, podemos esperar que todo tipo de dispositivos tales como implementos de cocina, cámaras de vigilancia, relojes, controladores de alumbrado, etc., estarán conectados a un solo sistema distribuido (Tanenbaum & Maarten Van, 2008).

López Fuentes, señala que Multimedia quiere decir “múltiples medios”. Los medios pueden ser desde texto e imágenes hasta animación, sonido o video. Específicamente, la multimedia es una combinación de texto, sonidos, imágenes o gráficos ya sea estáticos o en movimiento. El video es un tipo de contenido que combina los tipos de medios anteriores, por lo que se constituye en un caso ideal de multimedia, ( 2015).

Por lo anteriormente mencionado por los dos autores podemos definir a la distribución de video en dispositivos se refiere al proceso de enviar y reproducir contenido de video entre múltiples dispositivos. Esto puede incluir transmisión en vivo, video bajo demanda (VOD), y la transferencia de archivos de video

## COMPONENTES

Existen componentes que hacen efectiva la distribución de video estos son:

- Fuente de Video: El dispositivo o software que origina el video, como un servidor de medios, una computadora o una aplicación de streaming.
- Receptores de Video: Los dispositivos que reciben y reproducen el video, como televisores, monitores, teléfonos móviles y sistemas de proyección.

Algunos softwares utilizados en un sistema de distribución de Streaming (audio y video en tiempo real):

- Kafka
- Elastic Search
- AWS
- Azure Server
- Google Cloud



## Capítulo 2: Desarrollo

Como ya se mencionó la distribución de audio y video en dispositivos es una aplicación directa de los principios de los sistemas distribuidos. Ambos tipos de contenido requieren la transmisión eficiente de datos, la gestión de recursos distribuidos y la tolerancia a fallos para proporcionar una experiencia de usuario fluida y de alta calidad.

Para que se entienda mejor a manera de práctica se desarrolló un sistema distribuido que manda una señal de video a los demás equipos.

Para la realización de esta práctica se va a necesitar estas librerías en Python.

- pygame para sincronización de audio y video
- ffmpeg o moviepy para la manipulación y transcodificación de videos.
- socketio o websockets para la transmisión en tiempo real.

### Herramientas y Software que posiblemente se utilizarán:

- FFmpeg: Una herramienta poderosa para la manipulación y transcodificación de videos.
- Docker: Para contenedores y despliegue.
- Nginx: Para servir contenido estático y como servidor de proxy inverso.
- php Kafka: para el procesamiento y distribución de Streaming

### Infraestructura que propone utilizar:

- Servidor: Un servidor dedicado o una instancia en la nube (AWS, GCP, Azure).
- CDN (Content Delivery Network): Para mejorar la distribución y reducir la latencia.
- Base de Datos: PostgreSQL, MySQL, o MongoDB para almacenar metadatos de videos y gestionar usuarios.

Entonces comenzaremos por realizar las instalaciones cabe aclarar que se esta utilizando Windows así que, lo primero que hay que hacer es instalar python (en dado caso de que no lo tengamos instalado) para eso nos vamos a la página y descargamos python



