SISTEMA DE STREAMING DISTRIBUIDO PARA DJS TIPO CLOUD COMPUTING, BASADO EN OGG PARA LA DIFUSIÓN DE MÚSICA MEZCLADA EN VIVO PARA UN CLIENTE ANDROID 4.4.

CAMILO ALEJANDRO CANO CASTRO

UNIVERSIDAD SANTO TOMÁS DE TUNJA
FACULTAD DE INGENIERÍA DE SISTEMAS
INGENIERÍA DE SISTEMAS
TUNJA
2015

SISTEMA DE STREAMING DISTRIBUIDO PARA DJS TIPO CLOUD COMPUTING, BASADO EN OGG PARA LA DIFUSIÓN DE MÚSICA MEZCLADA EN VIVO PARA UN CLIENTE ANDROID 4.4.

CAMILO ALEJANDRO CANO CASTRO

Trabajo de grado presentado como requisito para optar por el título de:
Ingeniero de sistemas

Tutor:
ALEX PUERTAS GONZÁLEZ
Ingeniero de sistemas

UNIVERSIDAD SANTO TOMÁS DE TUNJA
FACULTAD DE INGENIERÍA DE SISTEMAS
INGENIERÍA DE SISTEMAS
TUNJA
2015

CONTENIDO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABLAS

Pág.

LISTA DE ANEXOS

Pág.

INTRODUCCIÓN

Los sistemas de streaming han sido una tecnología ampliamente utilizada para la transmisión de datos de audio y video con el fin de hacer llegar la información a una audiencia mayor. Esta tecnología es utilizada en diferentes campos y progresivamente aparecen nuevos aplicativos para facilitar o mejorar actividades, como actualmente sucede con los DJs. Para estos actores de la escena musical han surgido diferentes necesidades de interacción y alcance para ellos y su audiencia, que desean compartir experiencias en tiempo real.

La tecnología streaming, surgió a mediados de los 90 con el fin de facilitar la transmisión de multimedia a través de internet. A partir de esta década se comenzó el desarrollo de diferentes plataformas que ofrecían servicios de entretenimiento, seguridad, comunicación y difusión de información para hacer uso de la red mundial y proveer nuevos alcances a las personas del mundo.

En el siguiente documento, se conocerá la propuesta de desarrollo de un sistema de streaming para DJs, en la cual se podrá conocer: el problema que se desea solucionar; la metodología paso a paso que será utilizada para cumplir la finalidad deseada, ya sea para comprobar la hipótesis o desarrollar el sistema; la justificación que explica el porqué del proyecto; el estado del arte que representa cómo se encuentra actualmente el desarrollo de los sistemas en esta área y por último se encontrarán los marcos que demostraran el conocimiento de las teorías y herramientas que se requieren para el desarrollo del sistema.

Posterior a la teoría aplicada en el planteamiento del proyecto, se encontrará documentada la ejecución de las diferentes fases del ciclo del software, en la cuales se demostrará como se realizó cada uno de los módulos a partir del análisis, pasando por el diseño y llegando hasta la configuración del back end en el servicio de elastic cloud computing de Amazon y el desarrollo de las aplicaciones clientes basadas en el estándar HTML5 con soporte nativo a trasmisión OGG Vorbis en el sistema operativo Android.

1. TITULO

Sistema de streaming distribuido para djs tipo cloud computing, basado en ogg para la difusión de música mezclada en vivo para un cliente Android 4.4.

2. TEMA

El tema de este trabajo de grado obedece a la implementación de un sistema de streaming distribuido para DJs tipo cloud computing, basado en OGG para la difusión de música mezclada en vivo para un cliente Android.

En el ámbito de lo que se desea estudiar,se denota en el uso de sistemas para brindar más alcances a los DJs respecto a la transmisión del audio mixing en vivo, de manera que se conozcan las implicaciones en el rendimiento de los recursos utilizando formatos libres como OGGVorbis, teniendo en cuenta el uso de la transmisión remota por internet como una herramienta estratégica para la distribución de contenido de diferentes tipos y desde diferentes fuentes para lograr un impacto positivo en los grupos que lo consumen, como en el área de: videojuegos, reporteros independientes, deportes extremos, festivales y una serie de eventos de interés general, en donde es preciso considerar la realización de un análisis para estudiar los efectos positivos de una transmisión con formatos open source.

En cuanto a los avances tecnológicos del sector de los DJs, se puede destacar el consolidado y constante desarrollo de plataformas realizado por parte de las empresas del sector, en la cuales convergen el hardware y software para proveer herramientas donde sean compatibles, tanto la interfaz física como la digital para realizar transmisión remota. De esta manera se pueden desarrollar aplicativos de transmisión útiles para los DJs que cuenten con un conjunto de herramientas necesarias para realizar una transmisión de calidad.

Aparte de la evolución de estas herramientas, también se puede tener en cuenta el nuevo paradigma de computación en la nube que facilita el acceso a infraestructura robusta de alto rendimiento, inmediata escalabilidad, alta prestación y bajo costo, requerida para una aplicación de transmisión en vivo que pueda requerir más o menos recursos en cuestión de minutos.

2.1. ALCANCE

- El DJ podrá conectarse y transmitir mediante su propio canal.
- El DJ podrá conectarse en el momento que desee, mientras que el servicio esté en línea.
- El DJ podrá transmitir audio mixing desde un controlador DJ con interfaz digital.
- El DJ podrá usar otras aplicaciones para realizar la difusión como: Virtual DJ o Traktor.
- El DJ tendrá un canal propio con su información personal.
- El DJ podrá escoger una imagen para su perfil entre una lista predeterminada.
- El canal del DJ estará protegido por usuario y contraseña.

- El oyente podrá obtener la aplicación para Android del landing page de la aplicación.
- El oyente podrá ver la lista de canales en línea y escoger el que desea escuchar.
- El oyente podrá observar el canal del DJ con la información personal de este.
- La aplicación para Android podrá ser obtenida desde el landing page.

2.2. DELIMITACIONES

- Cada dj puede poseer sólo un (1) canal.
- La aplicación móvil será compatible con versiones iguales o posteriores del sistema operativo Android 4.4, ya que esta se usará como referencia para el desarrollo
- El DJ no podrá realizar audio mixing y transmitirlo desde dispositivos sin interfaz digital.
- El oyente no tendrá comunicación p2p con un DJ ya sea chat, VoIP o videoconferencia.
- No es posible hacer una topología de comunicación en anillo entre servidores para difundir todos los puntos de montaje en todos los servidores, debido a que la naturaleza de la misma genera una redundancia de flujo de datos al punto de permitir interrupciones entre los canales de tal manera que estos se caen, y no se garantiza la consistencia y la disponibilidad de los mismos.
- El sistema no permite que se personalice el nombre de un punto de montaje debido a que están predeterminados.

3. PROBLEMA

A continuación, se encontrará el análisis del problema que se presenta en esta investigación, se podrá conocer la descripción, el planteamiento y las preguntas planteadas para la sistematización.

3.1. DESCRIPCIÓN DEL PROBLEMA

Las plataformas actuales de audio mixing no basan su tecnología en formato OGG de alta disponibilidad y bajo consumo de recursos, ya que utilizan servicios de un mayor uso de banda ancha, basado en formatos MP3 , aumentando los costos operacionales de la infraestructura, debido a que la transmisión de los paquetes se torna más pesada en función de su tamaño y generan un mayor tráfico en la red, esto incrementa la latencia y la pérdida de paquetes, afectando directamente la calidad del contenido recibido debido a que este no llega en su total integridad, además de crear cuellos de botella que entorpecen el rendimiento y la disponibilidad; dando paso a que el requerimiento de red de los diferentes tipos de nodos sea considerablemente más elevado. **(*Richard C. Dorf,* 2006).**

Los sistemas de streaming en vivo están limitados por una cantidad máxima de usuarios, junto con un modelo estático de servicio que no permite su elasticidad y esto genera la subutilización del servidor, debido a que la máquina dispone toda su capacidad y puede ser que esta no esté siendo utilizada por una mínima cantidad de usuarios, o que, por el contrario, no haya suficientes recursos físicos y se afecte la disponibilidad del servicio.

Los djs afrontan un problema de visibilidad debido a que en las plataformas actuales no tienen un cliente que use formatos ligeros, esto aumenta el costo de operación de estas y este costo es traducido a los usuarios DJs, viéndose afectados al momento de querer difundir su transmisión a los dispositivos de su audiencia, teniendo en cuenta que los sistemas operativos Android son los más utilizados en el mercado.

Al dificultar el acceso de los DJs a una plataforma económica de baja compresión para facilitar la interacción entre los mismos y sus audiencias, se configura un escenario negativo para el sector debido a que se crea una barrera digital que lo afecta negativamente, ya que, al limitar el alcance del contenido, se está evitando que llegue a oyentes que no estén presentes en el mismo lugar de la fuente.

3.2. PLANTEAMIENTO

¿Implementar un sistema de streaming distribuido basado en OGG, mediante una metodología en cascada para que los djs transmitan en vivo su audio mixing e interactúen con la audiencia, solucionará los problemas referentes al alcance, rendimiento, disponibilidad y calidad de audio?

3.3. SISTEMATIZACIÓN DEL PROBLEMA.

¿Implementar el sistema sobre una plataforma de Cloud Computing garantizará la disponibilidad y rendimiento del mismo?

¿Usar códec OGG Vorbis, garantizará una mejora en el rendimiento del sistema?

¿Utiliza un formato de baja compresión con pérdida de datos permite mantener la calidad del contenido?

¿Desarrollar un sistema de streaming provee mas alcance a un DJ?

¿Un sistema de streaming distribuido con cliente en Android, aumenta la masificación del servicio?

¿La implementación de un sistema de streaming interactivo, mejora la conexión con la audiencia?

4. OBJETIVOS

4.1. OBJETIVO GENERAL

Implementar un sistema de streaming distribuido basado en OGG en la nube, para audio mixing en vivo, realizado por DJs desde controladores con interfaces digitales; con acceso mediante una aplicación cliente Android 4.4.

4.2. OBJETIVOS ESPECÍFICOS

- Identificar los procesos, las redes sociales y las tecnologías actualmente utilizadas por DJs, para realizar audio mixing.

- Determinar los requerimientos del sistema de información, teniendo en cuenta los procesos de conexión de controladores DJ y tecnologías utilizadas por los DJs para audio mixing.

- Diseñar las interfaces gráficas y la arquitectura del sistema de streaming basado en los requerimientos obtenidos.

- Implementar el sistema de streaming y la aplicación cliente para Android 4.4, haciendo uso de la arquitectura y el diseño desarrollado.

5. JUSTIFICACIÓN

El sector del entretenimiento musical se mantiene en constante innovación, en función de diferentes tecnologías como: pantallas LED de gran tamaño y alta definición, audio envolvente de la mejor calidad, amplio alcance técnico-musical, de dispositivos complejos usados por artistas y nuevas formas de comunicación e interacción.

Uno de los principales actores, que hacen uso extendido y constante de estas tecnologías son los DJs, ya que, al extender sus alcances se aumenta la experiencia de su audiencia y al permitir la comunicación bidireccional, se logra que los oyentes comuniquen sus deseos respecto al género musical o un tema específico, de esta manera el público siente que es parte del evento y puede expresar sus deseos directamente al artista sin hablarle al oído o encontrarse en la misma localización de éste.

Los sistemas que facilitan la difusión remota en conjunto a herramientas de interacción, no solo mejoran la experiencia del usuario, también la de los DJs, debido a la naturaleza de esta, que permite transmitir simultáneamente a varios puntos del planeta, logrando adquirir audiencias remotas que también tengan contacto o retroalimentación mediante la aplicación con el DJ.

Es importante tener en cuenta que, este tipo de sistemas en entornos de producción comercial pueden facilitar el análisis de tendencias que determinarán las preferencias actuales de una audiencia específica y brindarán información que puede ser utilizada por DJs y clubes de entretenimiento nocturno para fidelizar y aumentar su público con el fin de incrementar sus ingresos.

Es necesario evaluar las necesidades que tienen estos sistemas al momento de implementarlos e implantarlos, debido a que diferentes factores pueden mejorar el desempeño de estos y puede facilitar la accesibilidad de los clientes. Un sistema que mantenga sus servidores en una distribución geográfica y que permita accederlos en función de la distancia, aumenta la calidad de transmisión, debido a que los paquetes transitan menos redes para llegar a su destino, y un acceso basado en el meridiano, puede aumentar la accesibilidad, ya que es más probable que una fuente coincida con el mismo huso horario o uno cercano, en el cual se localizan los oyentes que se conecten en la misma hora. Es decir, es más probable que un DJ que transmite a las 11:00 pm de la noche en Santiago de Chile tenga más oyentes de Bogotá que en Kiev, debido a que la distancia de los husos horarios es menor entre Chile y Colombia, y esto hace que más usuarios tengan disposición de estar conectados en estos dos países. El planteamiento anterior da pie para pensar en una solución en la nube, usando un servicio que implemente una red distribuida de servidores, por ejemplo, en Amazon WS con su servicio EC2.

Gracias a que se da la estructura tecnológica y se posee el conocimiento para construir una solución que permita afectar positivamente el sector de los DJs en Colombia, se va a desarrollar un prototipo de plataforma de streaming de alto rendimiento, porque es importante facilitar herramientas para que los DJs tengan más alcance de transmisión en función de aumentar su audiencia en diferentes partes del territorio nacional e inclusive del mundo, para así mejorar su reputación y aumentar su visibilidad, gracias a que puede llegar a más personas en el mismo tiempo que antes de usar la aplicación.

La facilidad que ofrece Amazon para aumentar o disminuir en segundos la capacidad de los recursos de cómputo, permite que el costo de mantenimiento esté en función del uso de la plataforma del cliente de los servicios web, este modelo permite que esta solución funcione en pro de los usuarios y no use recursos innecesarios, por tal motivo, desarrollar el proyecto brinda un modelo para plataformas de streaming que gestionen inteligentemente sus recursos en función de las necesidades sus clientes.

6. MARCOS DE REFERENCIA

En la siguiente sección se podrán conocer los marcos de referencia que contienen la contextualización teórica y conceptual del proyecto, de manera que el lector pueda conocer las teorías que serán consideradas para el desarrollo del mismo. Además de lo anterior se encontrará el estado del arte respecto al área en el cual se desarrolla el estudio.

6.1. MARCO TEÓRICO

En el siguiente marco, se encontrarán los fundamentos aplicados en el proyecto, estos demuestran que se poseen las teorías y las herramientas para desarrollar el mismo con las características que se han planteado.

6.1.1. Streaming.

El término de streaming es conocido en estos días, debido a que los medios digitales son utilizados para difundir masivamente información con diferentes formatos. El concepto se ha adoptado ampliamente para realizar diferentes tipos de distribuciones como son: descarga progresiva, web casting, video conferencias, y podcast de algún tipo de información. Estas posibilidades se dan por la amplia gama de tecnologías desarrolladas para este fin.

En el año 2016, esta tecnología cumple 22 años de haber impactado las organizaciones a nivel global para generar un cambio excepcional en la manera como se comunican estas internamente y externamente, con el fin de optimizar sus operaciones y comunicar eficientemente los nodos que afectan su operación.

Esta tecnología ha tenido su evolución desde las redes de comunicación telefónica, ya que la línea entre la necesidad de comunicar y los medios digitales ha ido disminuyendo cada vez más, hasta el punto de cohesionar para dar paso a un amplio número de posibilidades que entregan valor a la organización que la implementa. (**D. Rayburn, 2012**).

Antes de la aparición del streaming en abril de 1995, la reproducción de archivos multimedia a través de una red como internet, implicaba que el cliente descargara completamente el archivo al disco duro, teniendo en cuenta que los archivos de audio y especialmente los de video son compuestos por una gran cantidad de Bytes, esto hace que su descarga sea lenta y compleja, debido a que, si el archivo no se descarga completamente, no es posible para el cliente reproducir un segmento del contenido.

Desde que internet impactó en el desarrollo tecnológico en los noventa (90), este se convirtió en el medio principal para obtener contenido multimedia para el consumo humano.

Internet en general y la televisión, promueven actividades a través de una red en concreto, esto ha supuesto un hito en el desarrollo de la humanidad, donde cada vez es más fácil acceder a diferente información y canales de difusión con fines investigativos, de entretenimiento y comunicativos.

El streaming también denominado como: transmisión, lectura continua, difusión en flujo, lectura en tránsito, difusión continua, descarga continua o media flujo.

Este consiste en la distribución de paquetes digitales de datos multimedia mediante una red de computadores, con la finalidad de que el usuario consuma el producto de audio o video en paralelo, mientras el archivo se descarga en un buffer. El concepto de streaming hace referencia a un flujo continuo de información, que fluye sin interrupción.

Esta tecnología utiliza un buffer de almacenamiento, en el cual se van descargando en la máquina del usuario los paquetes que componen los ficheros multimedia, para paralelamente irlos reproduciendo. Este método, funciona de una manera muy diferente al mecanismo de descarga tradicional, que requiere que el usuario descargue por completo el archivo en su máquina local, para posteriormente reproducir el contenido. Para mitigar los efectos de retardo que se presentan en la transmisión, al iniciar esta, el servidor envía una corriente inicial de información que compone el comienzo del archivo, y así, evitar que la reproducción se corte por los efectos del retardo, que pueden tener algunos paquetes del medio que se desea almacenar en el buffer al momento de llegar al nodo servidor.

Una de las principales restricciones al momento de trabajar con multimedia en una red es el ancho de banda que se requiere para transmitir la información, generalmente para solucionar esta restricción y optimizar el uso de esta, se cuenta con que el cliente y el servidor utilice su máxima capacidad de transmisión de datos para realizar el proceso de descarga de archivo, de esta manera la obtención de los recursos se ajusta a la capacidad de transmisión de la red de las máquinas.

El streaming moderno, hace uso de protocolos RTP (real-time transfer protocol) que transmite la información multimedia en tiempo real. Estos protocolos no utilizan el máximo ancho de banda disponible para la descarga y la visualización del medio, ya que utiliza únicamente los recursos computacionales necesarios para este procedimiento que permite la reproducción en tiempo real de llegada de los datos.

Posteriormente a experimentar el uso de la televisión para la trasmisión de información escogida por el proveedor de esta, se denota una falencia en la cual el usuario no puede escoger el contenido que desea consumir, debido a esto el streaming se constituye en una opción para suplir esta necesidad y permitir al usuario identificar y escoger el contenido que desea consumir.

Dado el paradigma anterior que le restainteraccion al usuario, el streaming bajo demanda se constituye como una opción viable para permitir al cliente tener control y libertad de escoger toda la información que desea. **(M. Pietro, 2007).**

En base al tipo de demanda que tenga el usuario, el streaming se divide en dos categorías:

En directo, este tipo de streaming consiste en permitir que la audiencia consuma el contenido en tiempo real, es decir que acceda al canal de transmisión al tiempo que la fuente realiza la difusión, Unos ejemplos pueden ser la transmisión de conciertos en tiempo real a través de internet o de un programa radial o televisivo que está siendo realizado en vivo y en directo. A este tipo de difusión se puede identificar como Broadcast, porque la transmisión en vivo se realiza a todos los usuarios con la misma información. De esta manera cuando un cliente se conecta a un punto de montaje específico del servidor, recibe en un tiempo determinado los mismos datos que reciben los clientes conectados a este mismo canal (excepto por las condiciones que generan retardo en las redes).

Para realizar este tipo de transmisión no solo se depende del servidor de streaming, es necesario contar con una fuente que capture, codifique y envíe la información que pueda ser entendida por el servidor, para ser posteriormente entregada, este nodo difusor es conocido como Broadcast. La fuente puede estar localizada en la misma máquina del servidor, si este no cuenta con muchos clientes, pero en un entorno profesional estos dos están separados por naturaleza para otorgar un servicio eficiente.

El otro tipo de streaming se conoce como," stream multimedia bajo demanda", a diferencia del otro, los medios que reproducen los usuarios están almacenados en el servidor o una máquina dedicada a mantener los ficheros. El usuario es libre de escoger el contenido o visualizar una lista de reproducción ya definida con anterioridad, sea por el mismo cliente o por el servidor. El cliente que se conecta a este tipo de montaje, no va a recibir los mismos paquetes de datos que otros clientes, ya que puede escoger el contenido que desea consumir.

Para difundir los paquetes a los clientes, se pueden usar dos modos, multicast o unicast.

En modo multicast, el servidor envía solo un stream que se comparte entre los diferentes clientes, de manera que este envía la información solo una vez y esta es distribuida a todos los usuarios. Esta técnica es recomendada para transmisiones en vivo, debido a que todas las conexiones al punto de montaje, esperan los mismos paquetes de datos en un mismo momento. Esta técnica evita congestiones en la red, sin embargo, implementarla es una tarea compleja ya que se requiere acceso a una troncal con soporte multicast.

En modo unicast, cada cliente inicia su propio stream de forma que se inicia una conexión uno-a-uno con el servidor, esto quiere decir que cada cliente consume la misma banda ancha que requiere la difusión, y el servidor utiliza la banda ancha de la difusión multiplicado por el número de clientes. Esta técnica requiere que el servidor tenga una gran cantidad de banda ancha, mucho mayor que la de los clientes. Esta es la más utilizada actualmente debido a que se basa en portocolo IP y la mayoría de proveedores de ISP no cuentan con soporte para multicast. **(M. Pietro, 2007).**

6.1.2. Icecast y ogg vorbis.

La implementación de un software para la realización de audio mixing en vivo puede ser una tarea compleja, que puede tardar una considerable cantidad de tiempo, debido a que es necesario estructurar varias piezas de código, que hagan la gestión de un buffer alimentado por diversas fuentes. Por fortuna el software tipo Open Source brinda una herramienta de streaming robusta que maneja diversos formatos de audio y video para realizar trasmisión en vivo y bajo demanda, esta herramienta es Icecast, un software robusto y escalable que actualmente se encuentra en su versión 2.5 Beta 1 y es libre mediante licenciamiento GNU GPL v2.

Los desarrolladores de Icecast lo definen de la siguiente manera:

"Icecast es un servidor de streaming de medios que actualmente soporta Ogg (Vorbis y Theora), Opus, WebM y audio MP3.

Se puede utilizar para crear una estación de radio por Internet o una máquina de discos corriendo en privado y muchas cosas en el medio. Es muy versátil al añadir nuevos formatos con relativa facilidad, además de ser compatible con los estándares abiertos para la comunicación y la interacción."[1]

La fundación xiph.org, es una corporación dedicada a proteger las fundaciones de multimedia en internet, de interés privados. Tiene como propósito soportar y desarrollar estándares y conjuntos de software libre como Icecast 2. *(**xiph.com**, **2015**).*

Este servidor es soportado en Linux (recomendado en RedHat y Debían), FreeBDS, OpenBDS, Solaris y las distribuciones de Windows 2000 o superior.

Icecast es un software con una experta corporación y una activa comunidad de desarrollo y soporte que ha permitido, que este llegue a un punto en el cual ofrece una serie de herramientas robustas y muy útiles para los desarrolladores y aficionados que deseen explotar su utilidad.

[1] ***Icecast***, Acerca de nosotros, Internet, Disponible en (http://icecast.org/). [Consultado el 18 de mayo del 2015]

Comprender su funcionamiento básico e implantar una copia en una máquina es bastante sencillo y puede realizarse en cuestión de horas e inclusive minutos.

Esta herramienta permite realizar streaming en vivo, con diferentes tipos de fuentes como Winamp, Traktor DJ, Virtual DJ, Music Player Daemon, Muse, VLC y etc.***(Icecast.com, 2015)***. Las fuentes deben tener implementado un protocolo RTP con soporte a codificación OGG, debido a que este formato será el utilizado en el sistema.

