



Laboratorio de Sistemas distribuidos

Requerimiento final

Fecha de sustentación 2 de diciembre o 9 de diciembre

Contexto del proyecto a realizar

En este Proyecto se reutilizará un **cliente** construido en javaScript, el cual se conecta con un **servidor de Streaming** para descargar vía WEB-GRPC fragmentos de audio correspondientes a una canción. Entre el cliente y el servidor de Streaming existirá un **Servidor Envoy** que traduce WEB-GRPC (utiliza http v 1.0)a GRPC (utiliza http v 2.0).

El cliente web se conectará con un **microservicio de reacciones** mediante webSockets. Luego de la conexión, los clientes pueden enviar reacciones asociadas a la canción que está reproduciendo. El microservicio reenviará la reacción a los clientes que están reproduciendo la misma canción mediante un callBack y utilizando la conexión previamente creada. Siga el prototipo planteado en la última práctica del laboratorio.

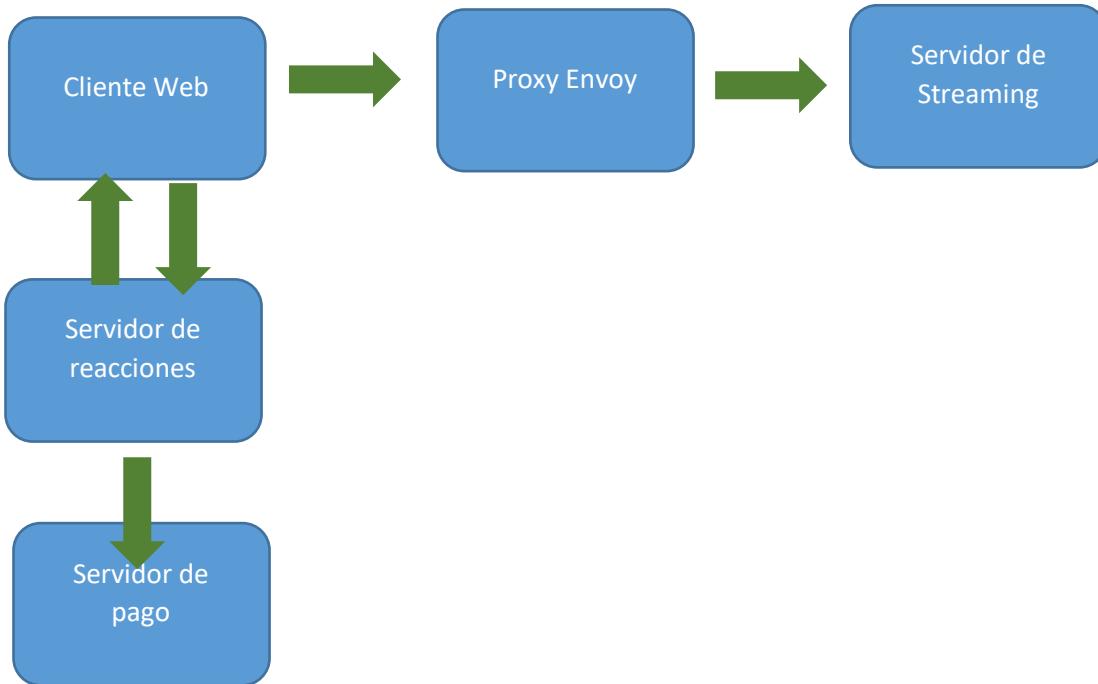
Cuando un cliente comienza a reproducir una canción se debe enviar el nickname a los usuarios que están reproduciendo la misma canción. El microservicio informará a los usuarios que un nuevo usuario ha comenzado a reproducir su canción. Cuando un cliente pausa la reproducción de la canción se debe enviar su nickname a los usuarios que están reproduciendo la misma canción. El microservicio informará a los usuarios que un usuario ha pausado de reproducir su canción.

Cuando un usuario envía al servidor de reacciones una reacción, el servidor para cada reacción obtiene un token generado por un **servidor de pagos** y luego envía un pago por cada reacción, el cual se almacena en el servidor de pagos.

Considere tres aspectos:

- Simule que un pago no se logró realizar y desde el servidor de reacciones utilice dos mecanismos el reenvío del pago y un temporizador entre reenvíos.
- Si un token fue usado en un pago previo, no permita registrar el pago.
- Una reacción vale \$10, cuando se alcance 5 reacciones no permita reenviar más reacciones. El almacenamiento de los pagos se hace en el servidor de pagos.

En el siguiente diagrama se muestran los clientes y servidores. Únicamente debe desarrollar los servidores de reacciones y pagos. Los demás, están adjuntos al parcial.



Para realizar el proyecto se adjunta el servidor de streaming, el servidor envoy, la configuración del servidor envoy y los archivos JavaScript del cliente web. **En el cliente web modifique los archivos index.html para crear la interface gráfica y el archivo funciones.js para registrar y des registrar el cliente de los canales del servidor de reacciones.**

El servidor de reacciones y servidor de pagos pueden ser realizados en java y ejecutados en Windows o Linux. Si ejecuta el servidor de reacciones y servidor de pagos en Windows considere la comunicación entre la máquina virtual y Windows.