## HERRAMIENTAS E INSTALACIÓN

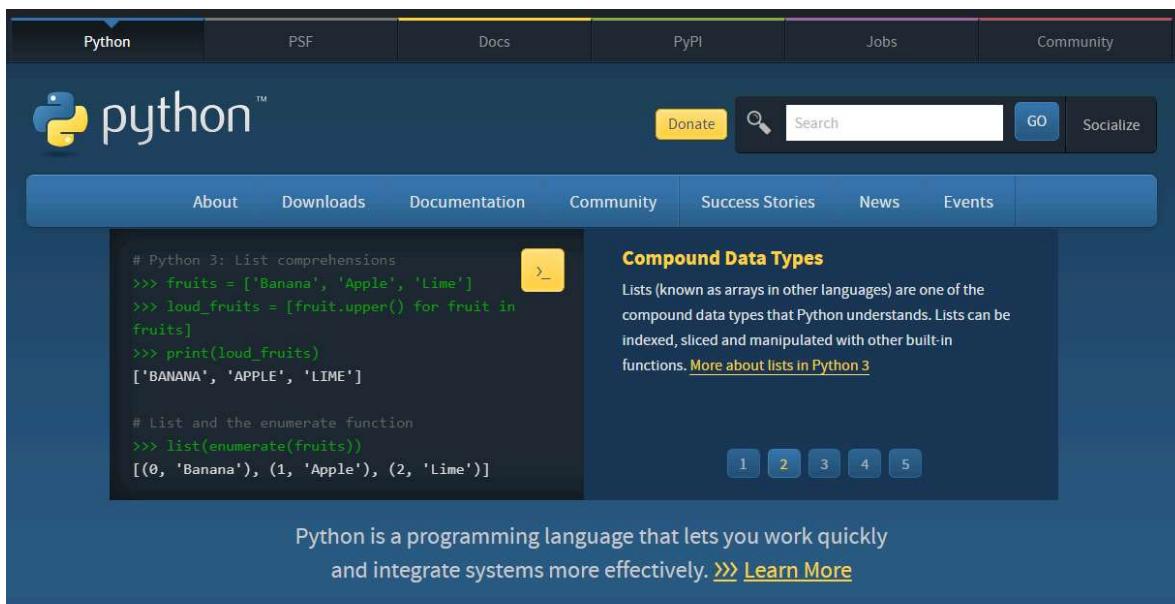


Figura 7: página principal de python <sup>7</sup>

una vez instalado python comenzamos a instalar las librerías que vamos a ocupar



Figura 8: instalación de biblioteca

Para la instalación de las librerías utilizaremos el comando pip ya que es el programa de instalación preferido.

una vez terminada la instalación de nuestras librerías empezaremos a crear nuestro servidor .Para eso nos guiaremos por los siguientes diagramas

---

<sup>7</sup> para descargar python puede irse la siguiente enlace:<https://www.python.org/>