Icecast 2 ofrecer paneles de administración y modulos públicos, que les permite a los usuarios conocer los estados de las transmisiones junto a sus metadatos, y así poder obtener más información de los medios que se están almacenando temporalmente y en tiempo real en el servidor. Mediante el panel de administrador, se pueden conocer todos los metadatos de los Medios que están en difusión. Por ejemplo, si se realiza un Stream desde Traktor Dj con un bitrate de 64 bps y una frecuencia de muestreo de 44000hz, como administrador se podrá conocer esta información mediante una interfaz gráfica que trae Icecast por defecto o un Servicio Web que entrega la información en formato XML o JSON dependiendo de la necesidad del administrador. A parte de la configuración técnica del Stream, el administrador y los usuarios públicos podrán acceder a datos como el título del tema musical, el género del canal, el número de oyentes conectados en tiempo real, el nombre del punto de montaje y otros con menor importancia.***(Icecast.com, 2015).***

Icecast permite configurar diferentes puntos de montaje personalizados con formatos OGG o MP3, que pueden ser accedidos con una contraseña por defecto o con una contraseña pre configurada para el punto de montaje específico, esto permite que el administrador o la fuente realice una configuración que considere más óptima para su necesidad.

Este servidor, además de proveer el núcleo para la transmisión de medios, también provee un recurso muy importante, que es la facilidad de conectar dos servidores remotamente con el fin de compartir entre ellos los puntos de montaje que cada uno está gestionando, al realizar la configuración de red un (1) servidor puede ser esclavo, el otro maestro. Los servidores no tienen límites para definir el número de conexiones que puede hacer con otros servidores con el rol de maestros o esclavos y pueden cumplir los dos roles a la vez.

Como anteriormente se mencionó, Icecast 2 soporta una amplia gama de formatos, pero el más importante para el sistema planteado en el proyecto es OGG Vorbis. Este es un nuevo formato de compresión, comparado en calidad con otros formatos como MP3, AAC y otras compresiones de audio. OGG es el nombre del formato contenedor para audio, video y metadatos.

Vorbis en cambio es el nombre del esquema de compresión de audio, diseñado para ser contendió en OGG. El mimetype application/ogg fue aprobado en febrero de 2003.

Este formato fue diseñado con el fin de reemplazar los formatos propietarios por uno libre con una calidad similar a los más utilizados.

Respecto a Vorbis es necesario saber que: este puede comprimir pequeños ficheros que disminuyen el costo de banda ancha; gracias a este formato, se estandarizó los comentarios de cabecera que proveen un espacio para distribuir notas con la música. Si se usa otro formato como MP3 se está obligado a pagar un porcentaje de las ventas, debido a que usan patentes, en cambio Vorbis tiene una licencia libre que determina que, al ser usado, no se debe realizar ningún pago**(*vorbis.com, 2015*)**.

Este formato es el que permite llevar los metadatos a la aplicación cliente para que esta conozca cómo debe realizar la reproducción del medio.

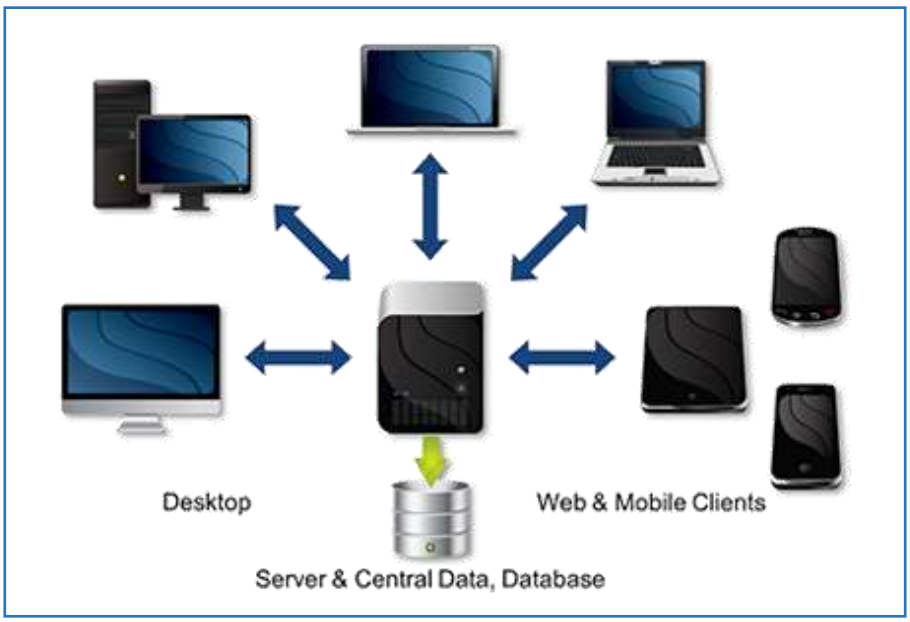
Gracias a lo mencionado anteriormente se da por hecho que se cuenta con el núcleo para realizar audio mixing en vivo y tener uno (1) de los componentes necesarios para cumplir con el objetivo del proyecto, pero es importante entender que Icecast 2 solo proveerá el núcleo de la transmisión, es decir, se utilizará para gestionar el streaming, pero este no será el encargado de gestionar la distribución de los oyentes y la gestión del servicio virtual, ya que este lo realizará un servicio web especialmente diseñado para esto.

6.1.3. Arquitectura cliente-servidor.

Teniendo en cuenta que el servicio de streaming debe estar en constante actividad y a la escucha de nuevas conexiones de aplicaciones clientes, se intuye que la arquitectura de la aplicación es cliente-servidor.

Se denomina servidor a la máquina que brinda servicios a solicitudes de muchos otros hosts que son los clientes. Los host clientes pueden estar activos o en constante intermitencia, y aun así el sistema continúa funcionando si un cliente se desconecta de los mismos.

Figura 1. Diagrama cliente-servidor



Fuente: tomada de dineroltd.com, 2015.

De esta arquitectura se puede ver un ejemplo clásico que es, el servidor web, que siempre está activo, recibiendo peticiones de los navegadores ejecutados en la máquina cliente.

Los servidores como tal reciben una petición de un cliente y estos responden con el objeto solicitado. En la arquitectura cliente-servidor los clientes no deben estar conectados entre sí para que el sistema sea funcional, otra característica principal de esta arquitectura es que el servidor debe tener una dirección fija y conocida para las aplicaciones clientes. La mejor manera de hacer visible el servicio es mediante un IP pública, que será mencionada más adelante. Entre las aplicaciones más conocidas que utilizan esta arquitectura se encuentran: las aplicaciones web, los servidores FTP, Telnet, de correo electrónico y streaming. **(J. Kurose and K. Ross, 2010).**

Normalmente en un entorno de trabajo exigente, un servidor no puede atender todas las peticiones, ya que si recibe un considerable número de estas, puede sobrepasar la capacidad de sus recursos y este será desbordado rápidamente. Por esta razón se suele usar una agrupación de host, para crear un servidor virtual de gran capacidad. Los aplicativos cliente-servidor a menudo precisan de infraestructuras robustas, ya que requieren que el servicio se mantenga disponible en todo momento.

Los proveedores de servicios basados en esta arquitectura deben afrontar costos de interconexión y ancho de banda para recibir datos de entrada y enviar datos de salida mediante internet.

Este tipo de servicios robustos como motores de búsqueda, comercio electrónico y redes sociales precisan de una infraestructura de alto costo e intensivo soporte.

Una aplicación de red, consta de parejas de procesos que intercambian mensajes entre sí. Por ejemplo, en un aplicativo web el navegador intercambio mensaje con el servidor para peticionar los elementos requeridos. En términos de redes se puede ver el servidor como un nodo computacional, pero en términos de una aplicación se ve como un proceso. No obstante, en la intercomunicación entre una pareja de procesos, se podrá etiquetar a uno como como cliente y al otro como servidor. En el contexto de una sesión de comunicación entre una pareja de procesos, el que inicia la comunicación se determina como el cliente y el que espera y responde la petición es el servidor.

Esta arquitectura es históricamente la más importante y aun es la más utilizada, ya que los servidores web y la mayoría del resto de los servicios de internet son clientes del servicio DNS, que traduce nombres de dominio en internet o direcciones de red.

Como ya se había mencionado antes, los motores de búsqueda utilizan esta arquitectura para su finalidad, que consiste en permitir a los usuarios buscar resúmenes de páginas que desean visitar.

Esto resúmenes están realizados por programas llamados escaladores web, que se ejecutan en segundo plano en el sitio del buscador para acceder a otros servidores web mediante internet. Por tanto, una máquina que contenga el algoritmo buscador, es tanto cliente como servidor, que responde a las peticiones de los usuarios y a la vez conecta a otros servidores para hacer peticiones.

En el ejemplo mencionado las tareas del servidor y las de los escaladores son independientes las unas de las otras, no es necesario una sincronización estricta y pueden ejecutarse concurrentemente. De hecho, al momento en que se realiza una búsqueda, se dispara un hilo de ejecución concurrente a los demás, unos procesando solicitudes y otros ejecutando escaladores. **(G. Coulouris, J. Dollimore, T. Kindberg and J. B. P. Junquera, 2001)**

Como se ha venido mencionando constantemente, la mayoría de aplicaciones utiliza parejas de procesos de comunicación que se intercomunican entre sí, cualquier mensaje enviado de una red a otro debe atravesar una red subyacente, un proceso envía un mensaje a la red y esta lo recoge a través de una interfaz de software conocida como socket.

Cuando el proceso desea enviar el mensaje lo hace a través del socket, este emisor supone la existencia de una red al otro lado de la puerta o socket que llevar el mensaje hasta el otro lado de la misma para entrégalo a otro proceso. **(J. Kurose and K. Ross, 2010).**

En la arquitectura de la aplicación de streaming en vivo se puededeterminar que, son 3 nodos computacionales los que se requieren para el correcto funcionamiento de un canal de streaming o stream, los nodos requeridos son: el servidor, la fuente y el oyente, en teoría la red fundamental se basa entre la comunicación de la fuente o Broadcast con el servidor, debido a que la comunicación entre estos es la base para la existencia de un punto de montaje, puesto que al haber una fuente de difusión y un servidor de captura, almacenamiento y distribución de paquetes, ya cuenta con el medio en disponibilidad, pero sin una aplicación cliente para los oyentes no podría funcionar coherentemente el sistema. Independiente del requerimiento de tres (3) tipos de nodos para el funcionamiento, las fuentes y los clientes no deben estar sincronizados, debido a que la fuente envía los paquetes al servidor y este a su vez envía estos a los clientes conectados. Evidentemente existen dos comunicaciones cliente servidor, totalmente independientes en su integridad la una dela otra, pero la disponibilidad de los medios está basada en el correcto funcionamiento de los tres (3) nodos involucrados en el proceso de transmisión.

6.1.4. Ip protocolo de internet.

Como ya se había mencionado, los servidores serán visibles gracias a un conjunto de IPs públicas para que de esta manera puedan ser accedidos por los clientes mediante internet.

El reenvío como el direccionamiento en internet son componentes importantes del protocolo IP, actualmente existen dos versiones utilizadas que son: la IPv4 y la IPv6 que es el propuesto para reemplazar a la primera que es el más utilizado actualmente.

Antes de ingresar al tema de lleno, hay que tener en cuenta los componentes que forman la capa de red en Internet. Son 3 capas que componen internet y son: El protocolo IP; el componente de enrutamiento, que determina la ruta que sigue un datagrama desde un origen hasta un destino; y el último componente que permite detectar los errores en los datagramas y responder solicitudes de la capa de red.

Los datagramas son los paquetes de red, estos desempeñan un papel importante en internet y están compuestos por:

- **La longitud de la cabecera**, contiene 4 bits que sirven para determinar donde comienza los datos del datagrama IP.

- **Tipo de servicio**, se incluyen en la cabecera con el fin de poder diferenciar entre diferentes tipos de datagramas, puede resultar muy útil identificar datagramas en tiempo real.

- **Longitud del datagrama**, es la longitud total del datagrama en bytes que tiene una longitud de 16 bits, el tamaño máximo teórico es de 65.535.

- **Identificador, indicadores y desplazamiento de fragmentación**. Estos tres están implicados en la fragmentación IP.

- **Tiempo de vida**, se incluye para garantizar que los datagramas no estarán eternamente en circulación a través de la red, este campo decremento en una unidad cada vez que el router procesa un datagrama, si el campo TTL llega a 0, el datagrama es descartado.

- **Protocolo**, Se emplea únicamente cuando un datagrama IP alcanza su destino, el valor de este campo indica el protocolo específico de la capa de transporte al que se pasaran los datos contenidos.

- **Suma de comprobación de cabecera**, ayuda a los routers a detectar errores de bit en un datagrama IP recibido, esta suma de comprobación se calcula, tratando cada 2 parejas de bytes de la cabecera como un número y sumándolos con complemento a 1.

- **Dirección IP de origen y destino**, cuando en el origen se crea un datagrama, se inserta su IP y la de destino en este campo.

- **Opciones**, permite ampliar la cabecera IP.

- **Datos**, este es el último campo y es el más importante, ya que este contiene la información que va a ser transferida por la capa de transporte.**(J. Kurose and K. Ross, 2010).**

Un datagrama contiene una cabecera de 20 bytes, si el datagrama transporta un segmento TCP, entonces el datagrama transporta un total de 40 bytes.

No todos los protocolos de la capa de enlace pueden transportar datagramas del mismo tamaño, algunos pueden transportar unos grandes mientras que otros unos datagramas más pequeños, Por ejemplo, las tramas Ethernet pueden transportar un máximo de 1.500 bytes de datos, mientras que las tramas para otros enlaces de área extensa no pueden transportar más de 576 bytes. La cantidad máxima de datos que puede contener una trama es conocida como, unidad máxima de transmisión MTU.

Debido a que cada datagrama IP se encapsula en una trama de la capa de enlace para ir de un router al siguiente, la unidad MTU impone un límite a la longitud del datagrama. El problema aquí es que cada uno de los enlaces existentes en la ruta entre el emisor y el destino, pueden utilizar diferentes protocolos de la capa de enlace y cada uno de estos puede utilizar un valor MTU diferente.

Antes de comenzar a mencionar el tema de la IPv4, se va a hablar de los routers y los host conectados en red. Un host dispone de un único enlace hacia la red, cuando la IP del host desea enviar un datagrama, lo hace a través de este enlace. La conexión entre el host y el enlace físico, se le conoce como interfaz. Debido a que la labor de un router consiste en recibir un datagrama por enlace y reenviarlo a otro enlace, quiere decir que el router está conectado a uno o más enlaces, debido a que todos los host y todos los routers son capaces de enviar y recibir datagramas IP, el protocolo IP requiere que cada host y cada router tengan asignada su propia IP, por ende una IP está asociada a una interfaz.

Las direcciones IPv4 tienen una longitud de 32 bits por lo que existen un total de 2^32 o 4000 millones de direcciones posibles. Estas direcciones se expresan usando la notación decimal separada por un punto del resto de los bytes de la dirección.

Cada una de las interfaces de un host o de un router de internet tiene que tener asociada una dirección IP que a nivel global es única, debido a esto las direcciones IP no se puede asignar aleatoriamente. Una parte de la IP estará definida por la subred a la que está conectada.

La estrategia de asignaciones de dominios en internet se conoce como enrutamiento entre dominios sin clase, igual como pasa con las subredes IP, la dirección de 32 bits se divide en dos, y se puede ver la notación decimal de la IP usando punto como a.b.c.d/x. Los x bits de una dirección en el formato de IP, indica la red de la dirección.

Por lo general una organización tiene un rango de direcciones asignadas. Las direcciones de los dispositivos de la organización compartirán los mismos prefijos.

Para obtener un rango de direcciones que puedan ser utilizadas en una subred de una organización, el administrador de esta debe contactarse con el proveedor de internet, para que le proporcionará direcciones extraídas de otro bloque superior.

Después de que la organización ha obtenido el rango de direcciones IP, puede asignarlas a las interfaces de sus host y routers, normalmente un administrador asigna estas manualmente al router.

Las direcciones del host, también se pueden asignar manualmente, pero actualmente se utiliza el protocolo de configuración dinámica de host DHCP, que permite a un host obtener la dirección IP automáticamente al conectarse a la red.

Teniendo en cuenta lo mencionado en la anterior sección del protocolo IP, es necesario mencionar que el sistema de streaming estará basado en este, ya que mediante IPv4 se les asignará una dirección IP pública a los servidores que compondrán el servicio.

EL sistema de streaming brindará un servicio virtual, ya que estará compuesto por varios servidores que serán visibles a nivel mundial gracias a sus IPs públicas, estos servidores serán administrados por un algoritmo que permitirá dividir eficientemente las fuentes en los diferentes servidores y recuperar los puntos de montaje para entregarlos a las aplicaciones clientes. **(J. Kurose and K. Ross, 2010).**

6.1.5. Sistemas distribuidos.

Se conoce como sistema distribuido, a aquel sistema que tiene sus componentes localizados en diversos computadores conectados en red, estos comunican y coordinan sus operaciones mediante el envío de mensajes. Esta definición expone varias características que son: carencia de un reloj global y fallos ajenos a los componentes.

Tres (3) ejemplos de sistemas distribuidos que funcionan en la actualidad son:

- Internet
- Una intranet
- Computación móvil y ubicua

La distribución eficiente de recursos es el objetivo principal para construir un sistema distribuido. Los recursos pueden ser utilizados por servidores o clientes, o pueden ser encapsulados como objetos y ser accedidos por otros objetos.

Los principales desafíos que se encontraran al construir un sistema distribuido son: la heterogeneidad de sus componentes, su característica de ser abierto, que permite sustituir o añadir componentes, la seguridad y la escalabilidad, que es la funcionalidad que posee al momento de aumentar el número de usuarios, tolerancia a fallos, concurrencia de componentes y la transparencia que experimentan los usuarios.

Los computadores que están conectados en una red pueden están separados por diferentes tipos de distancias ya sea por, continentes, ciudades o en localizaciones diferentes en un mismo edificio. La definición de sistemas distribuidos tiene las siguientes características:

- **Concurrencia**: la ejecución de programas concurrentemente es una norma en sistemas distribuidos, debido a que un usuario puede realizar un trabajo en el computador de él, mientras otro usuario realiza otra tarea en otro computador, pero a la vez estos dos pueden compartir recursos o servicios como un documento HTML o un fichero. La capacidad del sistema para manejar diversos recursos compartidos está en función de añadir más recursos. *"La coordinación de diferentes programas que comparten recursos y se ejecutan de forma concurrente es también un tema importante y recurrente"*[2]

- **Inexistencia del reloj global**: cuando los recursos necesitan interactuar para realizar operaciones, deben coordinar sus acciones mediante el envío de mensajes. Resulta que hay límites al momento de sincronizar dos máquinas debido a que no existe solo una noción del tiempo exacta a nivel global. Esto es consecuencia de la comunicación basado en mensajes enviados por red que pueden ocasionar problemas de temporización.

- **Fallos independientes**: ningún sistema informático esta excepto de errores y todos pueden fallar, los diseñadores de sistemas tienen la responsabilidad de planificar y manejar las consecuencias de los posibles fallos. Los sistemas distribuidos son vulnerables a fallos diferentes a los sistemas centralizados, estos fallos pueden hacer que los componentes queden aislados en la red sin detener su ejecución, de hecho, los procesos que se ejecutan en estos puede que no detecten su desconexión de la red o el desempeño de esta. De igual manera al momento de darse lugar a un fallo, este no será inmediatamente detectado por los otros componentes del sistema con los que se debe comunicar, de este modo cada componente puede fallar independientemente sin afectar a los otros.

(G. Coulouris, J. Dollimore, T. Kindberg and J. B. P. Junquera, 2001).

El motivo para construir un sistema distribuido consiste en compartir recursos, este término de recurso es ligeramente abstracto pero encaja perfectamente para definir un rango de elementos que pueden ser compartidos de manera óptima por las máquinas de una red.

Estos elementos se extienden desde componentes de hardware, como: discos duros e impresoras, o componentes de software como ficheros o base de datos. En nuestro caso el recurso distribuido que será determinante para el proyecto es el servicio virtual de streaming.

[2]**G. Coulouris, J. Dollimore, T. Kindberg and J. B. P. Junquera**, Sistemas distribuidos: conceptos y diseño, Pearson Educación, 2001, 726 páginas, ISBN: 9788478290499

Los sistemas distribuidos deben tener la capacidad de operar eficientemente a escalas diferentes, desde pequeñas a enormes implementaciones. Un sistema distribuido mantiene su eficiencia cuando ocurre un aumento en los recursos o cantidad de usuarios. Internet es un claro ejemplo, debido a que la cantidad de usuarios que constantemente se desconectan y conectan, haciendo que se incremente dramáticamente su uso.

El diseño de un sistema distribuido escalable presenta los siguientes retos:

- **Control del costo de los recursos físicos**: dependiendo de la demanda de un recurso, debería ser posible extender su capacidad en el sistema. Para que un sistema con n recursos sea escalable, este debe ser O(n), es decir que sea proporcional al número de usuarios n. Por ejemplo si un servidor de ficheros permite que 20 usuarios estén conectados simultáneamente, 2 servidores permitirían 40 usuarios, en teoría es simple pero en la práctica es mucho más complejo.

- **Control de pérdidas de prestaciones**: considere que administra un conjunto de datos que es proporcional al número de usuarios del sistema. Los algoritmos que utilizan estructuras jerárquicas se comportan mejor que uno que emplee estructuras lineales, aun así las estructuras jerárquicas permiten pérdidas en las prestaciones, ya que el tiempo que lleva acceder a datos estructurados jerárquicamente es O (log n) donde n es igual al tamaño del conjunto de datos.

- **Prevención del desbordamiento de recursos de software**: un ejemplo claro de pérdida de escalabilidad lo se puede ver en el protocolo usado para las direcciones de internet que emplea 32 bits, debido a esto se calcula un desbordamiento del sistema, ya que las direcciones se acabaran. Para ser objetivos, es difícil predecir la demanda que tendrá un sistema con años de anticipación y sobredimensionar las predicciones puede tener una consecuencia peor que la adaptación al cambio.

- **Evitación de cuellos de botella de prestaciones**: para evitar cuellos de botella, los algoritmos deben ser descentralizados.

  Para tener una mejor percepción se utilizará como ejemplo el sistema de nombres de dominio, el cual contiene la tabla de nombres de dominio en un solo fichero, que puede ser descargado en cualquier computador, esto es funcional si solo existieran pocos computadores conectados a internet, pero al incrementar estos, el servicio experimento cuellos de botella de prestaciones y administración.

Para eliminar este problema, las tablas de nombres de dominio se realizaron diferentes particiones entre diferentes servidores de internet para mejorar la prestación del servicio.

Un sistema escalable con un comportamiento ideal, se da cuando el sistema aumenta su escala operativa sin que la aplicación se vea afectada, pero esto es difícil de conseguir, debido a que requieren técnicas complejas de implementar.***(G. Coulouris, J. Dollimore, T. Kindberg and J. B. P. Junquera, 2001).***

Los servidores que componen el núcleo del servicio de streaming estarán distribuidos a nivel global, haciendo uso de los servicios Web de Amazon EC2, se aumentará la disponibilidad y mejorará la prestación del servicio.

Al utilizar la nube, se provee acceso a recursos en los diferentes continentes, esto permitirá que se haga un servicio virtual distribuido que ofrezca escalabilidad en función del número de usuarios conectados. Los servidores de streaming serán distribuidos en diferentes partes del planeta en función del continente. Esto permitirá que los usuarios DJ realicen su difusión al servidor más cercano y los oyentes se conecten al servidor que esté localizado a una distancia espacial menor, para poder recibir óptimamente el flujo de datos. Un algoritmo central será el encargado de conocer la geolocalización de los usuarios para facilitarles la conexión al servidor indicado, de esta manera se optimiza el uso del servicio virtual de streaming para aumentar la disponibilidad.

6.1.6. Amazon ec2.

Amazon Elastic Compute Cloud es un servicio prestado por Amazon, para proporcionar recursos informáticos que sean modificables de una manera rápida. Esta diseña para permitir que los desarrolladores accedan a una capacidad informática escalable mediante interfaces WEB.

La intuitiva interfaz de servicios EC2 de Amazon facilita la configuración de la capacidad deseada con una pérdida mínima de prestaciones, a la vez que provee un completo control sobre los recursos informáticos utilizados. El servicio provee mecanismos que reducen el tiempo que se requiere para iniciar nuevas instancias de servidores en minutos, de esta manera se puede escalar eficientemente la capacidad de un recurso, ya sea aumentado o reducido, en función de la necesidad del sistema.