Rubrica de calificación

- 1) (Valor 0,5) Aparición de los nickname de los usuarios que comienzan a reproducir la canción
- 2) (Valor 0,5) Aparición de los nickname de los usuarios que pausan la reproducción de la canción
- 3) (Valor 0,5) Aparición de las reacciones de los usuarios que están reproduciendo la canción
- 4) (Valor 0,5) Simulación de un pago que no se logró realizar y uso de los mecanismos de reenvío del pago y un temporizador entre reenvíos.
- 5) (Valor 0,5) Si un token fue usado en un pago previo, no permitir registrar el pago.
- 6) (Valor 0,5) Mostrar con un eco los pagos en el servidor de pagos cuando se envían las reacciones
- 7) (Valor 0,5) Cuando un usuario alcance \$50 por las reacciones, no permitir reenviar las reacciones.
- 8) (Valor 0,5) Animaciones generadas asociadas a los nickname y reacciones
- 9) (Valor 0,5) Explicación del código fuente
- 10) (Valor 0,5) Simulación de los reintentos



Pre requisites en linux

1) Instalar node.js

En el Proyecto necesitamos instalar node.js para permitir la gestión de dependencias con npm. Ejecutar los siguientes comandos.

sudo apt update

sudo apt install nodejs npm

2) Instalación de paquetes para usar GRPC en JavaScript

npm install grpc-web google-protobuf

google-protobuf,

Es un paquete que permite:

- Obtener las clases necesarias para serializar y deserializar mensajes Protocol Buffers en JavaScript
- Implementación oficial generada por Google
- Utilidades para trabajar con .proto compilados a .js

grpc-web

Es la biblioteca oficial de Google para usar gRPC desde el navegador o desde aplicaciones web. Permite:

- Hacer llamadas gRPC sobre HTTP/1.1
- Usar un *envoy* para traducir del navegador al backend gRPC real
- Consumir servicios gRPC desde Angular, React, Vue, etc.

3) Utilizar un servidor envoy

El navegador NO soporta gRPC nativo, por estas razones:

- El navegador solo puede usar HTTP/1.1 en la mayoría de casos.
- gRPC usa HTTP/2 y streaming completo.
- Hay restricciones de CORS, headers y seguridad.

Por eso no es posible conectar Angular, React o JavaScript directamente a un servidor gRPC. Por tal motivo, gRPC-Web es una versión adaptada para navegadores, pero necesita un proxy que traduzca las peticiones al backend. Envoy recibe gRPC-Web (desde el navegador, HTTP/1.1) y lo



transforma en gRPC normal (HTTP/2) hacia el backend en Go. Para ejecutar el servidor envoy, primero debemos descargar el servidor de la plataforma y el archivo de configuración y posteriormente ejecutar el siguiente comando:

```
./envoyServer -c envoyConfig2.yaml --disable-hot-restart
```

Imágenes de la aplicación funcionando

Para ejecutar la aplicación

Ejecute el servidor de Streaming con el siguiente comando:

```
go run main/servidor.go
```

Ejecute el servidor de envoy con el siguiente comando

```
./envoyServer -c envoyConfig2.yaml --disable-hot-restart
```

Lance el cliente

Abra el archivo index.html en el navegador

Al abrir la página se muestra un input para ingresar el nombre de la canción y un select para establecer el formato de la canción

The screenshot shows a web browser window titled "Cliente Audio gRPC-Web". The address bar displays the URL "file:///home/ingesisis/Documentos/practicas rpc/practica 5/versión 2/cliente html/index.html". The page content is as follows:

Transmisión de Audio con gRPC-Web

La conexión se intentará al servidor ubicado en localhost:8080

Título de la canción: Formato: mp3

Esperando petición...

A large empty rectangular area below the message is likely a placeholder for audio playback.

Cuando un usuario solicita una canción se envía una petición al servidor envoy que corre sobre el puerto 8080. Al recibir el primer fragmento el reproductor se vuelve visible.



The screenshot shows a web browser window titled "Cliente Audio gRPC-Web". The URL is "file:///home/ingesis/Dокументos/practicas rpc/practica 5/versión 2/cliente html/index.html". The page displays a media player interface with a play/pause button, volume slider, and time indicator (0:04 / 3:46). Below the player are input fields for "Título de la canción" (cancion1) and "Formato" (mp3), and a "Pedir Canción" button. A message "Esperando petición..." is displayed. In the bottom right corner of the page area, there is a text box containing log entries:

```
[8:13:14 p. m.] Primer fragmento recibido. Reproductor mostrado.  
[8:13:15 p. m.] Reproducción iniciada (play).
```

Al pausar la canción se muestra un eco indicando que se ha pausado la canción. En el archivo funciones.js hay un listener que se ejecuta cada vez que el usuario pausa o reproduce una canción. Utilice este elemento para registrar o des registrar al usuario de los canales de notificaciones.