## ARQUITECTURA PROPUESTA

### Diagrama de Despliegue

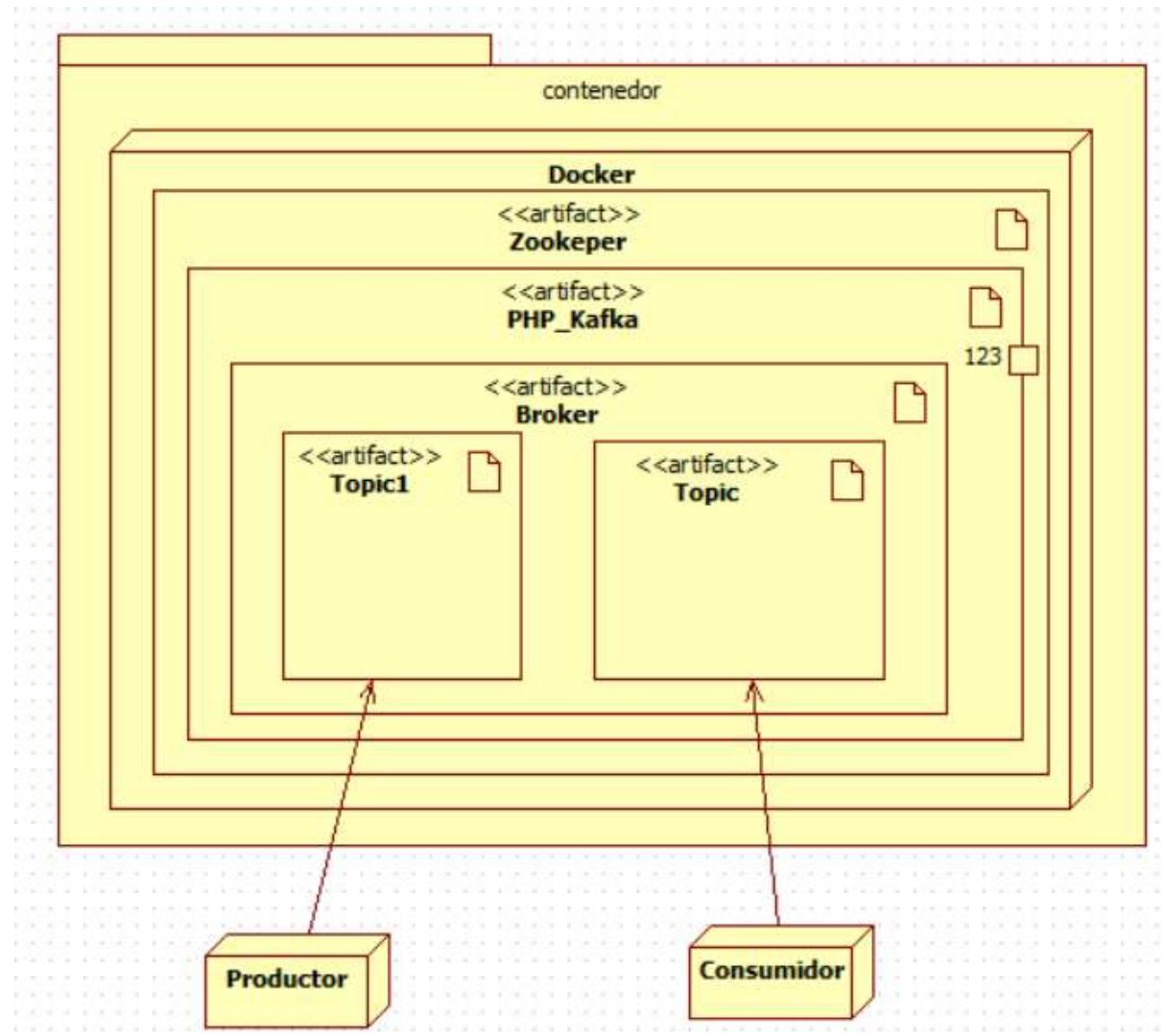
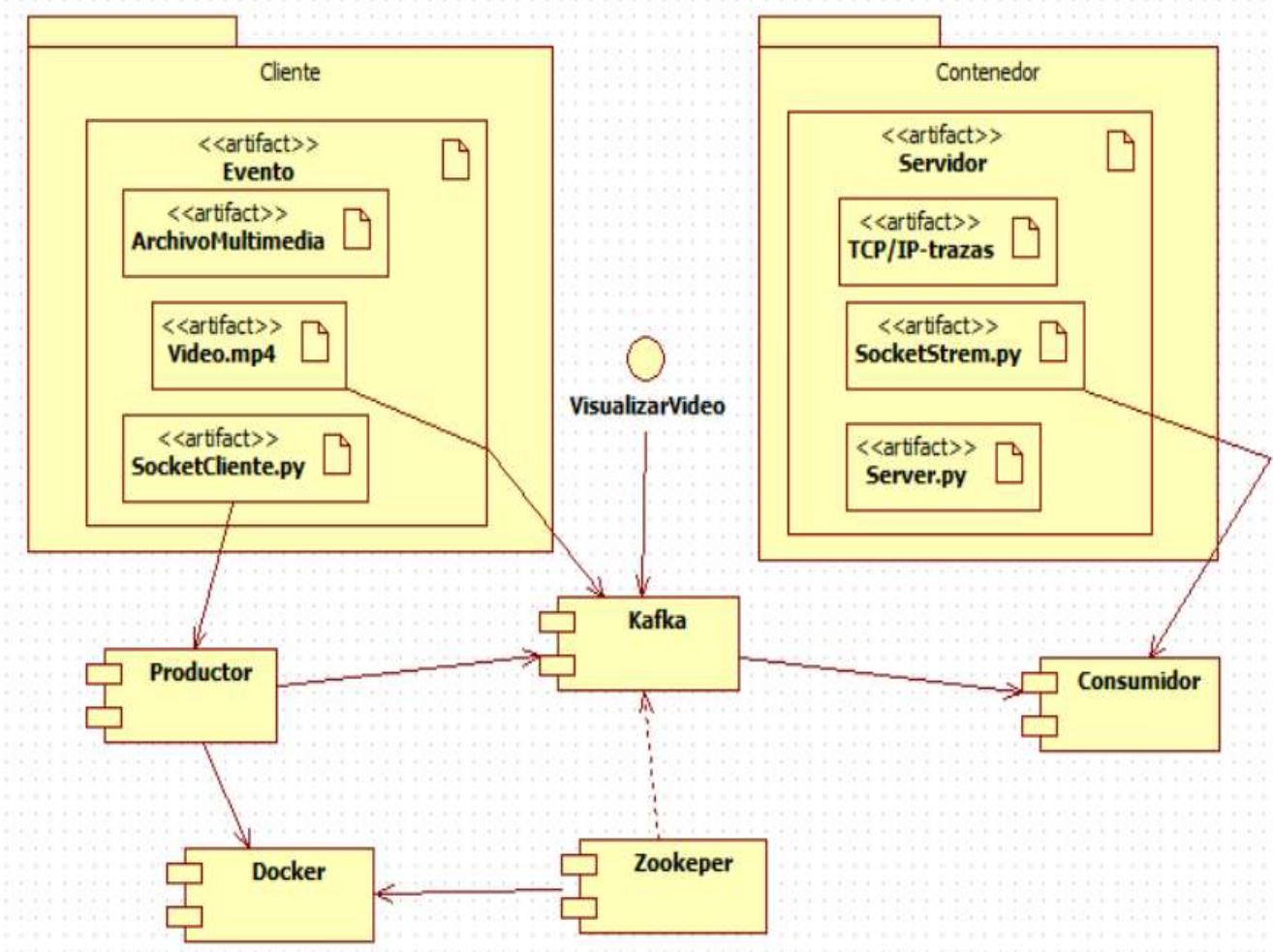