El modelo utilizado por Amazon, permite pagar únicamente por la capacidad que se está utilizando, a la vez que provee tolerancia a fallos para crear aplicaciones resistentes a errores y no incurrir en lo errores de implementaciones comunes.

El servicio EC2, permite escalar en minutos, de manera que se puede iniciar miles de instancias de servidores simultáneamente, esta escalabilidad se puede realizar mediante la interfaz WEB, dependiendo de los requerimientos del cliente del servicio EC2.

El cliente de Amazon podrá tener un control total de todas sus instancias de manera que tendrá acceso a la raíz de todas ellas, y podrá interactuar con estas como si fueran máquinas físicas. También se permite detener las instancias y mantener su estado para posteriormente recuperarlo al momento de reiniciarlas mediante la API del servicio WEB. Las instancias pueden iniciarse con diferentes sistemas operativos, ser cargadas en el entorno de la aplicación, gestionar los privilegios de estas y permitirles acceso a la red.

Para utilizar Amazon EC2 se necesita:

- Seleccionar una imagen de la máquina virtual de Amazon que se desee, esta puede ser una máquina con la instalación básica del sistema operativo que puede ser Linux O Windows. También puede ser una plantilla personalizada o una perteneciente a un conjunto de plantillas predeterminadas que incluyen diferentes servicios por defecto, que pueden ser personalizados y utilizados dependiendo de la necesidad del sistema.

- Después de iniciar la instancia, se debe proceder a configurar la seguridad y el acceso a la red.

- Iniciar y supervisar las instancias iniciadas para acceder a las herramientas de gestión proporcionadas.

- Determinar si se desea que la máquina se ejecute en una o más locaciones, definir IPs estáticas y adjuntar otros bloques que requiera la aplicación.

Por último pagar únicamente por los recursos que realmente se consuma, como las horas de uso de las instancias o la trasferencia de datos.

Este servicio permite iniciar instancias en diferentes ubicaciones. Estas ubicaciones de Amazon son compuestas por regiones y Zonas. Las Zonas de disponibilidad según Amazon son: *"regiones diferentes que están diseñadas para estar aisladas de fallos que se produzcan en otras Zonas de disponibilidad, y que proporcionan conectividad de red de baja latencia a otras Zonas de disponibilidad de la misma Región "*[3]

[3] ***Amazon,*** Details, Internet, Disponible en (http://aws.amazon.com/es/ec2/details/). [Consultado el 20 de abril del 2015]

Al iniciar las instancias en diferentes Zonas, las aplicaciones serán protegidas, en el caso que en una ubicación suceda un error. ***(aws.amazon.com, 2015).***

Las regiones, están compuestas por una o más zonas y están dispersas en diferentes puntos geográficos de los diferentes continentes. Amazon garantiza el 99,5% de disponibilidad es todas las Regiones.

Las regiones y zonas que las componen, donde opera el servicio EC2 son:

- **América**
  - Norte de Virginia
  - Oregón
  - Norte de California
  - Sao Pablo
  - GovCloud

- **Europa/Oriente Medio/África**
  - Irlanda
  - Frankfurt

- **Asia Pacífico**
  - Singapur
  - Tokio
  - Sidney
  - Pekin

Amazon usa un modelo de IPs conocidas como, IPs elásticas. Estas IPs no están directamente relacionadas con una instancia, sino que están asignadas a una cuenta, lo cual permite asignarlas dinámicamente a las instancias para disimular errores en alguna de las zonas, de manera que si una instancia se cae en una Zona, automáticamente la IP pública puede ser reasignada a otra instancia en otra Zona, y así mantener la disponibilidad de los servicios de nuestra aplicación.

Este servicio cuenta con una herramienta de supervisión llamada CloudWatch. Esta tiene una interfaz web que muestra el nivel de utilización de recursos, la funcionalidad operativa de las instancias y los patrones generales de demanda, como también las métricas de CPU, Disco y tráfico de Red. Mediante una interfaz gráfica se puede conocer los estados de las métricas expresados estadísticamente en gráficos y valores, también se pueden programar alarmas en función de los valores de las métricas. Para utilizar la herramienta de CloudWatch, simplemente se debe seleccionar las instancias que se quiere monitorear y este iniciara inmediatamente a registrar los valores.

Otra funcionalidad muy importante que facilita Amazon es la de Auto Scaling, que provee al usuario de capacidad automatizada de escalabilidad.

En el servicio se puede definir diferentes condiciones que, al ser cumplidas, automáticamente se dispararán acciones anteriormente especificadas, para aumentar o disminuir la capacidad de cómputo en función de las métricas presentadas. De esta manera si se detecta un pico en la demanda, las instancias se crearán automáticamente, pero si la demanda disminuye, estas también disminuirán proporcionalmente.

Una herramienta de Auto Escalabilidad es muy importante en un sistema distribuido de streaming ya que la demanda de este, puede variar considerablemente en función de la hora y día de la semana en alguna región.

Amazon proporciona una lista de instancias que se pueden iniciar. Estas, están optimizadas para adaptarse a nuestra necesidad, ya que son una serie de combinaciones de recursos como CPU, memoria, almacenamiento y red. También cuentan con un tamaño de escalabilidad límite, pero aun así se pueden iniciar otras de este mismo tipo si se requiere más capacidad.

Los tipos de instancias son:

- T2
- M3
- C4
- C3
- R3
- G2
- i2
- D2

Debido a que el prototipo que resultara del proyecto requiere muy pocos recursos para realizar las pruebas de, se utilizara y mencionara únicamente la instancia T2. Esta funciona con rendimiento CPU basado en ráfagas. Estas instancias son una buena opción para cargas de trabajo que no usan CPU por completo, es decir que de vez en cuento se les hacen solicitudes como un servidor web, una pequeña base de datos o un entorno para desarrolladores.

Para poder monitorizar los recursos utilizados por las instancias de AWS, se puede contar con la herramienta de análisis de metricas denominada CloudWatch, con la cual se puede recopilar y realizar el seguimiento de métricas y logs para conocer el rendimiento de las máquinas y establecer alarmas o configurar la infraestructura para que cambie en función del uso de los recursos. Este servicio permite monitorizar servicios de EC2, dynamoDB y Amazon RDS, además de permitir configurar métrica personalizadas de los servicios y aplicaciones desplegadas.***(aws.amazon.com, 2015).***

6.1.7. Servicios web.

Para recuperar los puntos de montaje de los servidores y sincronizar la distribución de estos mediante un formato JSON a los clientes, además de automatizar el proceso de distribución de recursos entre estos, se utilizará un servicio WEB. Este para permitirá que la plataforma esté equilibrada en términos de rendimiento y disponibilidad debido a que no permitirá que alguno de los servidores reciba más conexiones de las que puede soportar. Al implementar este sistema orientado a servicios WEB se va a poder acceder desde la mayor cantidad de plataformas móviles y de escritorio. Este servicio será conocido como, el algoritmo central de administración inteligente de recursos.

Los servicios WEB nacen de la necesidad de estandarizar un canal para la comunicación entre diferentes plataformas. Hasta la actualidad se han realizado varios intentos en la creación de un estándar que facilite la comunicación, como CORBA o JAVA RMI, pero no tuvieron éxito en su intento debido a que presentaban muchos inconvenientes para comunicarse con aplicaciones de alto nivel hechas en diferentes plataformas.

El desarrollo y amplia adopción de la WWW trajo consigo una tecnología para permitir que las computadoras o dispositivos clientes de las organizaciones, accedieran a servicios remotos fuera de sus redes locales.

Las organizaciones tenían una opción clara y viable para permitir convertir su información en formato HTML, que podía ser accedida mediante un navegador web para tener acceso directo a un conjunto de información. La utilización de este tipo de almacenamiento conlleva problemas, ya que no se puede acceder fácilmente a varios catálogos de datos. Para dar solución a este, surgieron los servicios web, mediante estos las organizaciones pueden hacer que su información sea accesible por otros programas realizados en diferentes plataformas. Esto es realmente una interfaz en la web. Esta interfaz define las rutas y maneras para acceder a los recursos. De una forma más general, un servicio web representa un recurso informático que pueda ser usado por otros programas.

Se puede ver un ejemplo como un servicio de recolección de información de una entidad tributaria en el que los usuarios diligencian un formulario para reportar sus impuestos, este será validado y enviado a un servicio de una entidad gubernamental para su control.

El objetivo de un servicio es la provisión de este, independientemente de la aplicación que lo utilice. Los proveedores de servicio pueden ofrecer servicios especializados y ofrecerlos a un cierto número de usuarios de diferentes organizaciones. Las aplicaciones pueden construirse mediante el enlace con estos servicios.

En la actualidad existen varios modelos de servicios, como el modelo JINI, servicios web y servicios de rendija. Todos los modelos mencionados operan de una manera similar.

Un proveedor de servicios oferta uno de estos junto a la definición de su interfaz y su funcionalidad; un solicitante del servicio enlaza este a su aplicación, mediante rutinas que código para consumirlos y procesar los resultados. En diferentes casos, el proveedor puede asegurarse que el servicio sea accedido por usuarios externos especificando un registro de entrada y la información necesaria.

El proveedor de servicios hace pública la información para que usuarios con permisos necesarios puedan usarlo, en muchos casos el proveedor y el usuario no tienen que hacer una negociación sobre los antecedentes del servicio antes de enlazar con la aplicación.

Las aplicaciones pueden retrasar su enlace debido a que algunas deben estar desplegadas en el entorno de trabajo para que funcionen correctamente. Por lo tanto, una aplicación que usa un servicio específico puede cambiar dinámicamente a otro mientras está en ejecución.

Un proveedor de servicios puede identificar diferentes oportunidades para ofrecer servicios innovadores en la manera de enlazar, de manera que sea más simple entablar una relación entre el servicio y el cliente.

Los usuarios pueden adquirir un servicio sin necesidad de comprar un componente de software costoso, de esta manera los clientes pagan únicamente por el tiempo que este es utilizado.

Las aplicaciones pueden ser más pequeñas debido a la exteriorización de procesos, estas también pueden ser dinámicas y adaptarse con su entorno enlazado a diferentes servicios.

Dada la naturaleza de un servicio web, se puede determinar que su arquitectura es débilmente acoplada y dependiendo de sus interacciones puede ser fuertemente colisionada, aunque muy estáticas.

Uno de los problemas o puntos de inflexión que puede afectar la arquitectura de servicio, haciéndolo más eficiente, se debe a que los estándares para los componentes reutilizables han sido desarrollados hace poco tiempo. Aun así, los servicios web desde su planteamiento inicial ya contaban con estándares. (**I. Sommerville and M. I. A. Galipienso, 2005**)

Los 3 estándares más utilizados son:

"*SOAP: Este protocolo define una organización para intercambio de datos estructurados entres servicios web*

WSDL: Este protocolo define como pueden representarse las interfaces de servicios web

UDDI: Este es un estándar de búsqueda qué define como puede organizarse la información de descripción de servicios, usadas por los solicitantes de los servicios para encontrar otros servicios"[4]

Los estándares mencionados están basados en XML, que es legible por personas y computadores; no conocer este estándar, no causa una limitante para conocer los servicios web.

Gracias a las implementaciones anteriormente nombradas, han surgido soluciones para innumerables problemas informáticos en función del consumo de servicios, pero aun así los procesos de implementación de APIs en distintas plataformas para consumir estos servicios pueden ser tediosos, debido a que el nivel de complejidad que requiere el cliente es muy alto. De esta dificultad reluce un nuevo estándar denominado RESTFUL, **REpresentation State Transfer**, que no es una arquitectura como tal, pero especifica restricciones que al ser aplicadas al diseño de un sistema, estructuran una arquitectura de software.

Este estándar facilita la creación de sistemas distribuidos sin ninguna restricción, gracias a que en las fases de soporte se facilita su mantenimiento y el de los clientes que la consumen. **(J. Sandoval, 2009).**

Gracias a esta facilidad mencionada y a la necesidad de escalabilidad que naturalmente se da en un sistema de streaming, se implementara este estándar para facilitar que diferentes aplicaciones de diversas plataformas accedan al sistema administrador central. El sistema administrador central será el software encargado de administrar el servicio de streaming para gestionar la entrega de los canales a las aplicaciones clientes y especificar cómo debe hacerse la conexión entre las fuentes y los servidores.

Utilizar el servicio web para distribuir los puntos de montaje entre las aplicaciones cliente, no solo ayuda a una gestión automatizada, sino que también hace parte de la arquitectura y forma una capa de gestión y una de seguridad, debido a que este servicio es el encargado de consultar a los servidores los puntos de montaje, unirlos en un solo objeto y entregarlo unificado a los cliente, de manera que estas no tienen acceso directo a la autenticación, este factor evita que al ser descomprimida una aplicación el atacante no pueda conocer la manera de acceder directamente a los servidores de transmisión mediante las contraseñas de administrador de Icecast 2.

6.1.8. Java y netbeans.

Bill Joy, de SUN MICROSYSTEM y su equipo de desarrollo, ejecutaban el proyecto “Green” que consistía en desarrollar aplicaciones para teléfonos móviles y televisores interactivos.

[4] **I. Sommerville and M. I. A. Galipienso**, Ingeniería del software, Pearson Educación, 2005, 687 páginas, 9788478290741

Aplicaron el paradigma de la programación orientada a objetos y utilizaron C++, pero este pronto mostró que sus capacidades eran limitadas para poder cumplir con el objetivo, ya que reveló diversas complicaciones experimentadas con las arquitecturas y sistemas operativos testeados. Gracias a esos problemas presentados, dos desarrolladores de SUN, James Gosling yPatrick Naugthon desarrollaron JAVA EN 1991. ***(T. Groussard, 2012).***

El objetivo de JAVA, consistió en entregar una plataforma destinada al desarrollo multiplataforma y al despliegue de aplicaciones seguras en sistemas heterogéneos de entornos distribuidos, que funcionen en cualquier dispositivo digital físico.

JAVA cubrió varias necesidades a la vez, que fueron:

- Define una sintaxis sencilla, orientada a objetos e interpretada que optimiza la compilación y la ejecución.

- Las aplicaciones son portables y pueden ser instaladas en diferentes plataformas sin requerir modificaciones.

- Java tiene un mecanismo de gestión de errores estricto, que se encarga de la gestión de la memoria y es más fácil eliminar errores en los programas.

- Permite ejecutar varios procesos multihilos para hacer las aplicaciones gráficas más ligeras.

- El funcionamiento de las aplicaciones es seguro, ya que el motor no permite que las aplicaciones, sobretodo tipo Applets que pueden realizar acciones peligrosas.

"SUN caracteriza a JAVA como un lenguaje sencillo, orientado a objetos, distribuido, interpretado, robusto, securizado, independiente de las arquitecturas, portable, eficaz, mutihilo y dinámico."[5]

La sintaxis de java es basada en la de C y C++, pero no tiene características semánticas que hacen los lenguajes complejos, como:

- Java solo posee 3 tipos de datos primitivos y son: los numéricos, el tipo carácter y el tipo booleano.

- Las tablas y cadenas de caracteres son abstracciones de clases o más conocidos como objetos.

[5]**T. Groussard**, JAVA 7: Los fundamentos del lenguaje Java, Ediciones ENI, 2012, 410 páginas, 9782746073180

- Tiene un sistema llamado recolector de basura, que se encarga de gestionar la memoria, que requiere un objeto y posteriormente liberarla cuando no ya no se requiera.

- JAVA, no requiere ficheros de encabezamiento.

- JAVA, permite la simulación de herencia para evitar problemas de ambigüedad mediante Interfaces.

- Utiliza la sentencia break o continue para salir de ciclos en ejecución, en vez utilizar la sentencia goto.

JRE es el componente encargado de gestionar la memoria de los objetos y no el compilador como sucede con C++. Siendo JAVA un lenguaje distribuido, implementa los protocolos de acceso a red como: ftp, http, telnet y etc. Estos pueden ser configurados en el JRE, para autorizar diferentes tipos de acceso a la aplicación.

Una compilación de JAVA es ajena a la arquitectura, debido a que no genera un código específico para cada una, sino que genera bytecode que es independiente de ésta y del sistema operativo. La ventaja de este modelo es que el bytecode es fácil de interpretar y transformar en código nativo para optimizar el rendimiento.

En conclusión, se puede notar que JAVA sobresale por, el hecho que se compone de una parte de software, que se puede ejecutar en diversas plataformas físicas y diversos sistemas operativos***. (T. Groussard, 2012).***

Netbeans es el IDE oficial para desarrollar en JAVA. Provee funciones para editar, analizar código y convertir las aplicaciones para la plataforma que se desee.

Con su constante mejora del editor de JAVA, se caracteriza por su soporte a la codificación y su amplia gama de herramientas, plantillas y muestras que componen el conjunto de tecnologías de vanguardia. Es más que un editor, ya que contribuye a la codificación, numerando las líneas de código y las alertas que pueda tener respecto a estas, usando como criterio la sintaxis y la semántica. Permite refactorizar y organizar fácilmente el código, mientras mantiene un historial del avance de un fichero, permitiendo una visión del avance del programa. El editor también provee recomendaciones de código, consejos de codificación y generadores de código. Siendo JAVA el campo fuerte de Netbeans, se conoce que también tiene soporte para C, C++, XML, HTML, PHP, CSS, JavaScript, CSS y JSP. El editor es extensible y por ese motivo, permite dar soporte a diferentes idiomas.

Netbeans facilita la gestión intuitiva del proyecto, ya que gestiona proyectos de hasta miles de carpetas y archivos, y millones de líneas de código. Este IDE ofrece diferentes distribuciones de vistas del proyecto y también permite usar herramientas como Mercuri o Gif.

Provee herramientas de diseño y desarrollo de interfaces de usuario, mediante arrastrar y soltar para facilitar al desarrollador la tarea del diseño y la navegabilidad. Estas herramientas son útiles para crear prototipos de interfaces para clientes.

Maven es una herramienta open source incluida en Netbeans, que sirve para administrar un proyecto JAVA, este provee recursos para todas las fases del desarrollo desde la creación del proyecto, hasta la compilación del mismo.

Una función muy importante que cumple Maven es facilitar la administración de las librerías mediante un repositorio, que hace fácil la adquisición de componentes de software, que se puedan requerir en el proyecto***. (CAMACHO Erick, 2015).***

RestEasy es un proyecto JBoss que facilita un (1) framework que ayuda a construir Servicios Web tipo RESTFul. Esta API está certificada por la especificación JAX-RS, y provee los mecanismos para facilitar el desarrollo y despliegue de servicios web sobre protocolo HTTP.

Utilizando Maven para JAVA se va a importar del repositorio los ficheros de RestEasy, y con esta tecnología, se realizará el algoritmo central de administración inteligente de recursos.

6.1.9. Android y android studio.

La sociedad actual ha cambiado gracias a la tecnología móvil, que ocasionó un impacto similar al que causó internet. Esta revolución hasta ahora está tomando fuerza, ya que los nuevos terminales cada vez ofrecen capacidades más cercanas a los ordenadores personales, al punto de permitir a las personas, realizar actividades como: revisar correo electrónico, navegar por internet, ejecutar aplicaciones e inclusive ejecutar videojuegos con robustos renders 3D. Pero a diferencia de un computador personal, los móviles los se puede llevar en el bolsillo a cualquier sitio, esto da lugar a una nueva gama de aplicaciones enfocadas en la movilidad que estos dispositivos proveen.

Android desde su lanzamiento causó grandes expectativas y estremeció el mundo de los sistemas operativos móviles, al punto de convertirse en la alternativa estándar frente a otras plataformas como: IPhone, Symbian, Windows phone, BlackBerry, Palm, Java Mobile Edition y Linux Mobile, Las características que le han permitido posicionarse frente a otros S.O son:

- Es una plataforma completamente abierta: es de desarrollo libre, está basada en Linux y es de código abierto.

- La mayor ventaja para los fabricantes de celulares o desarrolladores de versiones, consiste en que no se deben realizar pagos si se desea modificar el sistema.

- Adaptable a cualquier tipo de Hardware: el diseño del sistema operativo no fue enfocado solo en dispositivos móviles como teléfonos inteligentes y tabletas, y esto se puede notar con los surgimientos de: los relojes inteligentes, cámaras HD, electrodomésticos, automóviles y otra gran cantidad de dispositivos que lo soportan.

- Portabilidad asegurada: el lenguaje por defecto de las aplicaciones es JAVA, esto permite que las aplicaciones sean ejecutadas en cualquier arquitectura.

- Arquitectura basada en componentes inspirados en Internet: la configuración gráfica se estructura en un XML, lo que permite que la interfaz de usuario se adapte a cualquier tipo de pantalla.

- Dispositivo con conexión constante a internet: siendo la movilidad uno de los fuertes en la telefonía móvil, se puede deducir que sin accesibilidad a Internet no sería lo que es.

- Servicios físicos incorporados: la considerable cantidad de servicios abarcan, GPS para geolocalización, bases de datos con SQL, reconocimiento y síntesis de voz, software para navegación, aplicaciones multimedia y etc.

- Considerable nivel de seguridad: debido a que las aplicaciones se ejecutan cada una sobre una instancia de una máquina virtual, no tienen acceso entre ellas.

- Optimizado para usar pocos recursos: Android se ejecuta sobre Dalvik, una versión optimizada de la máquina virtual de JAVA.

- Alta calidad de gráficos y de rendimiento de audio: Android implementa estándares comunes de alta calidad como H.264 (AVC), MP3, AAC y otros muy utilizados. ***(Desarrollo Android, 2015).***

Como se ha visto anteriormente, Android tiene características que lo pueden convertir en el sistema operativo estándar para móviles, pero no hay que obviar el esfuerzo que realizan otras empresas como Apple con IOS y Windows con su sistema operativo, sin embargo, Android tiene un 75% del mercado y difícilmente disminuirá su participación en este.

Android provee a los desarrolladores, una plataforma llena de innovaciones y estructuras bases para acceder a diferentes soluciones como: sensores, GPS, servicios como notificaciones y otra serie de componentes físicos.

Está también simplifica la reutilización de código, debido a la estructura de sus lenguajes de programación orientada a objetos. Este mismo mecanismo permite a los usuarios finales hacer actualizaciones y reemplazos de componentes.

El SDK de Android, no tiene por completo la implementación de la máquina virtual de JAVA (JRE), pero si con un conjunto limitado de sus especificaciones como:

- **Views**: conjunto de Frames para diversas vistas.
- **Resource Manager**: permite acceso a recursos diferentes al binarycode.
- **Activity manager**: maneja de la ejecución de las aplicaciones y el sistema de navegación entre ellas.
- **Notification Manager**: permite que los procesos envíen notificaciones al usuario.
- **Content Providers**: mecanismos para acceder a datos de otras aplicaciones si el desarrollador lo permite.

Actualmente el lenguaje por defecto, para desarrollar aplicaciones es JAVA y para ello, se utiliza el SDK, pero si se requiere hacer la implementación en C++ es posible usar la NDK para Android.

Las aplicaciones para Android pueden ser soportadas en todo el ciclo de desarrollo por diferentes entornos integrados como Eclipse, Netbeans y el que se utilizará para la Aplicación que será, Android Studio que es catalogado como el IDE oficial para desarrollo de aplicaciones Android.

Android Studio es un entorno de desarrollo integrado que permite desarrollar aplicaciones desde un editor inteligente capaz de complementar código complejo, refactorizarlo o analizar piezas de código en busca de errores.

Este poderoso editor ayuda a hacer más productiva la codificación, gracias a las ayudas que posee como, el lanzador de aplicaciones que permite iniciar un proyecto Android de una manera muy sencilla o las recomendaciones en tiempo real de la codificación. Este IDE posee plantillas integradas con funciones de codificación y estructurado, que pueden ser utilizadas para facilitar la comprensión del lenguaje y brindar bases, para diferentes proyectos a programadores de alto nivel. Este también permite enlazar con una cuenta de repositorios Gif para importar proyectos a nuestro espacio de trabajo.

El editor también posee el soporte para construir aplicaciones para diferentes tipos de dispositivos con pantallas de tamaños diferentes como, celulares inteligentes, televisores, relojes inteligentes, para autos o gafas de realidad aumentada. Esto se facilita a tal punto, que no se debe implementar un proyecto por cada plataforma, sino que se puede tener diferentes vistas configuradas con la misma implementación de código. De esta manera se pueden realizar diferentes compilaciones utilizando el mismo proyecto. ***(Desarrollo Android, 2015).***

Debido a la movilidad de Android y la actual masificación de la plataforma junto a su IDE oficial que facilita enormemente el desarrollo, se toma la decisión de realizar el proyecto en esta enriquecida plataforma con el framework oficial.