The screenshot shows a web browser window titled "Cliente Audio gRPC-Web". The URL is "file:///home/ingesis/Dокументos/practicas rpc/practica 5/versión 2/cliente html/index.html". The page displays a media player interface with a play/pause button, volume slider, and time indicator (0:16 / 3:46). Below the player are input fields for "Título de la canción" (cancion1) and "Formato" (mp3), and a "Pedir Canción" button. A message "Esperando petición..." is displayed. In the bottom right corner of the page area, there is a text box containing log entries:

```
[8:13:14 p. m.] Primer fragmento recibido. Reproductor mostrado.  
[8:13:15 p. m.] Reproducción iniciada (play).  
[8:13:29 p. m.] Reproducción pausada (pause).  
[8:13:51 p. m.] Reproducción iniciada (play).  
[8:13:55 p. m.] Reproducción pausada (pause).
```

En la consola del navegador se muestran ecos que establecen que se ha realizado una conexión con el servidor de streaming y se están recibiendo los fragmentos.

The screenshot shows the Chrome DevTools Console tab. The log output includes:

```
Exported iniciar_streaming_cancion to window
Iniciando llamada gRPC-Web a través de Envoy...
Solicitud preparada: > Object { título: "cancion1", formato: "mp3" }
Fragmento recibido desde el servidor. Tamaño: 65536 bytes
Fragmento recibido desde el servidor. Tamaño: 41055 bytes
Transmisión de la canción finalizada. Preparando reproducción...
Reproducción de la canción iniciada con éxito.
```



En la consola del servidor envoy se muestran ecos que determinan que el servidor está arriba.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE
○ ingesion@debian:~/Documentos/practicas rpc/practica 5/versión 2$ ./envoyServer -c envoyConfig2.yaml --disable-hot-restart
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE
pe envoy.config.route.v3.VirtualHost Using deprecated option 'envoy.config.route.v3.VirtualHost.cors' from file route_components.proto. This configuration will be removed from Envoy soon. Please see https://www.envoyproxy.io/docs/envoy/latest/version_history/version_history for details. If continued use of this field is absolutely necessary, see https://www.envoyproxy.io/docs/envoy/latest/configuration/operations/runtime#using-runtime-overrides-for-deprecated-features for how to apply a temporary and highly discouraged override.
[2025-11-24 20:16:58.537][16722][info][config] [source/server/configuration_impl.cc:164] loading stats configuration
[2025-11-24 20:16:58.537][16722][info][runtime] [source/common/runtime/runtime_impl.cc:551] RTDS has finished initialization
[2025-11-24 20:16:58.537][16722][info][upstream] [source/common/upstream/cluster_manager_impl.cc:248] cm init: all clusters initialized
[2025-11-24 20:16:58.543][16722][warning][main] [source/server/server.cc:1012] There is no configured limit to the number of allowed active downstream connections. Configure a limit in `envoy.resource_monitors.global_downstream_max_connections` resource monitor.
[2025-11-24 20:16:58.543][16722][info][main] [source/server/server.cc:1031] all clusters initialized. initializing init manager
[2025-11-24 20:16:58.543][16722][info][config] [source/common/listener_manager/listener_manager_impl.cc:1009] all dependencies initialized. starting workers
[2025-11-24 20:16:58.545][16722][info][main] [source/server/server.cc:1051] starting main dispatch loop
```

En la consola del servidor de Streaming se muestran ecos que determinan que el servidor está arriba.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE
ingesion@debian:~/Documentos/practicas rpc/practica 5/versión 2/servidor$ go run main/servidor.go
Creando instancia única del Logger...
Abriendo archivo de log...
Servidor gRPC escuchando en :50051...
```

Si realiza un cambio en el cliente web (archivo cliente_receptor_fragmentos) debe ejecutar el comando npx webpack con el fin de descargar librerías y resolver imports, y posteriormente vuelva a abrir el archivo index.html

```
ingesion@debian:~/Documentos/practicas rpc/practica 5/versión 2/cliente html$ npx webpack
asset bundle.js 281 KiB [compared for emit] (name: main)
modules by path ./*.js 17.7 KiB
  ./client-web.js 3.39 KiB [built] [code generated]
  ./servicios_pb.js 11.1 KiB [built] [code generated]
  ./servicios_grpc_web_pb.js 3.26 KiB [built] [code generated]
modules by path ./node_modules/ 256 KiB
  ./node_modules/google-protobuf/google-protobuf.js 230 KiB [built] [code generated]
  ./node_modules/grpc-web/index.js 26.1 KiB [built] [code generated]
webpack 5.103.0 compiled successfully in 593 ms
ingesion@debian:~/Documentos/practicas rpc/practica 5/versión 2/cliente html$
```