Figura 9: Diagrama de despliegue<sup>8</sup>

<sup>8</sup> El Diagrama solo se muestra como se está desplegando el sistema de manera en capas. Topic es un evento.



## Componentes



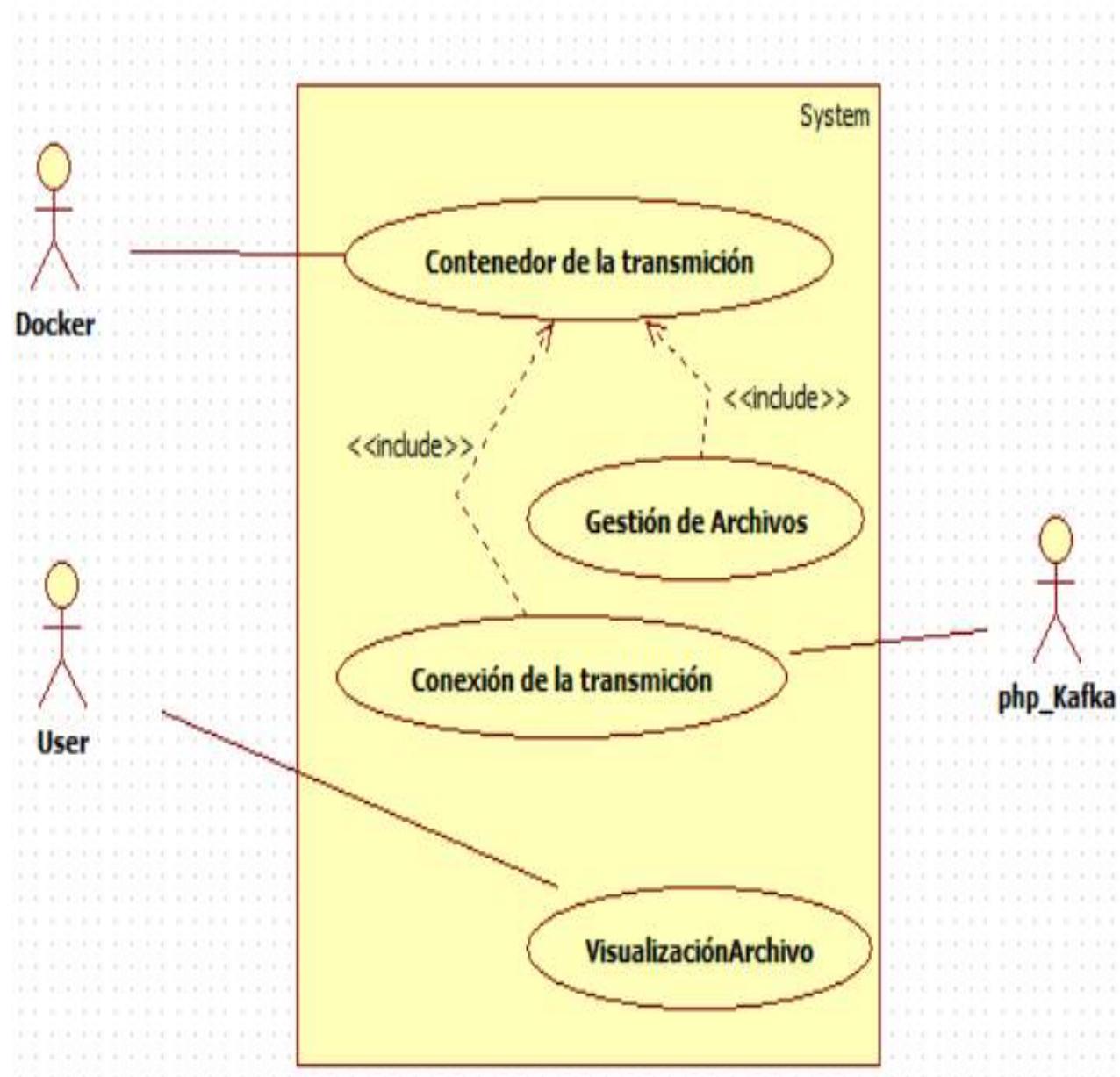
**Figura 10:** Diagrama de despliegue<sup>9</sup>