6.1.10. Apache cordova.

Es un framework open source, que permite construir aplicaciones multiplataforma, utilizando HTML5. Los desarrolladores de esta, deseaban hallar un mecanismo para desarrollar aplicaciones para múltiples dispositivos digitales con diferentes sistemas operativos, y la opción que más se adecuaba a su necesidad era el estándar HTML5, por su auge en las diferentes plataformas y su amplia adopción.

Desafortunadamente las aplicaciones requieren más que solo HTML5, ya que no todos los navegadores de las plataformas lo soportan de la misma forma, debido a que la implementación de una aplicación puede requerir el acceso a los servicios del dispositivo como la cámara, los sensores o el GPS y cada plataforma ofrece mecanismos completamente diferentes.

Por ende, se desarrolló la API de Cordova, que es capaz de interactuar con el lenguaje nativo de las plataformas para tener acceso a los diferentes periféricos y servicios presentes en el dispositivo.

Apache Cordova está compuesto por los siguientes componentes:

- EL código nativo, que contiene el soporte para las diversas plataformas
- EL núcleo de la API que permite ejecutar una aplicación web en un entorno con acceso a la implementación nativa.
- Una serie de herramientas que permiten el ciclo de vida de la aplicación.

Para crear la aplicación Cordova, se puede iniciar realizando el proyecto WEB y posteriormente, este se puede importar al contenedor de la aplicación nativa y enlazarla con la API de Cordova. ***(J. M. Wargo, 2014).***

Dentro de la aplicación nativa, la interfaz de usuario consiste de un simple frame que carga la aplicación web, despliega el índex y pasa el control al web view para permitir la interacción con la parte nativa.

Estas aplicaciones se ejecutan de manera similar a si estuvieran implementadas en el navegador WEB del dispositivo, pero con acceso al lado nativo.

Las aplicaciones compiladas con Cordova, tienen la capacidad de utilizar JavaScript embebido en la aplicación o ubicado en una URL externa, estos pueden ser los que provean la lógica para comunicarse con el servidor, explorar imágenes, reproducir multimedia y mediante una interfaz JavaScript que trae consigo Cordova, se puede enviar mensajes a la implementación nativa. Para realizar el diseño gráfico de estas se puede utilizar CSS en toda su magnitud sin que ningún atributo sea discriminado.

Apache Cordova soporta una serie de plataformas y la lista empieza con Android, Samsung BADA, Blackberry, Apple IOS, Palm WebOS, Symbian, Windows Phone 7 y 8, Firefox OS, Ubuntu OS y Amazon Fire OS otra serie de soportes que se están desarrollando.

Las aplicaciones con Cordova se programan usando tecnología web de todos los días como HTML, CSS Y JavaScript. Utilizando todos estos estándares de tecnología, se puede utilizar todo el potencial que tienen, como por ejemplo, JavaScript que posee un soporte de un gran conjunto de librerías que funcionan con diferentes fines.

Cordova no provee un IDE oficial, ya que desarrollar un proyecto con este framework solo requiere un editor de texto u otro IDE como Dreamweaver y eclipse, que proveen herramientas para dar soporte a la implementación en todos sus ciclos de vida y así facilitar la codificación.

Cada plataforma que soporta Cordova, tiene sus herramientas específicas para compilar las aplicaciones, para construir estas se debe agregar el proyecto web a la estructura especificada por las plataformas independientemente cada una de las otras, generalmente esta adición se realiza en un marco de contenido HTML que se configura de manera diferentes en las herramientas de cada una. La diferencia entre una y otra plataforma se da en los ficheros de configuración que requiere la aplicación, para funcionar correctamente, estos ficheros junto a la estructura de los mismos se especifican en base a la plataforma. Adicionalmente a esto en contadas ocasiones se requiere de un Sistema operativo específico para realizar la compilación como en el caso de IOS, que requiere tener un MAC para compilar la aplicación. ***(J. M. Wargo, 2014).***

Córdova es una excelente opción para desarrollar proyectos multiplataforma, pero tiene diferentes limitantes, debido a que se adhiere otra capa a la aplicación, esta disminuye seriamente el rendimiento en los dispositivos. Esto afecta los dispositivos de baja gama que posee menos procesamiento que en teléfonos de baja media y alta.

Es deducible que Cordova es funcional para desarrollar proyectos de pequeña y mediana escala, pero si el proyecto es grande, sería mejor optar por realizar el desarrollo, utilizando únicamente los lenguajes nativos.

6.1.11. MySQL.

Es un motor que sirve para gestionar bases de datos relacionales de manera rápida y sencilla. Es recomendada para desarrollar sistemas de información que requieran almacenamiento.

Es ideal para sistemas como páginas web, aplicaciones móviles y en general todo tipo de solución informática que sea transaccional.

MySQL ofrece ventajas sobre otros motores de bases de datos:

- Es software tipo open source.
- Fue construida con C y C++, facilita el acceso a aplicaciones desarrolladas con estos lenguajes.
- Es de acceso gratuito mediante la WEB oficial.
- Para quienes desean hacer desarrollos comerciales, también pueden adquirir esta licencia.
- Utiliza lenguaje SQL para permitir un acceso estandarizado a bases de datos relacionales.
- Pertenece a la arquitectura de cliente/servidor, teniendo la opción de ocupar los dos roles, ya sea en sistemas con bases de datos distribuidas o ejecución de réplicas de seguridad.
- Es multihilos, de manera que admite conexiones simultáneas y múltiples subprocesos.
- Dispone de amplio soporte en línea para resolver diversos inconvenientes.
- Es portable, ya que entre sus distribuciones se soporta: MAC, UNIX, LINUX y Windows.
- Se puede encontrar un gran soporte de usuarios de todo nivel en internet.

Las características anteriores hacen de MySQL uno de los motores de bases de datos más utilizados por pequeñas y grandes empresas.

De MySQL se puede obtener dos tipos de aplicaciones, el cliente de monitoreo y el servidor.

El servidor se encarga de recibir las peticiones realizadas por los clientes y el cliente de monitoreo se encarga de facilitar una interfaz gráfica para la gestión. Si no se cuenta con una interfaz gráfica de usuario, se puede utilizar el cliente de monitoreo que facilita mediante la línea de comandos él envió de diferentes órdenes al sistema, se puede enviar desde comandos de ejecución del programa, hasta consultas SQL para una base de datos específica.

Al utilizar MySQL por intermedio de un sistema cliente de monitoreo o interfaz gráfica, se requiere de un nombre de usuario, una contraseña de acceso y una conexión válida al servidor.

Iniciar MySQL es muy sencillo y se puede realizar fácilmente desde la línea de comandos o mediante una interfaz gráfica, como por ejemplo los mecanismos de inicio de sistemas como los poseen programas como XAMPP, WAMP, LAMP, Vertrigo, etc.

Al momento de instalar MySQL se crean automáticamente cuatro cuentas; dos (2) de administrador con usuario root sin contraseñas y dos (2) como invitado sin usuario ni contraseña. Para garantizar la integridad de las bases de datos almacenadas en el motor, es evidente que la primera operación que se debe realizar, posterior a la instalación, sería restablecer las contraseñas de acceso de los usuarios.

El mecanismo para manejar privilegios por parte de MySQL, varía entre diversas configuraciones con el fin de aumentar la seguridad y mantener la integridad.

Por este motivo el control de privilegios puede establecerse de manera individual para cada base de datos e inclusive para cada tabla de la base de datos.

MySQL tiene un extenso conjunto de funciones para manipular los registros y realizar diferentes operaciones en torno a los datos y el conjunto de información de una base de datos específica. Permite: inserción de registros, eliminado de registros, actualización de estos y una herramienta muy importante para exportar e importar datos.

Existen varios mecanismos de exportación de datos, uno de estos es el comando mydump que permite generar un archivo plano con los comandos necesarios para reconstruir una base de datos. Este contendrá todas las sentencias para crear las tablas y restablecer los registros.

MySQL tendrá varias funciones en el sistema de streaming que son:

- Va a contribuir en la gestión de los servidores del núcleo del servicio, ya que en una base de datos, se va a realizar el almacenamiento de los datos necesarios de los servidores (IP, puerto, usuario, contraseña, etc.).

- Para poder facilitarle al algoritmo de gestión inteligente del servicio, y que este pueda extraer información que necesite, para realizar los cálculos y facilitar el manejo de los puntos de montaje y la distribución de los usuarios.

- En la base de datos se va a almacenar la información de los usuarios oyentes para permitir que la aplicación Android cumpla con su objetivo.

- También se registrará la información del usuario DJ, respecto a las características de su streaming como género musical, nombre artístico, perfiles de redes sociales y todo dato que requiera la aplicación y los usuarios para tener una buena experiencia con el sistema.

En la misma base de datos se va almacenar toda la información relativa al funcionamiento y a los usuarios, pero aun así es necesario destacar que el acceso a los datos de los usuarios de sistemas de streaming y a los de los servidores de streaming, por parte de los programas clientes o fuentes, estará limitado por los permisos a los usuarios de la base de datos.

Lo anterior se debe a que, no es seguro permitir que las aplicaciones tengan acceso a la ruta que pueda poner en riesgo, la integridad de los datos de funcionalidad, que permiten, que funcione el servicio de administración inteligente.***(Á. Cobo, 2005).***

6.1.12. Uml.

Este es un lenguaje de presentación gráfica para modelar sistemas y procesos. Está especificado para estructurarse en la programación orientada a objetos, lo cual lleva a que estos lenguajes como JAVA, C++ Y otros sean totalmente compatible con este, al momento, de iniciar un proyecto de software.

Este lenguaje es una especificación unificada de varias notaciones que lo preceden. UML es promovido por la OMG (object Management Group), que se constituye como un consorcio con más de 800 organizaciones que hacen parte de él.

UML está siendo adoptado cada vez con más rapidez debido a que es una especificación de diseño que permite abstraer la funcionalidad sin involucrarse en aspectos técnicos. Para utilizar esta notación no se requiere de un rol específico, ya que puede ser utilizado por ingenieros informáticos, jefes de proyectos, directores o en sí, una persona con el conocimiento necesario para tener una visión general de un sistema.

En los lenguajes de programación, se utilizan sintaxis muy estrictas, estas resultan ser ilegibles para las personas que no saben de programación y se tornan complicadas de entender para otros desarrolladores, ya que cada persona tiene una lógica diferente.

Para los humanos es más simple utilizar lenguajes gráficos para representar modelos, ya que este lenguaje se entiende con más facilidad.

En 1994 James Rumbaugh y Grady Booch, unen sus notaciones procedentes de sus métodos personales. En 1995 Yvar Jacobson decide unirse al equipo. En 1997 publican la primera versión, producto de una evolución de notaciones, pero al versen alcanzados por la tediosa labor, solicitan soporte a la OMG, que adopta UML en 1997 y lanza la versión 1.1. Posteriormente ellos crean una división especial para encargarse únicamente de la evolución de UML. Desde ese momento, estos se han encargado de sacar versiones con amplias posibilidades de describir inclusive acciones y la semántica de estas. Actualmente se encuentra en su versión 2.4.

Es necesario especificar que la notación UML no es una metodología sino un conjunto de herramientas gráficas, que soportan el modelado de sistemas de información y procesos.

UML especifica los siguientes conceptos, que deben ser entendidos para poder realizar los diagramas:

- **Objeto:** es una abstracción del mundo real, que poseen atributos junto a una serie de métodos que describen las acciones de comportamiento que poseen. Ejemplos de un objeto, puede ser: una persona, un animal, un mueble; o para ser más específicos un perro, un veterinario, una mesa, etc.

- **Abstracción:** determinar las propiedades de un elemento en UML, es una herramienta analítica muy importante, porque determina los objetos utilizados en los diagramas.

- **Clases de objetos:** es un conjunto de objetos que poseen atributos y métodos en común y poseen un comportamiento similar.

- **Especialización y generalización:** puede haber clases que contengan otros conjuntos de clases. Por ejemplo, la clase mamíferos tiene dos subclases que son perro y caballo (generalización), a la vez que mamíferos es la súper clase de las otras dos (especialización).

- **Herencia**: permite que una subclase, posea los atributos y métodos de la superclase a la que pertenece.

- **Clases abstractas y concretas**: una clase concreta, es aquella que describe con precisión un objeto, por ejemplo, un caballo o un perro, que se conoce específicamente su existencia y sus propiedades.

- Una clase abstracta, es una clase que no define un objeto existencial, en cambio define una representación más general, como la clase mamíferos, que es un conjunto de animales con ciertas características en común pero la clase como tal no define a un animal específico.

- **Polimorfismo**: es cuando la clase, representa un conjunto de objetos que pueden pertenecer a diferentes subclases, en pocas palabras, que poseen diferentes características.

- **Conclusión**: los diagramas de UML, representan elementos enfocados en la programación orientada a objetos, de esta manera se puede conocer los elementos que deben ser programados.

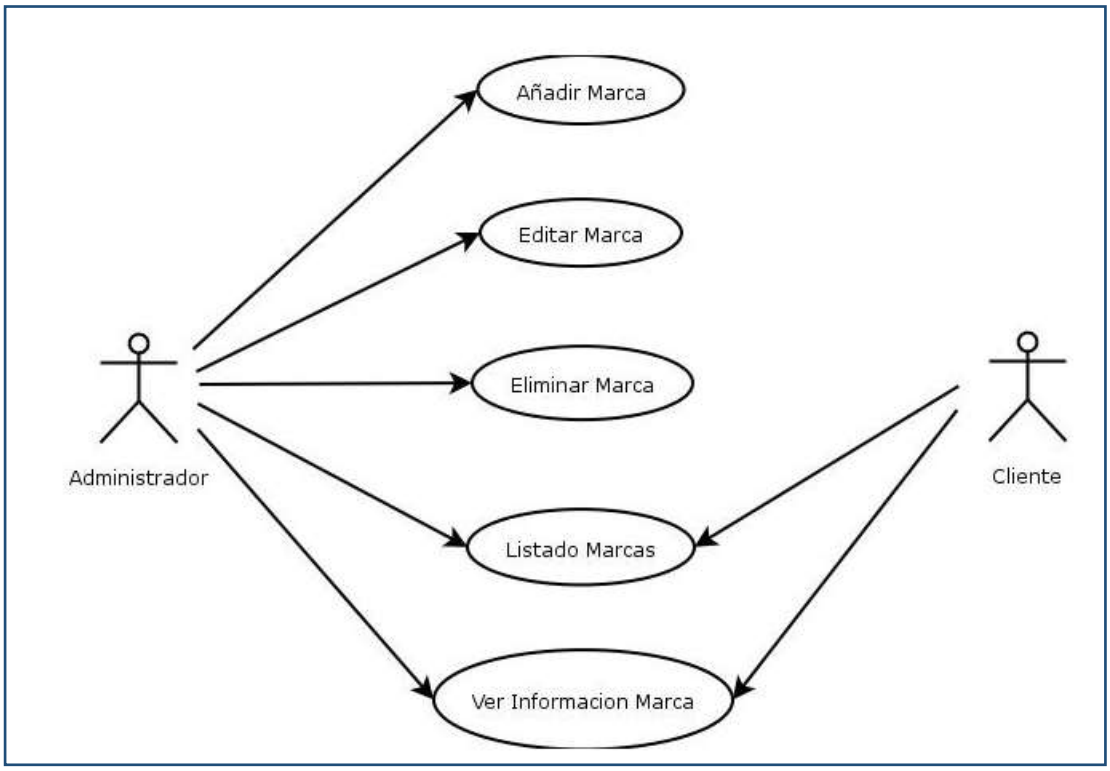
Para realizar un correcto diseño de un sistema se deben utilizar los diagramas de: casos de uso, flujo de datos, clases e interfaces de usuario.

El diagrama de **casos de uso** permite conocer las interacciones que van a tener los usuarios con el sistema y para esto, se deben definir correctamente:

- Verbos adecuados en la descripción de cada actividad
- Correcta abstracción de los diferentes tipos de usuarios en el sistema. *(**L. Debrauwer and F. van der Heyde, 2005**).*

A continuación, un ejemplo de un diagrama de casos de uso de un módulo de autenticación y registro de usuarios:

Figura 2. Ejemplo casos de uso



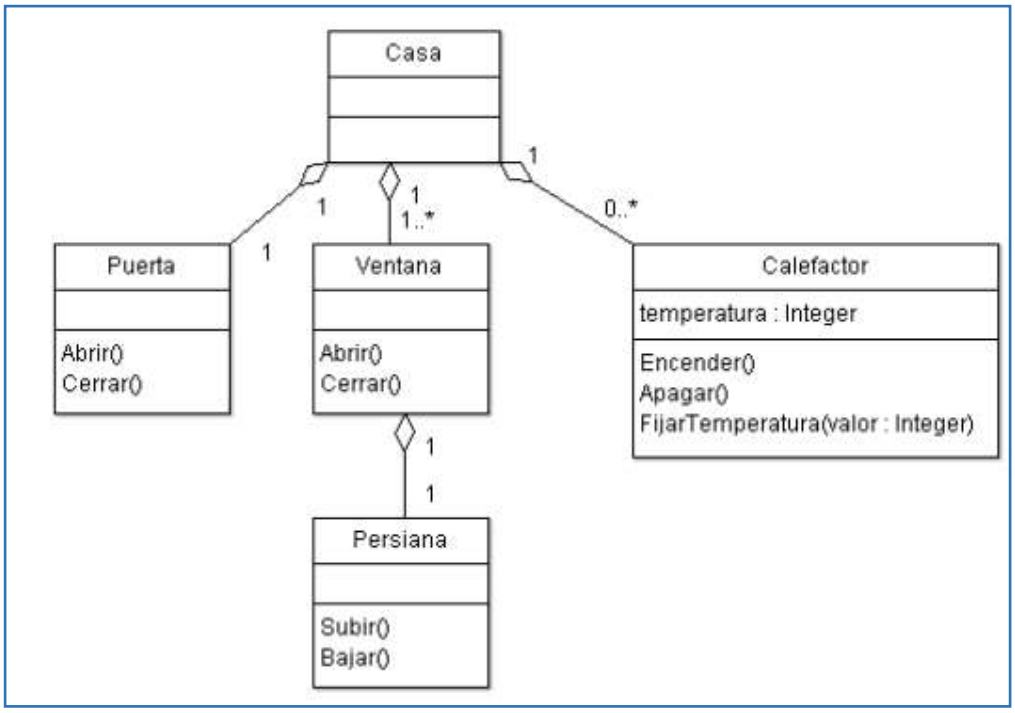
Fuente: Tomada de,uac.edu.co, 2015.

Los **diagramas de clase**, especifican la abstracción de las clases que van a ser utilizadas y las relaciones entre estas, los elementos abstractos deben contener:

- Nombre descriptivo de la clase
- Atributos característicos de los objetos que la componen
- Métodos que definen el comportamiento de los objetos

A continuación, un ejemplo de un diagrama de clases de un sistema del sistema de gestión de una casa con sus objetos principales que son: las puertas, las ventanas, las persianas y la calefacción.

Figura 3. Ejemplo de diagrama de clases



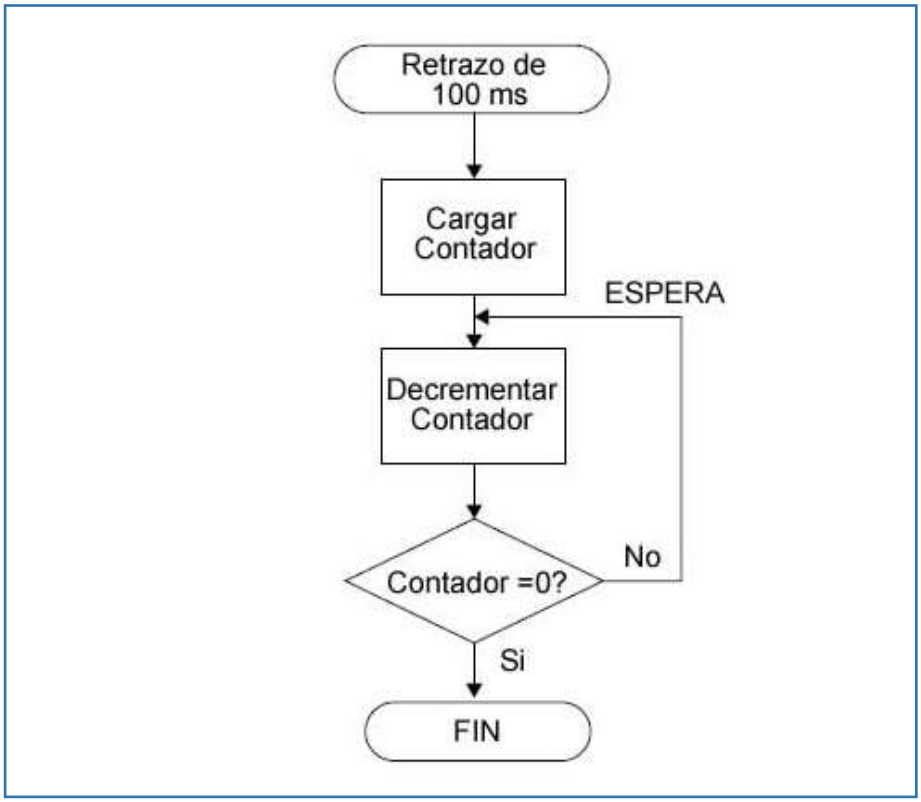
Fuente: Tomada de nachocabanes.com, 2015.

El **diagrama de flujo de datos,** permite determinar las rutas que van a poder tomar los datos en el sistema y las consecuencias que se generan por alguna acción de estos, en estos diagramas se deben tener en cuenta:

- Los datos de entrada los datos del sistema.
- Las entidades, que representan los diferentes actores que ejecutan acciones en el sistema.
- Los procesos que realiza el sistema y la relación que estos poseen con almacenes de datos y entidades.
- Las tomas de decisiones para permitir que los datos cumplan ciertas condiciones.
- Los almacenes de datos donde reposa información almacenada
- Las relaciones entre todos y cada uno de los elementos que están encadenados.
- La salida de los datos procesados o respuestas del sistema, si así se requiere.

A continuación, se observa la imagen de un diagrama de flujo de datos que plasma el proceso para realizar una suma de un número predeterminado y un número, ingresado por consola.

Figura 4. Ejemplo de diagrama de flujo de datos



Fuente: Tomada de datateca.unad.edu.co, 2015.

EL **diagrama de interfaz humana**, permite diseñar correctamente la navegación del usuario, para esto se debe especificar las rutas de acceso mediante la interfaz gráfica, a las funciones del sistema que se esté diseñando.

6.1.13. Metodología xp

Programación extrema es un estilo de desarrollo, está enfocado en mejorar la productividad de las técnicas de programación, mejorar la comunicación y proveer acoplamiento en los equipos de trabajo, XP incluye:

- La filosofía del desarrollo basado en los valores de , comunicación, retroalimentación, simplicidad, coraje y respeto.

- Las personas y sus interacciones son más importantes que los procesos.

- Es más importante realizar el software que la documentación exhaustiva.

- Se colabora activamente con el cliente.
- Se realizan cambios inmediatos, en lugar de seguir un plan riguroso.

Este movimiento tuvo sus inicios en el año 2001, cuando se reunieron representantes de metodologías como: SCRUM, DSDM, Crystal y llegaron a un acuerdo conjunto que dio lugar a XP.

Esta es definida por la universidad del valle como, *"una metodología de desarrollo ligera (o ágil) basada en una serie de valores y de prácticas de buenas maneras que persigue el objetivo de aumentar la productividad a la hora de desarrollar programas"*[6]

En esta se le da prioridad a las actividades que dan resultados concretos, para reducir la burocracia que hay en torno al desarrollo de software y así cumplir con el objetivo principal que fue, determinar un mecanismo para hacer del proceso de desarrollo de una actividad sea más sencilla.