---

<sup>9</sup> El Diagrama solo se muestra como se está desplegando el sistema de manera en capas. Topic es un evento.



## Casos de uso

Figura 11: Diagrama de Casos de uso<sup>10</sup>

<sup>10</sup> Diagrama de casos de uso representando el comportamiento del sistema.

## Objetos

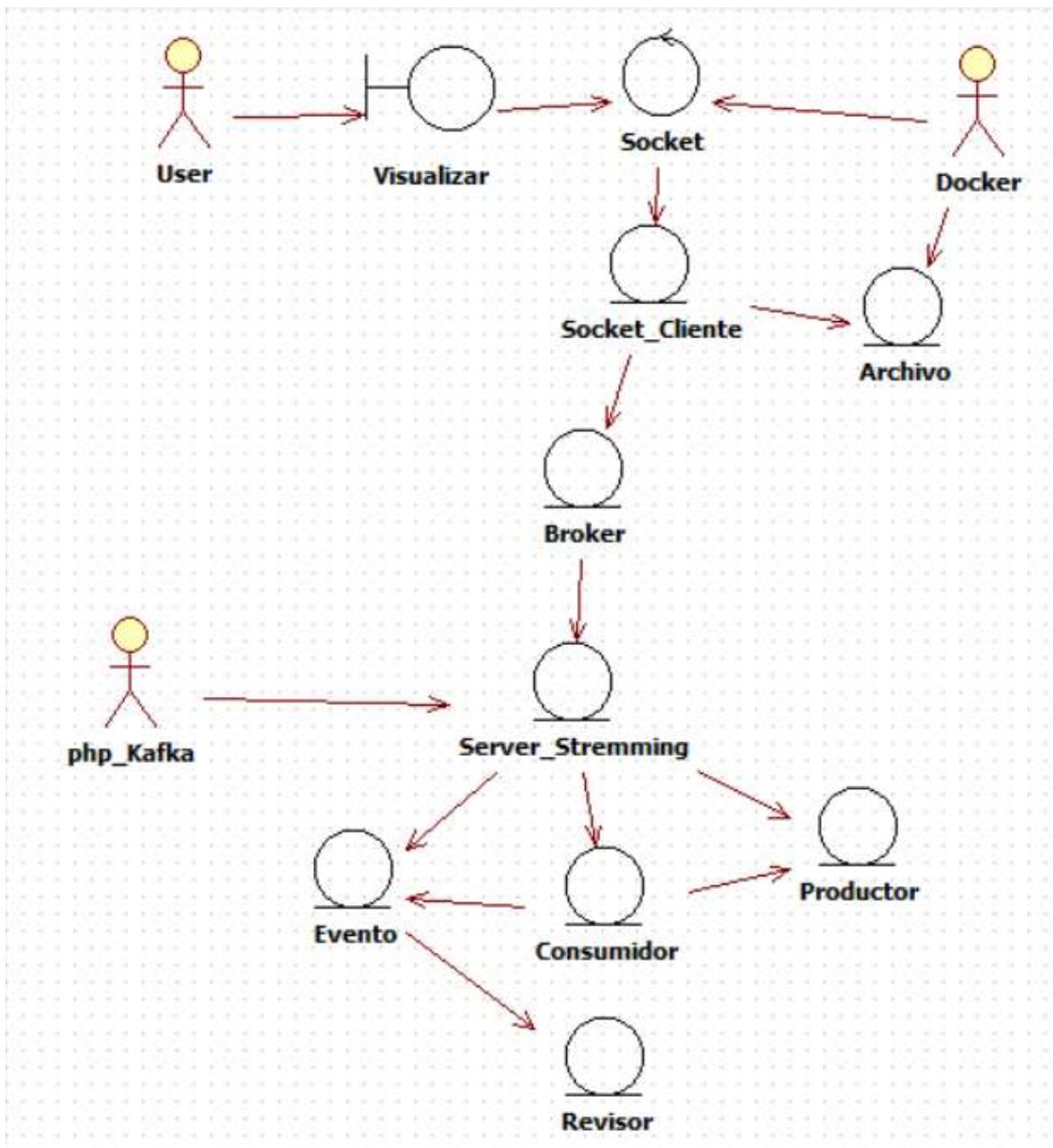


Figura 11: Diagrama de Casos de objetos<sup>11</sup>

<sup>11</sup> Diagrama de casos de objetos representando como se estructura el sistema en términos del modelo de negocio



## IMPLEMENTACIÓN DEL CÓDIGO

A continuación, se mostrará el código implementado en python con sus respectivas librerías para el funcionamiento del sistema distribuido de video.

El sistema que se alcanzó a implementar fue uno con arquitectura Cliente/Servidor, el cual tiene una estructura síncrona, donde se tiene que tener el servidor encendido para poder reproducir el cliente.

En este caso nosotros tenemos que iniciar cuales son los puertos de entrada y salida esto lo haremos esto desde Firewall



O usar el comando en modo administrador en la consola de Windows

```
netsh advfirewall firewall add rule name="PuertoReproductor" dir=in  
action=allow protocol=tcp localport=12345
```

```
C:\WINDOWS\system32>netsh advfirewall firewall add rule name="PuertoReproductor" dir=in action=allow protocol=tcp localport=12345  
Aceptar
```

Posteriormente a esto debemos importar las librerías moviepy, Pygame.

La primera es para reproducir gráficos y videos y la segunda es para la sincronización de audio, ya que se tenía importadas socket & Thread



Y para finalizar la ejecución se realiza primero la activación del servidor y posteriormente se ejecuta el cliente que este será el reproductor.

### Clase Reproducción (Cliente)

```
import random
from moviepy.editor import VideoFileClip
from pygame import mixer
import pygame
from pygame.locals import *
from threading import Thread# le integre hilos (procesos) para los fallos
import time#se integra la biblioteca de time para frenar los tiempos
```

```
# Simula la detección de una falla en un nodo
1 usage
def detectar_falla(nodos):
    if random.random() < 0.25: # Probabilidad de falla: 25%
        nodo_fallido = random.choice(nodos)
        print(f"Falla en el nodo {nodo_fallido} pero el usuario disfrutara de su video")
    return True
return False
```

```
# Función para reproducir el video
1 usage
def reproducir_video(segmento):
    print(f"Reproduciendo segmento {segmento}")
    video_path = "video.mp4"
    clip = VideoFileClip(video_path)
    clip.preview()
```

```
1 usage
def reproducir_video_con_falla(segmento, nodos):
    reproducir_thread = Thread(target=reproducir_video, args=(segmento,))
    reproducir_thread.start()

    while reproducir_thread.is_alive():
        if detectar_falla(nodos):
            print("reproducción normal, continua la reproducción")
            time.sleep(0.5) # Espera 0.5 segundos antes de verificar nuevamente
```



```
# Lista de nodos
nodos = ["Nodo1", "Nodo2", "Nodo3", "Nodo4"]

# Inicio de la reproducción del video
for segmento in range(1, 5):
    reproducir_video_con_falla(segmento, nodos)
```

## Clase Servidor

```
import socket
from threading import Thread
import random
import time
from moviepy.editor import VideoFileClip
from pygame import mixer
```

```
! usage
def detectar_falla(nodos):
    if random.random() < 0.25: # Probabilidad de falla: 25%
        nodo_fallido = random.choice(nodos)
        return f"Falla en el nodo {nodo_fallido}"
    return "No se detectaron fallas"
```



```

def manejar_solicitud(cliente_socket, nodos):
    # Reproducir el video
    video_path = "video.mp4"
    audio_path = "audio.mp3"

    clip = VideoFileClip(video_path)
    clip.preview()

    # Reproducir el audio
    mixer.init()
    mixer.music.load(audio_path)
    mixer.music.play()

    # Verificar las fallas mientras se reproduce el video y el audio
    for segmento in range(1, 5):
        cliente_socket.sendall(f"Reproduciendo segmento {segmento}\n".encode())
        time.sleep(5) # Simula la reproducción del segmento
        falla = detectar_falla(nodos)
        cliente_socket.sendall(falla.encode() + b'\n')