XP tiene 12 principios que se categorizan en 4 bloques:

Retroalimentación a escala final:

- **Principio de pruebas**: se debe determinar un tiempo específico, para realizar pruebas de funcionalidad, de manera que se ingresen datos al sistema y se analice la salida de estos, para observar cumplen con lo esperado.

- **Proceso de planificación**: se realizarán entrevistas y dinámicas con el cliente, para hacer que este defina claramente sus necesidades mediante la narración de historias, posterior a esto, se puede definir el cronograma de trabajo basado en las historias.

- **EL cliente en el sitio**: Se tendrá acompañamiento por parte de un representante del cliente, que podrá contribuir a definir requerimientos funcionales, definir prioridades y responder dudas del programador. De esta manera se obvia el proceso de creación de documentación extensa.

- **Programación en parejas**: se requiere que todos los programadores involucrados en el proyecto realicen su codificación en parejas.

[6]***Universidad del valle***,Programación extrema, Disponible en (http://eisc.univalle.edu.co /materias/WWW/material/lecturas/xp.pdf). [Consultado el 24 de mayo del 2015]

Proceso continuo

- **Integración continua**: permite que los desarrolladores tengan un rápido progreso, de manera que las implementaciones de nuevos requerimientos surgidos pueden ser realizados, para evitar cumplir un cronograma a cabalidad y no tener que esperar a realizar nuevas versiones.

- **Refactorización**: permite que el equipo de desarrollo, realice cambios y mejoras en diferentes puntos de ejecución del proyecto. La idea en mejorar la calidad y esto se hace, revisando constantemente el diseño y aplicar inmediatamente las correcciones necesarias.

- **Entregas pequeñas**: se realizan pequeñas entregas que van siendo implantadas en un servidor de trabajo, para evaluar su desempeño en el entorno real.

Entendimiento compartido:

- **Diseño simple**: se pretende entregar un producto sencillo que cumpla con los requerimientos.

- **Metáforas**: define una historia de cómo es la funcionalidad del sistema en su totalidad.

- **Propiedad colectiva del código**: nadie es el propietario de nada, XP tiene como argumento, que entre más personas trabajen sobre un código, menos errores contendrá.

- **Estándar de codificación**: se definen las reglas que deben aplicar los programadores para escribir y documentar código, además de cómo se comunican los diferentes módulos de la aplicación.

Bienestar del programador

- **Semana de 40 horas**: los programadores agotados escriben código con menos calidad, por ende, no exigir sobrecarga laboral es igual a código de alta calidad.

En la programación extrema es necesario definir roles y equipos, uno de diseño, uno de desarrollo y uno de clientes. XP permite que los involucrados en el desarrollo de un proyecto tengan una comunicación que no se ha visto en otras metodologías, debido a que el cliente pasa a ser parte implícita en el equipo de desarrollo.

El cliente tiene un papel muy importante, ya que debe estar interactuando diariamente con el equipo para ayudar a localizar posibles problemas, sobre todo después de la mejora o rediseño de algún componente, de manera que a medida que avanza el desarrollo, el cliente va dando vistos buenos a los componentes del sistema.

Un factor muy importante para el proyecto es la planificación, que debe ser realizada por fases, donde se especificarán las diferentes iteraciones y las reglas que debe cumplir el equipo para llegar a su objetivo final.

Las entregadas deberán realizar lo antes posible y los más frecuente que se pueda, de esta manera es más fácil detectar un error desde los módulos iniciales. Cada iteración debe ser planificada para definir exactamente, que historias del usuario se desarrollaran en la iteración como tal.

XP hace consciente al desarrollador que la planeación puede contener un error, pero se define un mecanismo para corregirlos y es evaluar las historias del cliente y revisar la planificación por si requiere alguna mejora.

Para XP, las etapas de desarrollo son las más importantes, pero también es importante el diseño y la definición de los mecanismos para que este sea revisado y mejorado de forma continua, a medida que se agregan funcionalidades.

La clave para evitar errores es la comunicación, pero aun así es necesario realizar pruebas unitarias de cada módulo, para así comenzar a codificar la pieza de software que cumpla con la prueba planteada.

De esta manera se podrá obtener un software 100% funcional según los requerimientos**. (*Universidad del valle, 2015*).**

XP será implementado en conjunto con UML, ya que el segundo se utilizará para realizar el diseño de la arquitectura, y el primero para la fase de desarrollo y del diseño de interfaces gráficas.

6.1.14. Historia y configuración de hardware Dj

En los años 40s Martin Clock hace la primera referencia al término de DJ al decir que es una actuación ficticia de las mejores orquestas del país que actuaban remotamente, pero en directo desde otro sitio, el programa fue conocido como Make Believe Ballroom. Posteriormente en 1943 Jimmi Savile realiza la primera fiesta con DJ en Ottley-Inglaterra donde se escuchaban discos de Jazz.

Llegado 1947 en parís se abre un club llamado Whiskey a Go-Go que sería reconocida como la primera discoteca, ya que no actuaban orquestas en directo sino se reproducían discos aportados por DJs. A partir de ese entonces se hizo tendencia y se comenzaron a abrir discotecas en Europa y Estados unidos.

En la década de los 50s comienzan a surgir los DJs radiofónicos en la radio estadounidense. Estos únicamente podían tocar un (1) solo tocadiscos de discos de 45rpm que contenían éxitos de la época. El DJ aprovechaba a hablar entre discos para hablar o anunciar lo temas. Ya en 1955 aparece el primer DJ que realiza programas con dos tocadiscos y es Bob Casey.

Entre los 60s y 70s aparece el clásico mezclador CMA-10-2DL de Rudy Bozak y este evento logra que los DJ se hagan muy populares en las discotecas de moda del mundo.

Posteriormente en 1969 el DJ americano Francis Grasso popularizo el "beatmaching" en el club Sanctury de Nueva York, que consiste en la transición de imperceptible entre pistas de audio localizadas en diferentes discos, este se lograba con la sincronización de dos discos en el mismo tiempo y velocidad.

En 1972 Technics lanza el primer tocadiscos SL-1200 para convertirse en un referente para los DJs y junto a un mezclador plantea la configuración mínima para realizar un audio mixing**(musicaelectronicaonline.com, 2015).**

6.1.14.1.Hardware utilizado por los DJs

- **Tornamesa**: es utilizado para reproducir vinilos con máximo 4 pistas en cada cara de 33 rpm y 45 rpm. La salida análoga se genera mediante la lectura de agujas con lectores que leen las sinuosidades contenidas en el surco de los discos.
- **Unidad de CD:** reproducen CDs y DVDs con diferentes velocidades y una cantidad mayor de pistas de audio que los tornamesas, el DJ puede monitorear su medio de reproducción mediante una pantalla Digital, puede observar: la pista reproducida, el progreso de las mismas y las características de onda.

- **Mezclador:** este en el encargado de permitir que el DJ tenga una interfaz entre dos unidades de reproducción para poder sincronizar las pistas de audio y mezclarlas para que haya un flujo de audio con cambios armónicos poco imperceptibles. Además de permitirles ecualizar el audio y enviarlo a un amplificador o salida de audio.

- **Controlador DJ:** tiene con función servir de interfaz física entre el DJ y un software tipo mixing que esté instalado en un pc o un portátil conectado por cables UBS con el controlador, de esta manera este sirve como un control eficiente de una cantidad de recursos digitales para DJs con grandes alcances como efectos y samplers de muchos tipos. Este trae incorporado un mezclador y por lo generar la fuente de pistas es el disco duro del PC.

- Esto permite que el DJ tenga una gran cantidad de pistas y se enfoque más en el flujo de audio y menos en la gestión de dispositivos de almacenamiento. La salida de audio es la del PC, debido a que el controlador sirve para controlar el software como tal.

- **Amplificadores:** estos reciben la señal de salida de audio producida por el mezclador o el pc, la amplifican y la envían a los parlantes o a las cabinas de sonido.

- **Audífonos:** son utilizados para monitorear las pistas de audio que van a ingresar en el flujo, para evitar que se desincronicen con la que está siendo reproducida y se pierda la armonía que debe ser constante.
- **Micrófono:** utilizado para enlazar los comentarios del DJs con la línea de salida de audio.

- **Cabina de sonido:** recibe el flujo de audio amplificado desde los amplificadores (valga la redundancia), y los transforma para enviarlos a la salida de audio final.

6.1.14.2. Software utilizado por los DJs

El software utilizado por los DJs para realizar audio mixing está enfocado en emular las características de un set físico con capacidades de expansión de módulos de efectos y una alimentación de pista de audio a partir del disco duro. En estos programas se encontraran los módulos de soporte a protocolos RTP que permiten la trasmisión de audio con formato OGG Vorvis y MP3 a servidores de streaming.

Este tipo de software puede ser controlador mediante interfaces físicas conocidas como controladores DJ. Unos de los programas más utilizados por los DJs que soportan RTP son:

- Virtual DJ
- Traktor
- Serato

6.1.14.3. Esquemas de configuración

Para poder realizar audio mixing es necesario cumplir con unas configuraciones básicas de dispositivos anteriormente mencionados, son 2 las posibles configuraciones:

La primera es la configuración para salidas análogas y digitales sin controlador. Esta es la configuración básica ya que debe tener dos unidades de reproducción ya sean tornamesas o unidades de CD, conectadas mediante un mezclador y con salida de audio directa hacia el amplificador o periférico de salida de audio. Incluyen Audífonos para monitorear, además un (1) micrófono para incorporar mensajes de voz en tiempo real.

Figura 5. Configuración DJ de dispositivos #1

Fuente: el Autor.

La otra es la configuración de controlador. Este tiene como nodo central una maquina computacional y trae integrado el mezclador y los controles de cada canal para permitir controlar el software para DJ mixing. En esta configuración el controlador solo ejerce la función de control ya que la salida de audio se da mediante el pc que contiene el software DJ, incluyen Audífonos para monitorear y un micrófono para incorporar mensajes de voz en tiempo real.

Figura 6. Configuración DJ de dispositivos #2.

Fuente: el Autor.

6.1.14.4. Redes sociales utilizadas por los DJs.

Las redes sociales de música utilizadas por los DJs como herramientas para aumentar su alcance son:

- Soundcloud
- Mixify
- Cloudmix
- Listen to my radio

6.2. MARCO CONCEPTUAL

Acoplado: Representa el grado de dependencia de componentes de software de un mismo sistema informático. Es el grado en el que un componente puede funcionar sin requerir de otro. ***(Alegsa, 2015).***

Broadcast: es la traducción de difusión en inglés.

Buffer: es un espacio de memoria en la cual se almacenan datos temporalmente.

Cliente: es un software que consume un recurso, mediante peticiones a un servidor, usando protocolos de comunicación por red.

Servidor: es un proceso ejecutado sobre una máquina computacional, que provee un servicio mediante protocolos de comunicación por red. Unos ejemplos son: un servidor web, servidor FTP o servidor de correo electrónico.

Difusión: *"es la acción o efecto de difundir."*[7]

Fuente: es un software que provee información a un servidor. En términos de streaming, es el nodo que realiza la difusión. Unos ejemplos pueden ser: cámara web, cámara profesional, un controlador DJ o un software de escritorio.

Heterogeneidad: hace referencia a un sistema que se compone por ordenadores con diferente hardware y software, pero que se comunican con protocolos comunes. (***Sistemas etereogeneos, 2015***)

Host: es un ordenador que se encuentra al inicio o final de una transferencia de datos. **(*Más adelante, 2015*).**

IDE: Integrated development environment, en español quiere decir entorno integrado desarrollo. Es un software que provee herramientas para simplificar la codificación con un lenguaje específico.

Interfaz: medio físico por el cual se comunican dos sistemas independientes**.**

---

[7] ***Definición***, Difusión, Disponible en (http://definicion.de/difusion/). [Consultado el 25 de mayo del 2015]

Instancia: es una representación específica de una clase o prototipo.

Metadato: son etiquetas que sirven para catalogar los recursos presentes en la WWW, con el fin de interpretar y clasificar información. ***(L.L. María Jesús, 2015).***

Mimetype: es un estándar, que sirve para clasificar los ficheros de internet. ***(AIBN, 2015).1***

Punto de montaje: en términos de streaming, es una instancia del servidor que permite realizar la difusión por intermedio de esta.

RTP: *Real Transfer protocol*, es un protocolo de red que transporta datos para aplicaciones de transmisión en tiempo real, puede transportar video o audio. **(*Network sorcery, 2015*).**

SDK: conjunto de herramientas, para desarrollo de aplicaciones Android.

SQL: lenguaje utilizado para realizar consultas a bases de datos relacionales.

Stream: es la traducción de corriente en inglés.En términos de streaming, hace referencia a una corriente o flujo de datos.

Transmisión: hace referencia a la acción de transmitir.

6.3. ESTADO DEL ARTE

En Colombia y a nivel mundial se ha vivido un avance acelerado en las tecnologías de la información y las telecomunicaciones, esto ha permitido que los sistemas sean más eficientes y portables a medida que pasa el tiempo. Actualmente los DJs componen uno de los segmentos que se han visto beneficiados por este entorno tecnológico, ya que empresas locales y extranjeras han desarrollado aplicaciones y han definido procesos que les permiten a los DJs llegar a audiencias remotas en otros puntos del planeta.

En Colombia son considerables las empresas que se están dedicando a la transmisión de eventos y son contados los aplicativos, surgidos para automatizar o facilitar la transmisión en tiempo real de: radio online tradicional, sitios públicos y eventos o festivales; pero ninguno enfocado y desarrollado para automatizar y determinar procesos , para facilitar a los DJs el acceso a un robusto servicio virtual de streaming de bajo consumo de recursos informáticos, que este integrado a un sistema de información con aplicaciones clientes en plataformas móviles, para distribuir su audio mixing en vivo y directo e interactuar con la audiencia, ya sea local o remota.

Las implementaciones nacionales, prestadoras de servicios de transmisión en vivo se dividen en dos.

Primero se encuentran las que ofrecen servicio automatizado de acceso a canales de audio streaming de propósito general, que pueden ser configurados para transmisiones en vivo o consumo bajo demanda, el acceso a los servicios de estas empresas es automatizado, pero en muchos casos de alto costo o disponibilidad a pocos oyentes.

De este tipo se puede encontrar empresas como: colombiaredes.com, kalsolucionesweb.com, streamingcolombia.com.co. Streaming Colombia fue fundada en 2012 con el objetivo de ofrecer soluciones de streaming de diferentes medios, incluyendo tv, radio y eventos en vivo, opera en diferentes países de Latinoamérica **(*PIETRO SANCHEZ Juan Pablo, 2015*)**.

Kalsoluciones fue fundada en 2008 con solo un (1) empleado **(*Findthecompany.com, 2015*)** y ofrece diferentes tipos de soluciones informáticas en las cuales una de estas es el montaje de servidores y redes para la difusión de medios. Colombia Redes es una de las empresas con más experiencias en el sector de redes en Colombia, ya que opera una serie de servicios de hosting, streaming, registro de dominios y servidores dedicados desde el 2006, esta empresa ofrece el servicio de servidores de transmisión para radio con un panel de auto dj incluido. **(*colombiaredes.com, 2013)***.

Por otro lado, se encuentra un segundo tipo de empresas que requieren comunicación directa con el cliente sea por vía: telefónica, formulario web o email para concertar un pago y eventualmente realizar la conexión manualmente sin que los procesos de conexión al servidor estén automatizados, posteriormente el operador de los servidores retorna los datos de acceso al servicio al cliente, este proceso puede tardar horas e inclusive días, dos de estas empresas son: LaJRadio o Cehis.

A raíz de programas de emprendimiento e innovación en el sector, varios grupos de emprendedores han desarrollado diferentes tipos de aplicaciones con streaming que afecta de diferentes maneras el mercado y genera o enriquece nuevas experiencias, aplicaciones como: Mirolo, PlayMyWay y SecretDj.

Es de conocimiento público que las antiguas rockolas y la experiencia de escoger la música que se desea escuchar, se sitúa en una lista de reproducción ,donde será seleccionada y posteriormente escuchada, así como esta tecnología solucionó un problema de organización y deseos al momento de compartir con amigos y escuchar la música que se quiere, nace SecretDj, una aplicación lanzada en el 2012, con su reciente actualización realizada el 12 de abril del 2015 que permite a los usuarios y clientes de un bar que esté inscrito, ser dj por un día. Los asistentes a un bar que tengan instalada la aplicación en sus dispositivos, pueden elegir la canción que desean escuchar, y esta va a sonar sin que los otros conozcan la identidad de la persona que puso la canción.

Esta aplicación está instalada en una amplia gama de bares en la capital del país y al estar sintonizado dentro de esta, se podrá conocer la ubicación de los bares más cercanos que cuentan con esta.

La aplicación cuenta con un límite de canciones por usuario para evitar que uno de estos monopolice la lista. **(*ROJAS Laura,* 2015)**

En 2014 lanzan PlayMyWay, una aplicación gratuita para Web y dispositivos móviles que permiten crear una emisora con el contenido que se desee y las noticias que le interesen al usuario. El objetivo de esta es poder combinar el contenido de interés con las preferencias musicales del usuario, para armar un perfil en base a una lista predeterminada de géneros musicales y temas de información como moda, tecnología o noticias de interés general. Acorde con el objetivo principal se denota el valor agregado, ya que las plataformas más comunes de streaming contienen poco contenido de temas de interés que puedan envolver más al usuario y darle más alcance a una aplicación de este tipo.**(*ROJAS Laura,* 2014)**

Mirolo se destaca como una aplicación WEB lanzada en 2013 gracias al programa apps.co ***(PEÑAREDONDA José Luis***, **2013)**, esta permite acceder remotamente a sitios de entretenimiento público mediante una webcam instalada en este, la localización de esta y el horario de transmisión es definidas por el administrador del sitio, el acceso al streaming se pude tener desde un navegador web en dispositivos móviles y computadores de escritorio.

Siendo usuario se puede tener acceso durante 20 segundos a la transmisión, ya que la finalidad es dar un vistazo rápido del estado del sitio y no violar la privacidad de las personas que asisten a eventos sin conocer que la aplicación da acceso al público en general.

A nivel de Latinoamérica existen empresas con soluciones de streaming genérico de audio y video para diferentes finalidades, como se mencionó anteriormente son de dos tipos, automatizadas y de contacto directo, entre estas se puede encontrar streaminginternacional.com y argentina-streaming.com de Argentina; streamingbrasil.com y delicast.com de Brasil; streaminghd.cl de chile y una gran variedad de empresas a nivel continental que ofrecen las misma soluciones sin que estén enfocadas en las necesidades que tiene los DJs y las audiencias al momento de ofrecer canales de streaming con contenido de interés para aumentar la visibilidad de este y fortalecer su perfil con las audiencias que más desea. En el 2014 Mixify con sede en New York lanzó un servicio popular conocido como clubcast que, permite a los clubes de entretenimiento y al dj un streaming bidireccional, de tal manera que el segundo pueda observar el estado de la fiesta en el club en los que transmite**(*Ihouseu.com,2014*)**.

A mediados del 2012 fue fundada Mixify, una plataforma que permite transmitir audio mixing realizado en vivo y en directo a todas partes del mundo, actualmente esta plataforma es muy famosa y permite abrir un canal que cuenta con diferentes herramientas para facilitar interacciones con el usuario mediante diferentes medios bidireccionales.

El impacto visual y la experiencia de usuario es más enriquecedora, debido a que los mismo son representados por un avatar localizado frente a un escenario que tiene una plataforma en la cual se encuentra la representación del dj y otros paneles para interactuar mediante un chat con otros oyentes, además de contener los links a las redes sociales. Streaming dj nació en 2013 con la finalidad de ofertar DJs en vivo para eventos de música electrónica, vía internet. La aplicación web permite que los usuarios conozcan una lista de DJs junto a su reputación y su género musical, para así poder escoger el que más se adapte a la necesidad de cliente, cuentan con DJs de 15 países del mundo y su servicio de acceso a las plataformas de streaming no están automatizados, requiere contacto directo. **(*CONILL Ernest, 2013)***

BeatBravo es una plataforma fundada en el 2012 con el fin de ofrecer una plataforma gratuita para permitir a los DJs realizar streaming, con el fin de conectarse con productores y fans. BeatBravo utiliza formato mp3 para realizar su transmisión y actualmente tiene una interfaz gráfica muy limpia, pero solo entrega el streaming de audio y el nombre del dj. **(*beatbravo.com*, 2015)**.

Radionomy fue fundada en 2007 ***(crunchbase.com, 2015)*** y tiene como finalidad ofrecer una plataforma con una enorme lista de temas musicales con derechos de autor, para montar emisoras en la cuales se puede hacer streaming tradicional tipo radio o en vivo y en directo, tal como los DJs lo requieran**(*radionomy.com, 2015*)**. La plataforma les permite a los usuarios crear una emisora con una lista de temas con derechos de autor gratuitos, las emisoras deben tener como mínimo una cantidad específica de oyentes para evitar que el canal sea dado de baja, al tener cierta cantidad de oyentes Radionomy premia a sus usuarios con prebendas generadas de la publicidad realizada en su emisora.

En 2007 nace Livestream ***(crunchbase.com, 2015)***, una plataforma que ofrece funcionalidad para que los usuarios transmitan a partir de la cámara web y el micrófono, esta ha sido ampliamente utilizada por los cibernautas para transmitir eventos musicales en vivo y poder sincronizar varias audiencias en el mismo canal, para así generar una experiencia única, con base a compartir festivas u eventos privados con personas de otro lugar del mundo, esta plataforma guía al usuario para realizar la transmisión en su perfil y le ofrece un chat para interactuar con la audiencia.**(*livestream.com, 2015*)**

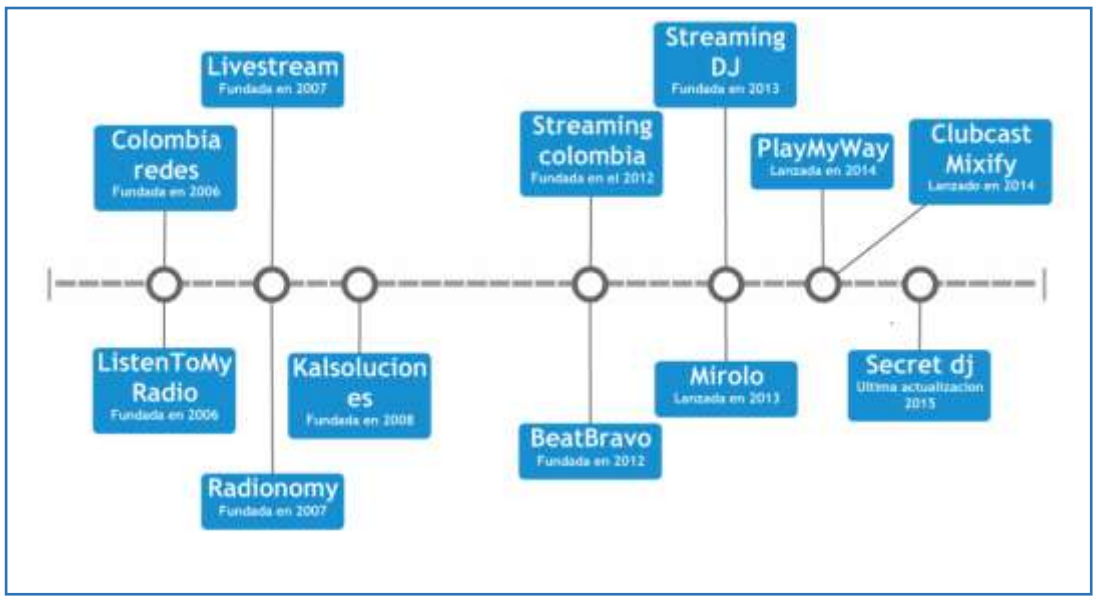
ListenToMyRadio se estableció en 2006 como la primera plataforma gratuita de Shoutcast, el popular servidor de streaming pago.

Hasta el día de hoy sigue siendo la plataforma más grande del mundo, ya que cuenta con alrededor de 1 millón de usuarios que tiene a la mano una robusta plataforma que contribuye a crear fácilmente emisoras online para realizar transmisiones de auto dj y audio mixing en vivo.