```

```

def manejar_cliente(cliente_socket, direccion, nodos):
    print(f"Cliente conectado desde {direccion}")
    manejar_solicitud(cliente_socket, nodos)
    cliente_socket.close()
    print(f"Conexión con {direccion} cerrada")

```

```

def servidor():
    host = "127.0.0.1"
    port = 12345

    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as servidor_socket:
        servidor_socket.bind((host, port))
        servidor_socket.listen()

        print("Servidor escuchando en", (host, port))

        while True:
            cliente_socket, direccion = servidor_socket.accept()
            cliente_thread = Thread(target=manejar_cliente, args=(cliente_socket, direccion, nodos))
            cliente_thread.start()

```



```
def reproducir_segmento(segmento):
    try:
        print(f"Reproduciendo segmento {segmento}")
        video_path = "video.mp4"
        audio_path = "audio.mp3"

        # Cargar el video
        video = mp.VideoFileClip(video_path)

        # Extraer el audio del video
        audio = video.audio

        # Reproducir el video
        pygame.init()
        pygame.display.set_caption("Reproducción de video")
        screen = pygame.display.set_mode(video.size)
        video.preview(fps=30, audio=audio, screen=screen)

    except Exception as e:
        print("Error en la reproducción del segmento:", e)
```

```
# Lista de nodos
nodos = ["Nodo1", "Nodo2", "Nodo3", "Nodo4"]

# Iniciar el servidor
servidor()
```



## Clase Cliente

```
import socket

def solicitar_reproduccion():
    host = "192.128.0.2"
    port = 1000

    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as cliente_socket:
        cliente_socket.connect((host, port))

        while True:
            mensaje = cliente_socket.recv(1024).decode()
            print(mensaje)
            if mensaje.startswith("Falla"):
                print("Se detectó una falla en un nodo!")
            elif mensaje.startswith("No se detectaron"):
                print("No se detectaron fallas en los nodos.")

# Solicitar la reproducción del video
solicitar_reproduccion()
```



## Demostración de funcionamiento

En el repositorio se podrá encontrar un video de como se reproduce asincrónamente en 3 distintos dispositivos.

The screenshot shows a Windows desktop environment. In the center is a video player window titled "MoviePy" displaying a video of a man singing into a microphone. The video has text overlays: "ENCERRADOS", "PERO", "Enfiestados", "LIVE", and "VOL.1". To the left of the video player is a code editor window showing Python code for a client socket program. Below the code editor is a terminal window showing the output of running the program, which includes messages about socket errors and fallback to normal reproduction. At the bottom of the screen is a taskbar with various icons, and the system tray shows the date and time as 09:42 p.m. on 20/05/2024.

```
import socket

# Función para conectar al servidor y solicitar
# uso
def solicitar_reproducción():
    host = "192.128.0.2"
    port = 1000

    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as cliente_socket:
        cliente_socket.connect((host, port))

        while True:
            mensaje = cliente_socket.recv(1024)
            print(mensaje)
            if mensaje.startswith("Falla"):
                print("Se detectó una falla en")
            elif mensaje.startswith("No se detectó ninguna falla"):
                print("No se detectó ninguna falla, continuando la reproducción")
            else:
                print("Mensaje recibido:", mensaje.decode())

if __name__ == "__main__":
    solicitar_reproducción()
```

```
C:\Users\HP\AppData\Local\Programs\Python\Python38\python.exe C:\Users\HP\Documents\universidad\semestres\2024-1\distribuidos\proyecto\reproducción.py
Reproduciendo segmento 1
Falla en el nodo Nodo4 pero el usuario disfrutara de su video
reproducción normal, continua la reproducción.
Falla en el nodo Nodo3 pero el usuario disfrutara de su video
reproducción normal, continua la reproducción.
```



## CAPÍTULO 3: OBSERVACIONES FINALES

### **Observación Balamm:**

por cuestiones de gestión de tiempos y recursos el desarrollo se ve afectado considerablemente por el simple hecho que el tiempo en realizar la investigación ocupa la mayoría del tiempo de gestión ya que no solo se trata de investigar, sino de comprender y analizar los procesos que se realizan al generar este sistema. como también el que al investigar se pudo notar que se requería de varios recursos al que no tenemos acceso y al buscar alternativas el costo aumenta en cuestión de tiempo