Inicialmente su interfaz gráfica no ofrecía una gran experiencia de usuario a los oyentes que deseaban escuchar audio mixing online, debido a que el impacto visual era muy pobre y no se centraban en el desarrollo de este tipo de experiencias, pasado el tiempo y habiendo llegado el auge de los dispositivos móviles y haber aumentado las exigencias de los usuarios por interfaces de gran calidad y diseño en los módulos de interactividad para los canales de transmisiones en vivo, se permitió que los oyentes enriquecieran su sintonía con la comunicación que podía tener con los DJs.**(*listentomyradio.com, 2015*)**

A continuación se puede observar una linea de tiempo del desarrollo del estado del arte.

Figura 7. Línea de tiempo del estado del arte



Fuente: El Autor.

7. METODOLOGÍA

Con la finalidad de obtener un software de streaming con un componente de rendimiento de banda ancha basado en OGG de 64 kbps para audio mixing en vivo, se determina que la investigación es de tipo aplicada y exploratoria, ya que se debe llegar al punto de obtener un producto que implemente un nuevo tipo de transmisión no utilizada en sistemas para DJs, este componente nuevo, deberá explorarse y evaluarse para comprobar que el uso de bajo bitrate puede optimizar el uso de recursos en una transmisión sobre una red estable sin sacrificar la calidad del audio.

7.1. HIPÓTESIS

La implementación de un sistema de streaming mediante una metodología en cascada (análisis, diseño, implementación, pruebas y soporte), basado en formato OGG con una aplicación cliente en Android, optimiza el rendimiento de una plataforma de transmisión en vivo y en directo sin causar una disminución en la calidad, a la vez que es transparente y aumenta el nivel de usabilidad del servicio.

7.2. VARIABLES DEPENDIENTES

- Transparencia en el uso de la aplicación
- Usabilidad de la aplicación
- Rendimiento de la aplicación cliente
- Rendimiento de los servidores
- Disponibilidad del servicio de streaming
- Calidad del contenido transmitido

7.3. VARIABLES INDEPENDIENTES

- Disponibilidad de red de datos de Amazon
- Versiones de plataformas de servirdos Icecast
- Versiones de plataformas de sistema operativo Android
- Red de datos del cliente

7.5. MÉTODO DE INVESTIGACIÓN

Al iniciar las pruebas de comprobación, se testea el software resultante para obtener un conjunto de datos en función del rendimiento, disponibilidad y usabilidad, que medirán las variables cuantitativas, pero independientemente a estas, también se deben generar datos cualitativos para observar las conductas de los usuarios y detectar posibles situaciones.

Se va a utilizar un método de observación debido a que la ruta para comprobar el funcionamiento consiste en, capturar datos en tiempo real del comportamiento del usuario y del sistema, para analizarlos de manera directa y generar conclusiones en base a lo observado en la fase de pruebas.

7.6. POBLACIÓN

DJs de los 18 años en adelante, de Tunja y Bogotá, con conocimientos en audio mixing y manejo de software y hardware para su realización. DJs con tendencia al uso de sistemas de información y redes sociales para soportar su actividad y aumentar su visibilidad mediante estos.

Oyentes de música electrónica entre los 17 y 35 años, de Tunja y Bogotá, con acceso a dispositivos Android de gama media en adelante, con deseos de escuchar y seguir sus DJs locales favoritos.

7.7. ENFOQUE METODOLÓGICO

El sistema presenta dos factores que deben evaluarse, uno en función de la usabilidad del usuario que contendrá variables cualitativas para determinar el nivel de la satisfacción, y facilidad de uso del sistema para garantizar una buena experiencia para los usuarios.

Otro factor que se debe evaluar consta de la relación entre el rendimiento y la disponibilidad, que serán medidos con variables cuantitativas para determinar el uso exacto de recursos utilizados por un servidor con.

La experiencia del usuario está dada por la facilidad con la que el usuario llegue a realizar una actividad en el sistema de manera transparente, y la satisfacción generada por esta, que puede ser medida en función de la satisfacción que obtiene el usuario con el uso de esta, para esto se puede usar indicadores como: el nivel de felicidad causada, la experiencia obtenida, la facilidad de uso y el deseo de usar nuevamente la herramienta.

Teniendo en cuenta que el uso de una herramienta para el usuario es trasparente en la medida que este no sea obligado a realizar una actividad técnica y no detecte ninguna anomalía funcional del sistema en la interfaz gráfica de usuario, reluce una necesidad que obliga a evaluar el diferentes métricas de los servidores del sistema para analizar la disponibilidad y el rendimiento, de tal manera que por estos factores no se vea afectada la aplicación cliente en un entorno de trabajo.

Es importante medir el comportamiento del servidor para observar y analizar las variables de las que dependen estos factores proceder para comprobar que el sistema con la configuración planteada si cumple con lo esperado.

7.8. FUENTES DE LA METODOLOGÍA

GONZALES MANTEIGA Maria Teresa, "Estadísticas aplicadas, una visión instrumental", Ediciones Díaz de Santos, 2012, 786 páginas. 8499691005, 9788499691008.

M. Galetto, A. Romano and S. A. Zayas, Experimentar. Aplicación del método científico a la construcción del conocimiento, Ministerio de Educación, Cultura y Deporte. Área de Educación, 2012, 9788436954029

MOLINA Maria Jose, El Método Científico Global, Molwick Segunda edición: Enero 2013, 81 paginas. 978-84-15328-70-4.

7.9. INSTRUMENTOS PARA MEDIR LA HIPÓTESIS, UTILIZANDO LAS VARIABLES EN LA POBLACIÓN OBJETIVO

- Test de diagnóstico y comprobación de servidores en tiempo real.
- Panel estadístico CloudWatch de Amazon EC2.
- Encuestas y cuestionarios.

7.10. METODOLOGÍA DEL ESTUDIO

La siguiente metodología tiene como finalidad medir la disponibilidad del sistema, mediante la utilización del servicio, para demostrar que el mismo cumple con su objetivo sin tener interrupciones debido al formato o a la plataforma en la nube. Además de lo anterior se medirá el rendimiento de las AMIs mediante las métricas de red y memoria RAM para comprobar que el uso del formato OGG Vorbis utiliza los recursos netamente necesarios para su funcionamiento sin exceder sus especificaciones.

- Proceder con la definición de 6 mountpoint para DJ, 1 en el servidor de Tokio, 1 en el servidor de Frankfurt y 1 en el servidor de SaoPablo.
- Proceder con la definición de entre 3 a 6 clientes, ya sean web o Android para generar consumo en el servicio.
- Proceder con la definición 3 fuentes, para realizar el audio mixing y la transmisión.
- Lanzar los servidores Icecast en las zonas de disponibilidad donde estén instalados.
- Diseño de prueba, posteriormente a lanzar los servidores Icecast, definir los puntos de montajes, las fuentes y los clientes, se debe:
    - Se debe configurar el modulo RTP(Real Transfer Protocol) de cada instancia del software utilizado por las fuentes. Estas instancias pueden ser de: Virtual dj, Traktor o Seratodj.

- Conectar el computador o computadores que posea la instalación del programa fuente, a una banda ancha mínimo de 512kb de subida.
- En la primera parte de la prueba, se debe iniciar la transmisión en los programas fuentes escogidos a un (1) punto de montaje diferente en cada uno de los servidores. Para esto se deberá conocer la dirección IP del servidor que utilizará, el punto de montaje y la contraseña de acceso.
- Posteriormente se desplegarán las aplicaciones clientes en los dispositivos seleccionados y se navegará hasta la lista de canales para escoger el indicado por el director de la prueba.

- Se iniciarán las pruebas de la aplicacion y se observarán las métricas de rendimiento, disponibilidad y transmisión en tiempo real de cada instancia en CloudWatchy en el sistema operativo:
  - Al iniciar estas, se desplegará el panel deCloudWatch.
  - Se configurará un CloudWatch de 3 horas para cada instancia con las siguientes métricas:

    La métrica utilizada para medir el rendimiento es:

    - **CPUUtilization**, para determinar el poder de procesamiento de una instancia seleccionada.

    Las métricas utilizadas para medir el uso de banda ancha en la transmisión son:

    - **NetworkIn**, para conocer el número total de bytes recibidos por las interfaces de red de la instancia.
    - **NetworkOut**, para conocer el número total de bytes enviados por las interfaces de red de la instancia.
    - **NetworkPacketsIn**, utilizada para conocer el número de paquetes de red que ingresan por las interfaces de red de la instancia.
    - **NetworkPacketsOut**, utilizada para conocer el número de paquetes de red enviados por las interfaces de red de la instancia.

    La métrica utilizada para medir la disponibilidad es:

    - **StatusCheckFailed**, informa si algunas de las herramientas de comprobación del estado del sistema falla.***(aws.amazon.com, 2016).***

  - Se ingresará a la terminal del sistema operativo de la instancia EC2 y se ejecutará el comando `sudo ps -aux --sort pmem | grep icecast2`para observar la cantidad de memoria utilizada por el servidor icecast2.

- Se ejecutará el software llamado recortes de Windows 10 y se tomaran pantallazos del comportamiento de las métricas estudiadas y de la memoria RAM.
- Se realizarán capturas de la pantalla en el total de laprueba.
-

- Por último, se observarán los datos registrados en CloudWatch y se realizará una interpretación al respecto.

7.11. METODOLOGÍA DEL DESARROLLO

La metodología del desarrollo del software está definida en cuatro (4) fases de una metodología en cascada, en la cual se realiza, el análisis, el diseño, la implementación, la implantación en servidores de prueba y la realización de las pruebas de cada elemento, a continuación, cada fase con sus respectivas actividades.

En la fase de análisis, se procederá a recolectar la información requerida para obtener conocimiento para realizar los requerimientos respecto a la solución del problema, a continuación, los pasos de la fase:

Tabla 1. Fase de Análisis

ID	Actividad
1	Lluvia de ideas
2	Obtención de conocimiento (encuestas y registros de información)
3	Análisis de requerimientos
4	Realización de diagramas

Fuente: El Autor.

En la fase de diseño, se procederá a utilizar los requerimientos para realizar un diseño de la arquitectura y de las interfaces gráficas del usuario final, a continuación, los pasos de la fase de diseño:

Tabla 2. Fase de Diseño

ID	Actividad
1	Diseño de arquitectura
2	Diseño de wireframes de Aplicación Android
3	Diseño de comps de Aplicación Android
4	Diseño de landing page

Fuente: El Autor.

En la fase de implementación se realizará el desarrollo del sistema en base al diseño y a los requerimientos obtenidos para cumplir con la funcionalidad requerida.

Tabla 3. Fase de Implementación

ID	Actividad
1	Desarrollo de servicio virtual de streaming en AWS EC2
2	Desarrollo de API restfull para gestión de servidores y usuarios
3	Desarrollo de navegabilidad de app Android
4	Desarrollo de módulo HTML5 (Android) para consumo de API restful
5	Desarrollo de reproductor HTML5 para OGG Vorvis en Android
6	Generación de APK
7	Desarrollo de landing page

Fuente: El Autor.

En la fase de pruebas, se va a proceder a implantar y unir cada componente, para realizar las pruebas de la función de cada uno, a continuación, las actividades de la fase.

Tabla 4. Fase de Pruebas

ID	Actividad
1	Instalación de servidor Tomcat
2	Instalación de API restfull en Tomcat
3	Instalación de landing page
4	Instalación de app en landing page
5	Pruebas de rendimiento y funcionalidad de servidores Streaming
6	Pruebas de rendimiento y usabilidad de aplicación Android
7	Pruebas conjuntas de funcionalidad
8	Correcciones de errores simples en APP Android.

Fuente: El Autor.

8. DESARROLLO

En la siguiente sección se encontrará el desarrollo del proyecto basado en los ciclos del software que son requeridos para el cumplimiento del objetivo general y de los específicos.

8.1. ANÁLISIS

En el análisis se encontrarán todos los requerimientos funcionales y no funcionales que deben ser implementados para garantizar la funcionalidad del sistema, además de los casos de uso que se presentan en el sistema.

8.1.2. Requerimientos funcionales

Estos requerimientos hacen referencia a las funciones que debe soportar el sistema respecto a la interacción de este con el usuario.

Tabla 5. Requerimiento módulo de autenticación

ID: RF-01	**Título:** Módulo de autenticación y registro de usuario DJ
Criterios de aceptación: Desarrollar en la página web, un módulo para que DJ pueda registrarse y loguearse con un usuario y contraseña para acceder a las funciones de transmisión. En el registro, el DJ debe especificar: contraseña, región, continente, género musical, mail y nombre artístico.	

Fuente: el Autor.

Tabla 6. Requerimiento servicio web para servicio virtual

ID: RF-02	**Título:** Servicio web para la entrega de información de servicio virtual de streaming.
Criterios de aceptación: Se debe desarrollar un servicio restfull con salida en formato JSON que entregue a las aplicaciones cliente, la información de los canales del servidor Icecast2 más cercano según el continente, incluyendo: la dirección del punto de montaje, el nombre del DJ y el título del tema tocado en tiempo real.	

Fuente: el Autor.

Tabla 7. Requerimiento módulo web de conexión a servidor

ID: RF-03	**Título:** Modulo web que gestione la conexión del DJ a un servidor.
Criterios de aceptación: El sistema debe entregar al DJ los datos de acceso de un punto de montaje con soporte OGG, de un servidor localizado en su continente o región.	

Fuente: el Autor.

Tabla 8. Requerimiento módulo cliente para consumo de API

ID: RF-04	**Título:** Módulo JQuery de consumo de API Restfull en formato JSON
Criterios de aceptación: Se debe desarrollar un módulo HTML5 compatible con phongap para que pueda: • Consultar los puntos de montaje del servicio virtual y la información que este incluye. • Actualizar el tema musical que este tocando el DJ cada 30 segundos.	

Fuente: el Autor.

Tabla 9. Requerimiento reproductor OGG en Android

ID: RF-05	**Título:** Reproductor OGG Vorvis para phongap Android
Criterios de aceptación: Desarrollar un reproductor de streaming con formato OGG Vorvis, que pueda manejar la captura de datos, además de tener las opciones de “play” y “stop”. Esté debe contener la información del punto de montaje que está reproduciendo.	

Fuente: el Autor.

Tabla 10. Requerimiento panel de información de canales

ID: RF-07	**Título:** Panel que recibe la información de un canal y la enseña al usuario
Criterios de aceptación: Es necesario un panel HTML5 que reciba y enseñe la información de un canal de streaming, debe contener: los datos personales del usuario y los datos para la conexión.	

Fuente: el Autor.

Tabla 11. Requerimiento transmisiones

ID: RF-9	**Título:** Transmisiones
Criterios de aceptación: Desarrollar un módulo que muestre al DJ mediante una lista, los canales que se están transmitiendo actualmente	

Fuente: el Autor.

8.1.3. Requerimientos no funcionales

Los requerimientos no funcionales especificarán los componentes y entidades que deben estar presentes en el entorno del sistema para proveer la infraestructura y tecnologías necesarias que soporten la ejecución de las aplicaciones.

Tabla 12. Requerimiento AMI ubuntu en EC2

ID: RNF-01	**Título:** AMI Ubuntu-Icescat2 EC2 de Amazon
Criterios de aceptación: Se debe crear una imagen de máquina de Amazon (AMI) con Icecast pre configurado sobre un sistema operativo Ubuntu.	

Fuente: el Autor.

Tabla 13. Requerimiento montaje de servicio

ID: RNF-02	**Título:** Montaje de servicio preconfigurado de AMIs
Criterios de aceptación: Se debe proceder a crear instancia de la AMI preconfigurada en las zonas de: Sao Pablo, Oregón y Frankfurt.	

Fuente: el Autor.

Tabla 14. Requerimiento configuración AMIs

ID: RNF-03	**Título:** Configuración de AMIs
Criterios de aceptación: Se debe configurar: una (1) IP elástica para cada instancia, el servidor Icecast2 de cada una para determinar los servidores maestros de cada servidor y se debe agregar a la base de datos, el acceso a cada servidor.	

Fuente: el Autor.

Tabla 15. Requerimiento Ubuntu OS

ID: RNF-04	**Título:** Ubuntu OS
Criterios de aceptación: Se utilizará como sistema operativo, Ubuntu versión 12.4 o superior.	

Fuente: el Autor.

Tabla 16. Requerimiento JAVA

ID: RNF-05	**Título:** JAVA
Criterios de aceptación: El lenguaje para realizar los servicios web será JAVA.	

Fuente: el Autor.

Tabla 17. Requerimiento MAVEN

ID: RNF-06	**Título:** MAVEN
Criterios de aceptación: Se utilizará el repositorio Maven 3.0 o superior para cargar las dependencias requeridas para el proyecto.	

Fuente: el Autor.

Tabla 18. Requerimiento RestEasy

ID: RNF-07	**Título:** RestEasy
Criterios de aceptación: Será utilizada la versión 3.0 o superior, como librería para desarrollar los servicios Web.	

Fuente: el Autor.

Tabla 19. Requerimiento Tomcat

ID: RNF-08	**Título:** Tomcat
Criterios de aceptación: Se utilizará Tomcat 8 como servidor web para las aplicaciones y servicios.	

Fuente: el Autor.

Tabla 20. Requerimiento Icecast

ID: RNF-09	**Título:** Icecats
Criterios de aceptación: Se debe utilizar Icecast 2.4 o superior para la creación de la plantilla AMI.	

Fuente: el Autor.

Tabla 21. Requerimiento MySQL

ID: RNF-10	**Título:** MySQL
Criterios de aceptación: El motor de bases de datos que se utilizará será MySQL 5.0 o superior.	

Fuente: el Autor.

Tabla 22. Requerimiento HTML5

ID: RNF-11	**Título:** HTML5
Criterios de aceptación: Las aplicaciones clientes serán construidas en HTML5 y harán peticiones al Servidor con los métodos de JQuery Ajax 2.1 o superior.	

Fuente: el Autor.

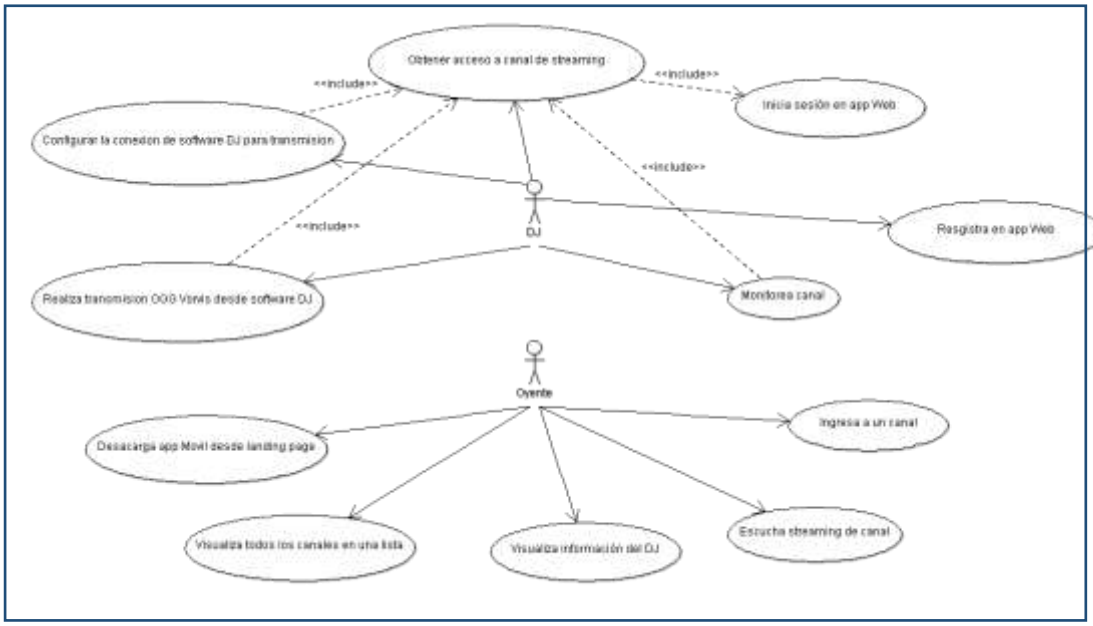
8.2. ESTUDIO DE CASOS DE USO

En esta sección se puede encontrar el análisis gráfico y descriptivo, en donde se puede tener una visión de interacción entre el usuario y el sistema.

8.2.1. Diagrama de casos de uso

A continuación, se puede observar el diagrama de casos de uso para conocer de manera gráfica, las interacciones que van a tener los usuarios con el sistema.

Figura 8. Diagrama de casos de uso



Fuente: El Autor.

8.2.2. Fichas de casos de uso

Ahora se van a conocer las fichas de cada caso de uso con su respectivo identificador, nombre, descripción y otros datos importantes para conocer en detalle la interacción del usuario con el sistema.

Tabla 23. Caso de uso Registra en App Web

Caso de uso	Registra en App Web
Identificador	CDU-01
Resumen	El DJ debe registrarse en el sistema para acceder a las funciones de streaming.
Precondiciones	Acceso a la página del sistema.
Curso básico de eventos	Al ejecutar el navegador, debe dirigirse a la página principal del sistema y buscar el link al formulario de registro.
Postcondiciones	Sus datos personales y de acceso quedan almacenados en la base de datos.

Fuente: El Autor.

Tabla 24. Caso de uso inicio de sesión

Caso de uso	Inicia sesión en App Web
Identificador	CDU-02
Resumen	El DJ puede proceder a identificarse en el sistema para acceder a las funciones de streaming.
Precondiciones	El DJ debe estar registrado en la base de datos junto a un usuario y una contraseña.
Curso básico de eventos	El DJ debe dirigirse al formulario de autenticación de usuario para ingresar su usuario y su contraseña para acceder al sistema.
Postcondiciones	El DJ inicia una sesión que le permite acceder a funciones restringidas.

Fuente: El Autor.

Tabla 25. Caso de uso obtener acceso a canales

Caso de uso	Obtener acceso a canal de streaming
Identificador	CDU-03
Resumen	El DJ debe conocer los datos para conectarse con el servicio de streaming y estos deben ser entregados automáticamente por el sistema cuando el DJ los solicite.
Precondiciones	El DJ debe estar autenticado con un usuario registrado en la base de datos.
Curso básico de eventos	Al autenticarse el DJ, inmediatamente cargará el panel de administración del canal, donde podrá observar su información personal y los datos para realizar una conexión con el servidor de streaming.
Postcondiciones	E DJ ya tiene los datos necesarios para poder realizar la configuración de conexión.

Fuente: El Autor.

Tabla 26. Caso de uso configurar conexión de DJ

Caso de uso	Configurar la conexión de software DJ para transmisión
Identificador	CDU-04
Resumen	El DJ puede dirigirse al panel de configuración de transmisión del software DJ o fuente y entablar la comunicación con el servidor.
Precondiciones	El DJ debe conocer los datos de acceso al servidor.
Curso básico de eventos	El DJ ejecuta el software y posteriormente despliega el panel de configuración para ingresar los datos de conexión. La conexión es validada al terminar de ingresar los datos.
Postcondiciones	El software fuente está configurado para realizar la transmisión de audio OGG Vorvis.

Fuente: El Autor.

Tabla 27. Caso de uso realizar transmisión OGG

Caso de uso	Realizar transmisión OGG Vorvis desde software DJ
Identificador	CDU-05
Resumen	El DJ puede comenzar a transmitir el audio streaming OGG Vorvis al servidor.
Precondiciones	El DJ debe haber configurado y validado la conexión con el servidor.
Curso básico de eventos	El DJ puede proceder a realizar su audio mixing normalmente desde su controlador y este será automáticamente transmitido.
Postcondiciones	El punto de montaje configurado por el DJ, recibe el Stream de audio y puede realizar el Broadcast del mismo a los clientes.

Fuente: El Autor.

Tabla 28. Caso de uso monitorear canal

Caso de uso	Monitorea canal
Identificador	CDU-06
Resumen	El DJ procede a consultar como está el estado de su punto de montaje para saber si efectivamente se realiza la transmisión.
Precondiciones	El DJ debe estar registrado y autenticado para poder ver el estado del canal.
Curso básico de eventos	El DJ ingresa a su panel de usuario y puede conocer el estado de transmisión mediante un(1) indicador.
Postcondiciones	El DJ conoce el estado de transmisión.

Fuente: El Autor.

Tabla 29. Caso de uso descarga app móvil

Caso de uso	Descarga app móvil desde landing page
Identificador	CDU-07
Resumen	El oyente podrá descargar la aplicación móvil desde la página web.
Precondiciones	Ingresar a la página web desde el dispositivo.
Curso básico de eventos	El oyente ingresa a la página web y se dirige al botón para descargar la aplicación.
Postcondiciones	El oyente tiene el apk de la aplicación en su dispositivo y debe poder ejecutarla inmediatamente.

Fuente: El Autor.

Tabla 30. Caso de uso visualiza canales

Caso de uso	Visualiza los canales en una lista
Identificador	CDU-08
Resumen	El oyente puede visualizar en una lista, los canales de transmisión habilitados en el servidor conectado.
Precondiciones	El oyente debe haber definido su continente.
Curso básico de eventos	Después de que el oyente define su continente, se cargara la lista de Canals para que el DJ los visualice.
Postcondiciones	El oyente obtiene acceso a todos los canales de streaming.

Fuente: El Autor.

Tabla 31. Caso de uso ingrese a un canal

Caso de uso	Ingresa a un canal
Identificador	CDU-9
Resumen	El oyente puede escoger un canal de la lista para ingresar al mismo.
Precondiciones	Obtener la lista de canales.
Curso básico de eventos	El oyente escoge e ingresa a un canal de streaming de la lista desplegada.
Postcondiciones	Se encuentra conectado con el canal de streaming.

Fuente: El Autor.

Tabla 32. Caso de uso visualiza información de DJ

Caso de uso	Visualiza información del DJ
Identificador	CDU-10
Resumen	El oyente puede observar los datos del DJ.
Precondiciones	El oyente debe estar conectado al canal.
Curso básico de eventos	El oyente, estando dentro del canal, puede proceder a conocer toda la información del DJ dueño del mismo.
Postcondiciones	El oyente conoce los datos principales del DJ.

Fuente: El Autor.

Tabla 33. Caso de uso escucha streaming del canal

Caso de uso	Escucha streaming del canal
Identificador	CDU-11
Resumen	El oyente puede escuchar el streaming del DJ.
Precondiciones	El oyente debe estar conectado al canal como tal.
Curso básico de eventos	El oyente, estando dentro del canal, puede proceder a escuchar el Stream de audio.
Postcondiciones	El oyente escucha el streaming del DJ.

Fuente: El Autor.

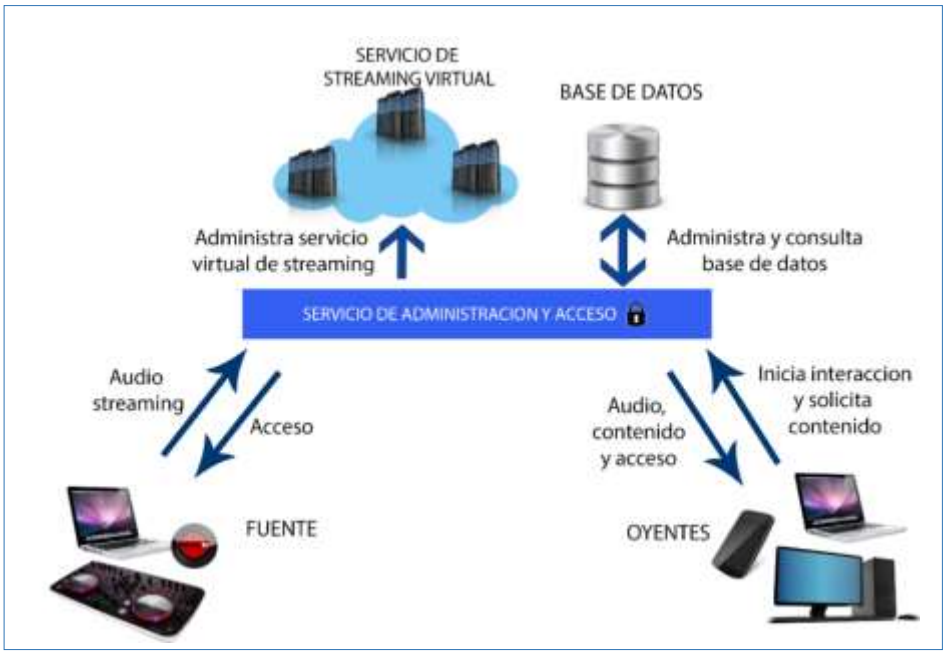
8.3. DISEÑO

A continuación, se van a observar los diagramas e imágenes que componen el diseño del sistema y asi poder tener un entendimiento del mismo.

8.3.1. Diseño estructural.

En el diseño estructural se conocerá el diseño general de las capas del sistema, a continuación, una figura de la arquitectura que puede ser comprendida por un usuario que deseé tener una perspectiva de alto nivel del sistema.

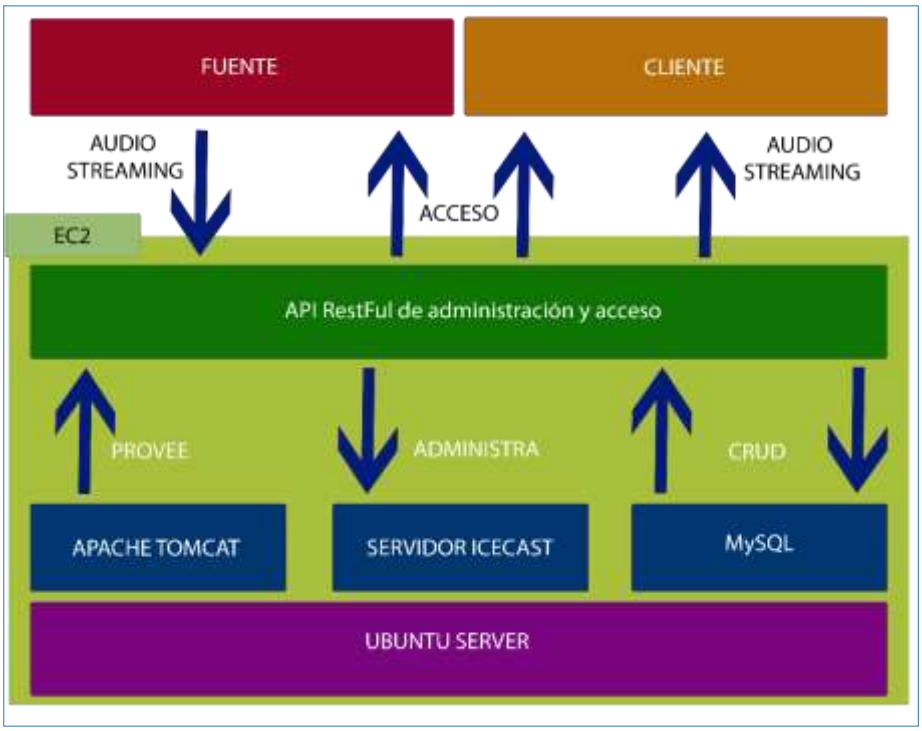
Figura 9. Arquitectura de alto nivel



Fuente: el Autor.

Ahora se podrá conocer el diagrama de capas para poder entender la arquitectura del mismo y las interfaces de comunicación entre componentes.

Figura 10. Diagrama de capas



Fuente: el Autor.

8.3.2. Diseño de interfaces de usuario.

El diseño para interfaces de usuario, da a conocer la interfaz gráfica de la aplicación web y móvil, que permite al usuario interactuar con el sistema.

8.3.2.1. Diseño web.

Inicialmente se conocerá el diseño de la interfaz gráfica de la aplicación web, en la cual se podrá observar como es el diseño de la página principal, en esta se encontrará el panel de log in para acceder al sistema, el formulario de registro, una lista de los canales en línea, una sección para informar del proyecto y por último el link para descargar la aplicación para sistemas operativos Android.

Figura 11. Página de inicio

Fuente: el Autor.

En la siguiente figura se podrá observar el panel que se despliega cuando un DJ ingresa con su usuario y contraseña, en este se encontrará la información de su punto de acceso asignado por el sistema, para realizar la transmisión en vivo, además de otra sección que le informará del estado del punto de montaje, si está o no publicado, junto a otros puntos de montaje.

Figura 12. Panel de DJ

Fuente: el Autor.

8.3.2.2. Diseño de aplicación Android.

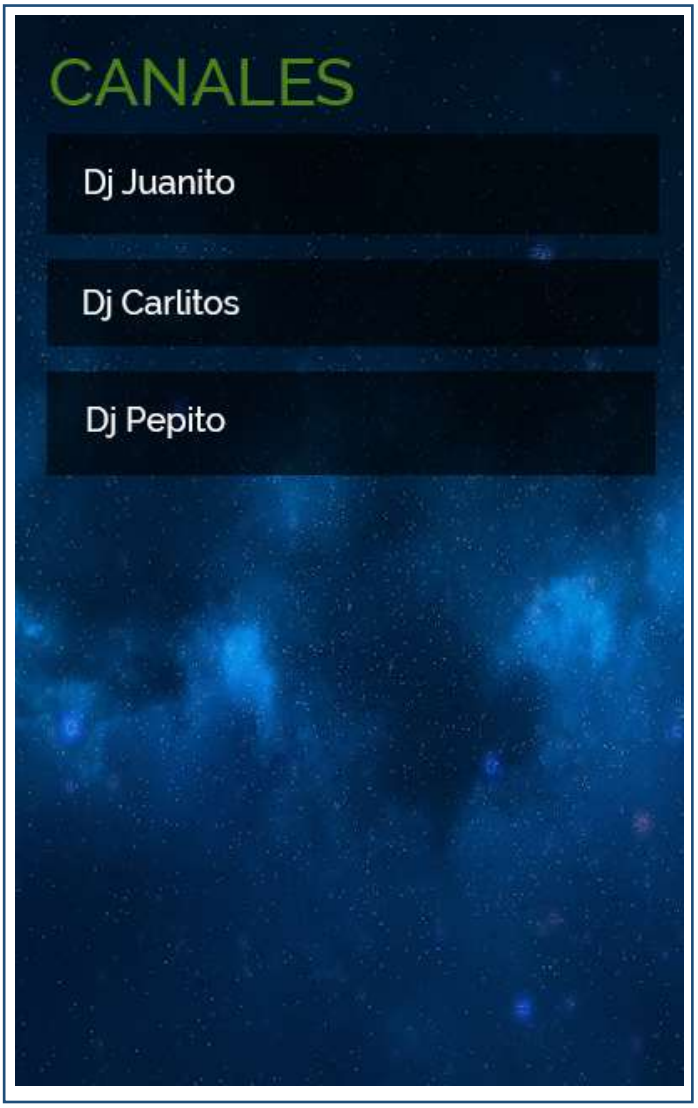
Ahora se puede proceder a ver el diseño de interfaz gráfica de la aplicación móvil para Android, inicialmente se puede observar la página de despliegue que será la que dará la bienvenida a los usuarios oyentes.

Figura 13. Inicio de aplicación.

Fuente: El Autor.

Posteriormente se puede observar la pantalla donde se cargan los canales que están transmitiendo, junto a una opción de actualización que permite recargar los canales.

Figura 14. Listado de canales



Fuente: El Autor.

Por último, la vista del reproductor de audio que se desplegara al escoger un canal. Este contiene, el nombre del dj, un botón para retornar a la lista de dj y un registro que contiene el nombre de las canciones que están siendo reproducidas en tiempo real desde que el oyente inicia la conexión.

Figura 15. Reproducción en curso y lista de temas

Fuente: El Autor.

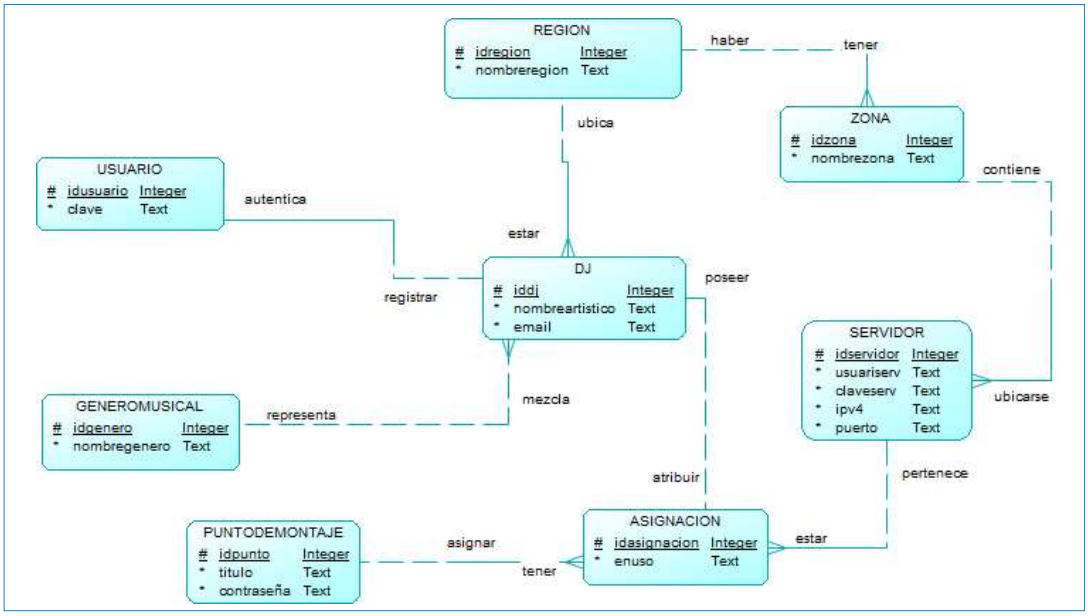
8.3.4. Diseño de datos.

En esta sección se conocerá la narrativa y los modelos que componen el diseño de la base de datos, estos son: el diseño conceptual y diseño lógico.

La narrativa de la base de datos es:

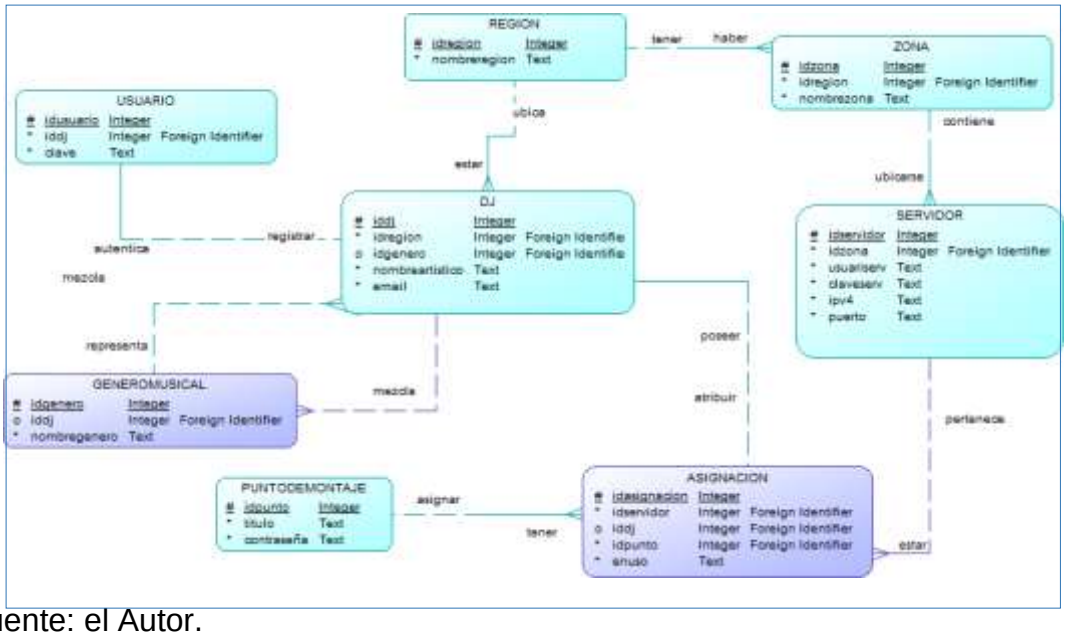
Se necesita una base de datos que permita almacenar la información de un DJ, que contenga: su nombre artístico, país, mail y el nombre de su perfil. Los servidores de streaming deben estar registrados en la base de datos con: su IP, usuario administrador y contraseña de acceso. Además de lo anterior se deben poder registrar los puntos de montaje de cada servidor, junto a: su título, su usuario y su contraseña. Un (1) DJ debe tener un punto de montaje y un punto de montaje debe pertenecer a un servidor. Se debe poder conocer la información del punto de montaje de un dj específico. Deber poder registrar nuevos usuarios del sistema, cada usuario pertenecerá a un DJ. Para dar solución a los requerimientos necesarios se va a iniciar por conocer el modelo conceptual, el modelo lógico y por último el modelo físico.

Figura 16. Modelo conceptual



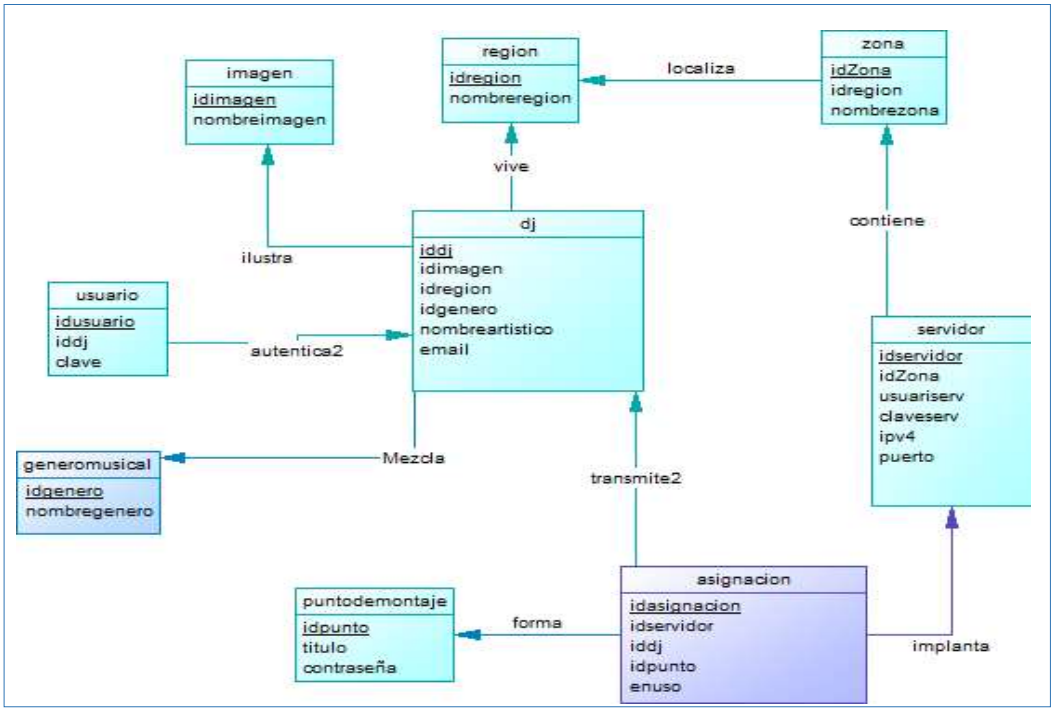
Fuente: el Autor.

Figura 17. Modelo Lógico



Fuente: el Autor.

Figura 18. Modelo Físico



Fuente: el Autor

8.3.5. Diseño procedimental

Por último, se conocerá el modelo procedimental y la interacción de diferentes componentes para lograr una visión del flujo de comportamiento de un usuario, ya sea DJ u oyente sobre el sistema en general.

Figura 19. Modelo procedimental

Fuente: el Autor.

8.4. PROCEDIMIENTO.

En la siguiente sección, se van a conocer los procedimientos desarrollados, que generen los componentes necesarios para cumplir con los requerimientos funcionales y no funcionales del sistema de transmisión en vivo de audio mixing.

8.4.1. Caja de herramientas.

A continuación, las herramientas informáticas, utilizadas para realizar el desarrollo de los componentes:

- Amazon EC2.
- Power Designer 16.5.
- Putty 0.65.
- MySQL Workbench 6.3.
- Netbeans 8.0
- Dia 0.97.
- Argo UML 0.34.
- Brackets 1.5.
- Android Studio.1.5.1.

8.4.2. Requerimientos iniciales.

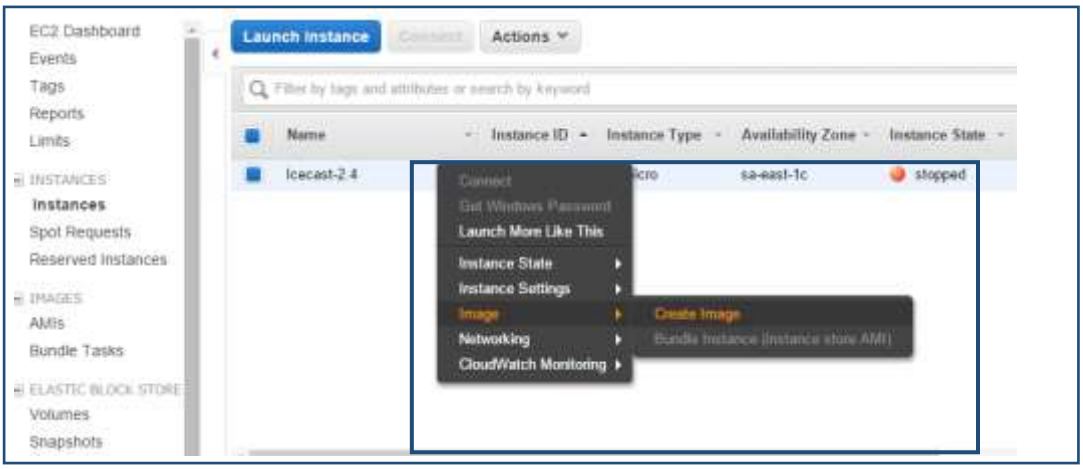
Para poder realizar el desarrollo es necesario contar con una cuenta de Amazon Web Service (AWS), debido a que es necesario tener acceso a las funcionalidades de la plataforma de Elastic Compute Cloud desde ahora mencionada como EC2, para hacer la apertura de la cuenta es necesario tener disponible una tarjeta de crédito.

8.4.3. Creación de AMI personalizada y duplica de sevidores.

Independientemente de la gran oferta de AMIs ofrecidas por particulares y las AMIs predeterminadas que ofrece el servicio EC2 como tal, es posible crear una AMI personalizada que cumpla los requerimientos de back end y permita que el sistema sea escalable en minutos y en función de la demanda del servicio.

Para crear la AMI personalizada de debe utilizar una instancia anteriormente creada y configurada con los servicios que se requieran, en este caso Icecast2 con los 8 puntos de montaje. Para comenzar la creación, se debe dirigir a la opción de “Intances”, dar clic derecho sobre la instancia que se desea reflejar en la AMI y se utiliza la opción “Image->Create Image”.

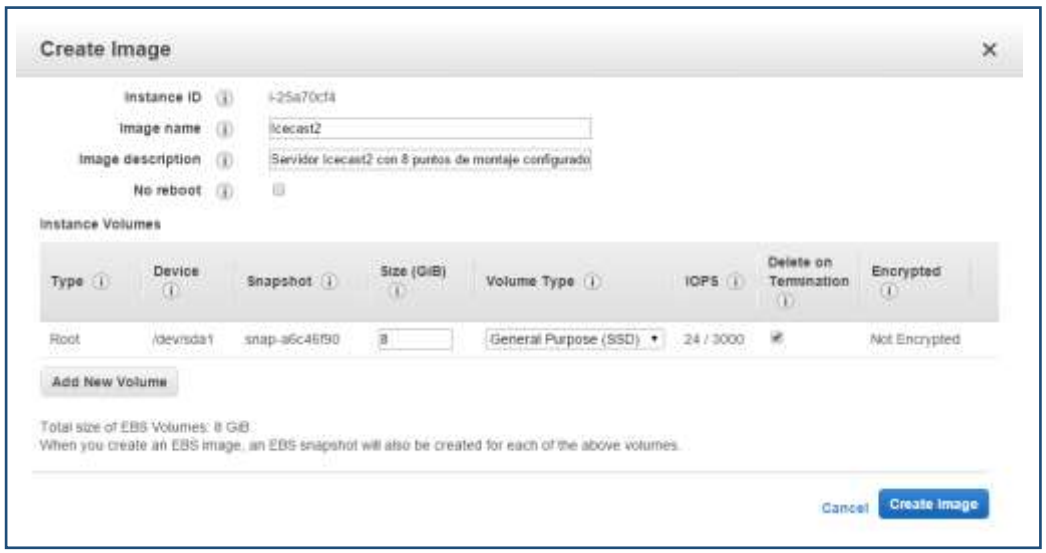
Figura 20. Crear Imagen de Instancia



Fuente: el Autor.