### **Observaciones ANIZZ:**

Para la realización de este sistema se tuvo que hacer mucha investigación, al inicio buscando información general, para tratar de comprender conceptos y funcionalidades, hasta ir encontrando palabras clave y estructuras, buscando la información encontré la arquitectura general del servicio de Streaming más conocido, NETFLIX, analizando de manera cuidadosa y exhaustiva, la forma en cómo reestructurar su arquitectura a las herramientas que teníamos propuestas al inicio, utilizando contenedores, utilizando el protocolo TCP/IP y Cliente/Servidor, observando que la arquitectura utilizada es la de Consumidor y Productor, la cual se conecta por medio de broker, y distintos eventos, esto es lo que se puede ver en la arquitectura de Netflix, aunque mucha de su arquitectura utilizada es por medio de software libre, utiliza AWS para el procesamiento de datos de video por el protocolo TCP/IP de la capa de transporte donde es la división de datos por segmentación, llegando a la capa de aplicación la cual es la más cercana al usuario.

### **Observaciones Luis**

Por tema investiga y puesto en práctica con anterioridad se puede decir que un sistema distribuido de video es muy complejo ya que se requieren cortos componentes para su buen funcionamiento, estos componentes como docker, tomandolo como ejemplo si no se saben manejar es posible que la distribución de video o audio no vaya a funcionar por lo que se corre el riesgo de que el sistema colapse, por ese motivo la elaboración de la práctica se tardó ya que debido a problemas de manejo y de tiempo no se logró completar bien el objetivo, solo se logró una distribución local.

### **Comentario de Raul:**

Desde mi punto de vista me di cuenta que al hacer el código hicieron muchas pruebas para poder hacer ciertas ejecuciones como por ejemplo el proceso de hilos, threading e inclusive se vieron errores de librerías que nos costó demasiado poder incorporarlas ya que las versiones no compaginan.



## Bibliografía

- López Fuentes, F. (2015). *Sistemas distribuidos*. México: Universidad Autónoma Metropolitana. Recuperado el 2024 de mayo de 18, de [http://dccd.cua.uam.mx/libros/archivos/03IXStream\\_sistemas\\_distribuidos.pdf](http://dccd.cua.uam.mx/libros/archivos/03IXStream_sistemas_distribuidos.pdf)
- Martin, S. (21 de noviembre de 2004). Recuperado el 2024 de mayo de 17, de UTN: FacultadRegional Mendoza: [http://www1.frm.utn.edu.ar/soperativos/archivos/sistemas\\_distribuidos.pdf](http://www1.frm.utn.edu.ar/soperativos/archivos/sistemas_distribuidos.pdf)
- Ribelles Garcia , A. (2011). *Conceptos básicos de vídeo y audio*. Barcelona, Universitat Oberta de Catalunya.: Plataformas de distribución de contenidos. Recuperado el 18 de mayo de 2024, de [https://openaccess.uoc.edu/bitstream/10609/70605/7/Plataformas%20de%20distribuci%C3%B3n%20de%20contenidos\\_M%C3%B3dulo%201\\_Conceptos%20b%C3%A1sicos%20de%20v%C3%ADdeo%20y%20audio.%20Introducci%C3%B3n.pdf](https://openaccess.uoc.edu/bitstream/10609/70605/7/Plataformas%20de%20distribuci%C3%B3n%20de%20contenidos_M%C3%B3dulo%201_Conceptos%20b%C3%A1sicos%20de%20v%C3%ADdeo%20y%20audio.%20Introducci%C3%B3n.pdf)
- Tanenbaum, A. S., & Maarten Van, S. (2008). *SISTEMAS DISTRIBUIDOS:Principios y paradigmas* (segunda edición ed.). (A. Jiménez Govea, Ed., J. O. García Pérez, & R. Navarro Salas, Trads.) México: PEARSON EDUCACIÓN. Recuperado el 2024 de mayo de 18, de [https://moodle.uneg.edu.ve/pluginfile.php/114008/mod\\_page/content/6/Sistema-Distribuidos-esp.pdf](https://moodle.uneg.edu.ve/pluginfile.php/114008/mod_page/content/6/Sistema-Distribuidos-esp.pdf)