Posteriormente se despliega una ventana para realizar la configuración de la AMI en la cual se solicita un nombre, una descripción y una instancia de almacenamiento, para este caso se utilizara la que anteriormente fue creada.

Figura 21. Configurar imagen de instancia



Fuente: el Autor.

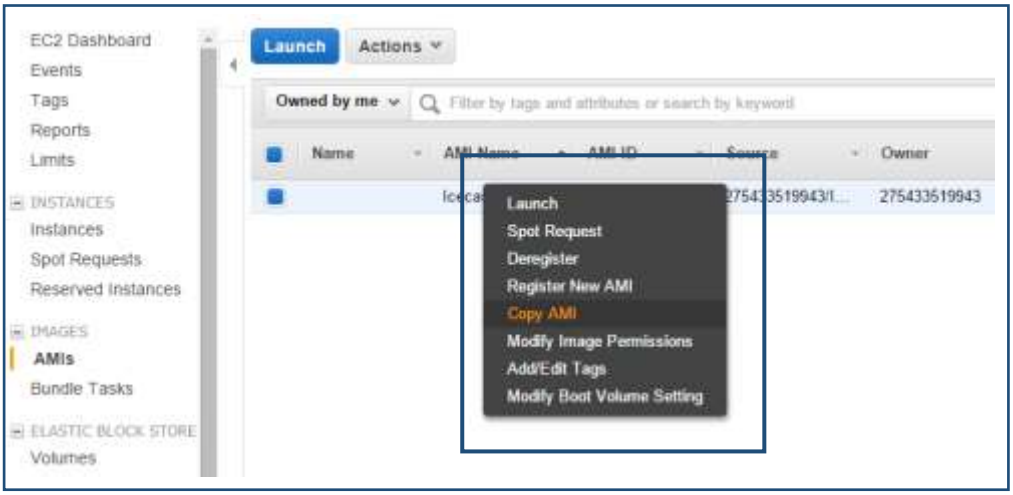
Después de crear la imagen, se desplegará una ventana de confirmación y la AMI estará disponible para realizar los despliegues necesarios, a continuación, se podrá dirigir a la opción AMIs, en donde encontrará la anteriormente creada.

Para distribuir el servicio por varias regiones es necesario duplicar la AMI por las mismas, ya que estas y las instancias pertenecen únicamente a la región sobre la que se crearon.

Para realizar la copia de la AMI, es necesario ir a la opción “AMIs”, en esta se observarán todas las AMIs personalizadas o pertenecientes a la cuenta.

Para hacer la duplica, se da clic derecho sobre la instancia y se utiliza la opción “Copy AMI”.

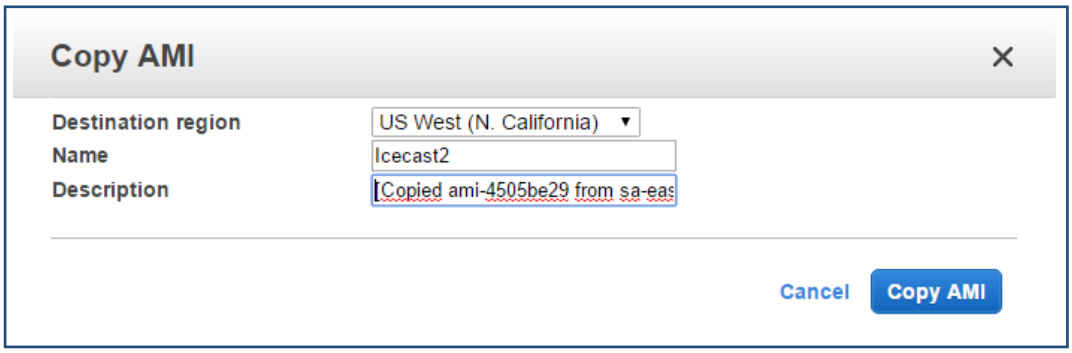
Figura 22. Duplicar AMI en otras regiones



Fuente: el Autor.

Posteriormente, se desplegará una ventana en la cual se debe escoger la región a la cual va ser copiada la AMI, el nombre con el que se observara en la otra región y una breve descripción. Al dar clic en el botón “Copy AMI”, se podrá ir la región objetivo e ingresar en la opción AMIs y crear una instancia como se explica anteriormente, se debe tener en cuenta que el proceso de duplicado tarda unos minutos en completarse satisfactoriamente.

Figura 23. Elegir destino de AMI duplicada



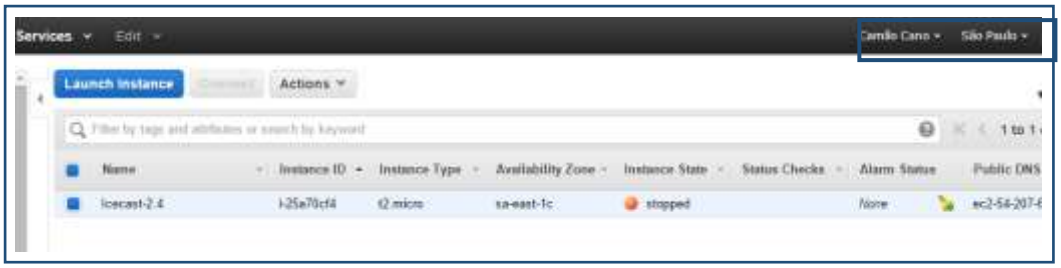
Fuente: el Autor.

Para cumplir con el objetivo del proyecto, se van a utilizar las zonas de:

- Sao pablo (Brasil).
- Frankfurt (Alemania).
- Tokio (Japón).

A continuación, se podrán observar 3 imágenes, en las cuales se demuestra la creación de una instancia con su IP elástica ya asignada, para cada zona especificada.

Figura 24. AMI duplicada en zona de Sao Pablo

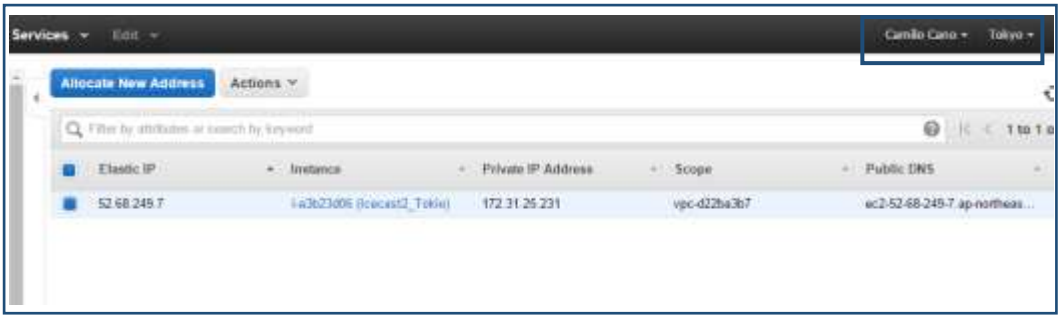


Fuente: el Autor.

Figura 25. AMI duplicada en zona de Frankfurt

Fuente: el Autor.

Figura 26. AMI duplicada en zona de Tokio



Fuente: el Autor.

8.4.4. Creación de base de datos.

De debe recordar que la base de datos tendrá dos propósitos, que son:

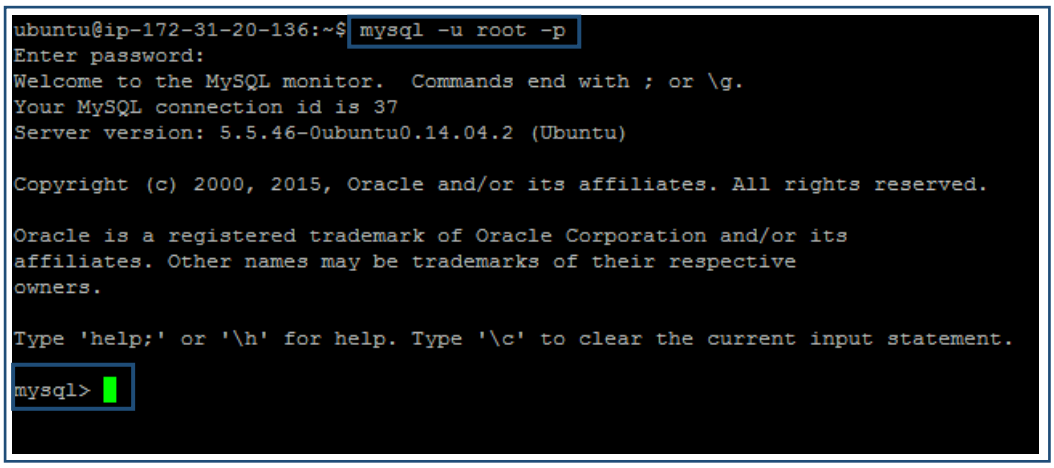
- El almacenamiento de usuarios de las aplicaciones clientes, puntos de montaje y administradores de los servidores, además de la asignación de puntos de montaje a cada usuario.

- El almacenamiento de la información de acceso a los servidores disponibles de streaming en EC2 y los puntos de montaje pertenecientes a cada uno, para permitir la confrontación de puntos de montaje con la base de datos y extraer el DJ correspondiente a cada uno para entregar información en tiempo real.

Teniendo en cuenta la fase de diseño anteriormente registrada, se debe recordar el modelo conceptual, lógico y físico de la base de datos. Más adelante, se va a conocer el contenido de un script inicial que se ejecuta posteriormente a la creación, y es utilizado para configurar correctamente las funcionalidades del sistema en general. Para ejecutar los scripts, se va a utilizar directamente la consola desplegada por la conexión SSH.

Primero se utiliza el comando `mysql -u root -p` para ingresar en la consola de mysql, posteriormente a ejecutar la orden anterior, mysql solicitará la contraseña y por último permitirá el acceso.

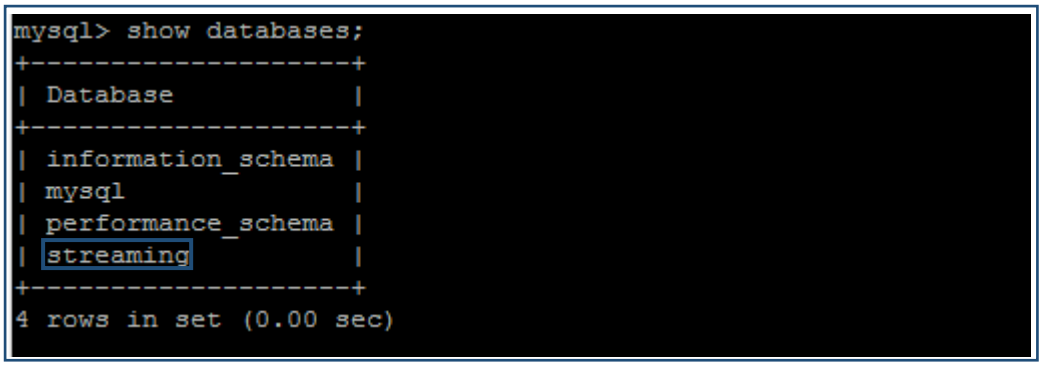
Figura 27. Acceder a consola MySQL



Fuente: el Autor.

Al ingresar a la consola, se puede proceder a crear la base de datos con el comando `create database streaming` y luego ejecutar el comando `show databases` para listas las bases de datos.

Figura 28. Confirmando creación de Base de datos



Fuente: el Autor.

Después de garantizar que la base de datos se creó correctamente, se debe ingresar a esta con `use streaming` y se puede ejecutar los scripts necesarios, a continuación se verá el script de creación:

```
create table asignacion
(
   idasignacion         int not null AUTO_INCREMENT,
   idservidor           int,
   iddj                 int,
   idpunto              int,
   enuso                text not null,
   primary key (idasignacion)
);
create table dj
(
   iddj                 int not null AUTO_INCREMENT,
   idimagen             int,
   idregion             int,
   idgenero             int,
   nombreartistico      varchar(25) not null,
   email                varchar(50) not null,
   primary key (iddj)
);
create table generomusical
(
   idgenero             int not null AUTO_INCREMENT,
   nombregenero         varchar(25) not null,
   primary key (idgenero)
);
```

```
create table imagen
(
   idimagen             int not null AUTO_INCREMENT,
   nombreimagen         varchar(50) not null,
   primary key (idimagen)
);
create table puntodemontaje
(
   idpunto              int not null AUTO_INCREMENT,
titulo               varchar(32),
   contrasena           varchar(32) not null,
primary key (idpunto)
);
create table region
(
   idregion             int not null AUTO_INCREMENT,
   nombreregion         varchar(35) not null,
   primary key (idregion)
);
create table servidor
(
   idservidor           int not null AUTO_INCREMENT,
   idzona               int,
   usuariserv           varchar(25) not null,
   claveserv            varchar(32) not null,
   ipv4                 text not null,
   puerto               text not null,
   primary key (idservidor)
);

create table usuario
(
   idusuario            int not null AUTO_INCREMENT,
   iddj                 int,
   clave                varchar(32) not null,
   primary key (idusuario)
);
create table zona
(
   idzona               int not null AUTO_INCREMENT,
   idregion             int,
   nombrezona           varchar(35) not null,
   primary key (idzona)
);

alter table asignacion add constraint fk_implanta foreign
key (idservidor)references servidor (idservidor) on delete
restrict on update restrict;
```

```
alter table asignacion add constraint fk_transmite2 foreign
key (iddj)references dj (iddj) on delete restrict on update
restrict;

alter table dj add constraint fk_ilustra foreign key
(idimagen)
references imagen (idimagen) on delete restrict on update
restrict;

alter table dj add constraint fk_mezcla foreign key
(idgenero)
references generomusical (idgenero) on delete restrict on
update restrict;

alter table dj add constraint fk_vive foreign key
(idregion)
references region (idregion) on delete restrict on update
restrict;

alter table asignacion add constraint fk_forma foreign key
(idpunto)references puntodemontaje (idpunto) on delete
restrict on update restrict;

alter table servidor add constraint fk_contiene foreign key
(idzona)references zona (idzona) on delete restrict on
update restrict;

alter table usuario add constraint fk_autentica2 foreign
key (iddj)references dj (iddj) on delete restrict on update
restrict;

alter table zona add constraint fk_posee foreign key
(idregion)
references region (idregion) on delete restrict on update
restrict;
```

Recodando que cada servidor tiene 8 puntos de montaje con sus respectivas configuraciones predeterminadas, el script que se observara a continuación, tiene como finalidad crear un Trigger que inserte los 8 puntos de montaje por cada servidor ingresado a la base de datos, de esta manera quedan inmediatamente asociados los puntos de montaje de los servidores del servicio virtual con la base datos y así, poder ser asignados a los DJs que se registren.

```
CREATE TRIGGER servidor_AI_trigger
AFTER INSERT ON servidor
FOR EACH ROW
INSERT INTO asignacion(idservidor,idpunto,enuso)
VALUES(NEW.idservidor,1,'n'),
(NEW.idservidor,2,'n'),
(NEW.idservidor,3,'n'),
(NEW.idservidor,4,'n'),
(NEW.idservidor,5,'n'),
(NEW.idservidor,6,'n'),
(NEW.idservidor,7,'n'),
(NEW.idservidor,8,'n');
```

Los siguientes scripts tienen como finalidad insertar las regiones pertenecientes al servicio EC2.

```
insert into region (nombreregion) values ('América');
insert into region (nombreregion) values ('Europa');
insert into region (nombreregion) values ('Asia');
```

Los siguientes scripts tienen como finalidad insertar las zonas pertenecientes a cada región del servicio EC2 y asociarlos a una región.

```
insert into zona (idregion,nombrezona) values (1,'Sao Pablo');
insert into zona (idregion,nombrezona) values (2,'Frankfurt');
insert into zona (idregion,nombrezona) values (3,'Tokio');
```

Los siguientes scripts tienen como finalidad insertar los 8 puntos de montaje que tendrán todos los servidores en común.

```
insert into puntodemontaje (titulo,contrasena)
values ('mount1','m1source');
insert into puntodemontaje (titulo,contrasena)
values ('mount2','m2source');
insert into puntodemontaje (titulo,contrasena)
values ('mount3','m3source');
insert into puntodemontaje (titulo,contrasena)
values ('mount4','m4source');
insert into puntodemontaje (titulo,contrasena)
values ('mount5','m5source');
insert into puntodemontaje (titulo,contrasena)
values ('mount6','m6source');
insert into puntodemontaje (titulo,contrasena)
values ('mount7','m7source');
insert into puntodemontaje (titulo,contrasena)
values ('mount8','m8source');
```

Los siguientes scripts tienen como finalidad insertar los 3 servidores anteriormente implantados en el servicio de EC2 y asociarlos a su correspondiente zona.

```
insert into servidor
(idzona,usuariserv,claveserv,ipv4,puerto)
values(1,'admin','numidia21','54.207.63.75','8000');
insert into servidor
(idzona,usuariserv,claveserv,ipv4,puerto)
values(2,'admin','numidia21','52.29.112.255','8000');
insert into servidor
(idzona,usuariserv,claveserv,ipv4,puerto)
values(3,'admin','numidia21','52.68.249.7','8000');
```

Los siguientes scripts tienen como finalidad insertar los géneros musicales que podrán ser seleccionados por los DJs.

```
insert into generomusical (nombregenero) values ('Techno');
insert into generomusical (nombregenero) values ('House');
insert into generomusical (nombregenero) values
('Minimal');
insert into generomusical (nombregenero) values ('Deep
House');
insert into generomusical (nombregenero) values
('Crossover');
```

8.4.5. Desarrollo de proyecto web.

Esta sección incluye la explicación del desarrollo de las funciones necesarias para los servicios Restful y la implementación de la funcionalidad del cliente web.

8.4.5.1. Servicio RestFul

Al estar completa la instalación de las tecnologías del back end (Hibernate y RestEasy) en el proyecto Maven, se puede llevar a cabo la implementación de los servicios RestFul que son dos: uno, está encargado de entregar todos los puntos de montaje del servicio virtual y el otro, el título del Stream de un punto de montaje especifico.

Para comprender mejor la línea de implementación de una función es bueno considerar la arquitectura de la aplicación de back end que está dividida por las siguientes capas:

- POJO: clases utilizadas para representar con objetos las estructuras de las bases de datos.
- DAO: métodos que utiliza HQL de hibernate para realizar el CRUD (create, read, update y delete). Tiene como retornos y parámetros objetos mapeados, es decir POJOs.

- SERVICIO: método que recibe parámetros y utiliza los DAOs para realizar una consulta y retornar la información a una vista.
- INTERFACES: son las clases que representan y ofrecen un recurso frente a otros sistemas.

➢ **Servicio Mount Points**

Este servicio es el encargado de entregar los puntos de montaje a las aplicaciones clientes, la primer interacción en el sistema del servicio de administración central es conectarse a los servidores Icecast2 que conoce gracias a los registros en la tabla servidores de la base de datos, posteriormente hace un JOIN entre, la información retornada por los servidores y los registros en la base de datos para determinar en tiempo real el DJ que emite por cada canal.

Para realizar la conexión a los servidores de streaming, se debe hacer una consulta a la base de datos para conocer las IPs, los usuarios y las contraseñas para autenticarse como administrador en cada uno.

Para hacer la consulta, se utiliza un método tipo select mediante la factoría de sesiones de hibernate con HQL en la capa de los DAOS, de manera que retorne una lista de objetos tipo “Servidor”.

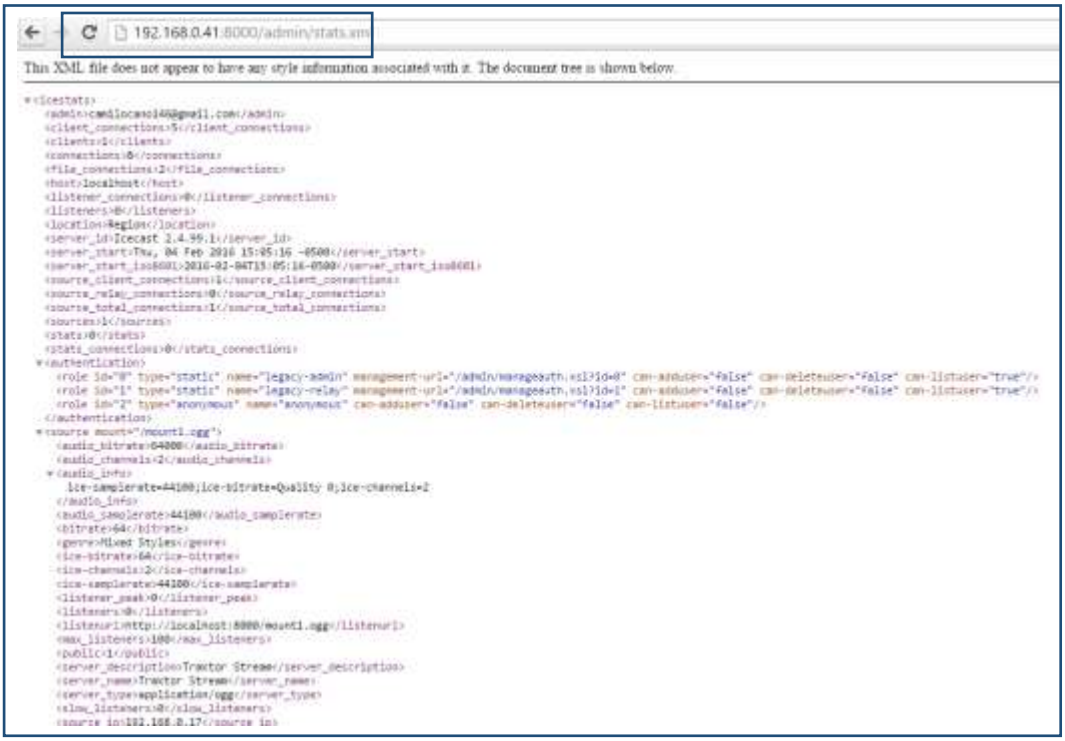
Figura 29. Consulta HQL a objeto Servidor

Fuente: el Autor.

Posteriormente a consultar y obtener la lista de los servidores, se puede proceder a pasar está como parámetro a otra función que, recibe la lista y por cada elemento realiza una nueva conexión al WS de cada servidor Icecast2 mediante autenticación https, para así poder extraer un archivo xml con la información de cada uno y poder procesarlo para obtener los puntos de montaje que funcionan en tiempo real en todos los nodos. Por último, se hace un casting de objetos tipo “Child” que representa la información de las etiquetas de los XMLs para formar una lista de objetos tipo “Mount” con todos los puntos de montaje y su DJ correspondiente, teniendo en cuenta que estén transmitiendo en tiempo real en el servicio virtual.

A continuación, se puede observar el servicio stats.xml de un servidor Icecast en red local que contiene los puntos de montaje del mismo. Este archivo se encuentra generalmente en la dirección “**dominio: 8000/admin/stats.xml”** y para accederlo se requieren credenciales de administrador.

Figura 30. Stats de Icecast en formato XML



Fuente: el Autor.

Inicialmente se puede observar la función para conectar a la url de cada servidor y la función para realizar la autenticación HTTPS.

Figura 31. Conectar y autenticar frente a Icecast

```
public URLConnection openconectionIcecast(String urlCadena) throws IOException {
    URL url = new URL(urlCadena);
    URLConnection con = null;//conexion en general
    try {
        con = url.openConnection();
        return con;
    } catch (MalformedURLException ex) {
        Logger.getLogger(ServicioVirtualStreaming.class.getName()).log(Level.SEVERE, null, ex);
    }
    return con;
}

//autenticacion HTTPS frente al servidor
private void autenticationHTTPSIcecast(final String user, final String pass) {
    Authenticator au = new Authenticator() {
        @Override
        protected PasswordAuthentication
                getPasswordAuthentication() {
            return new PasswordAuthentication //("admin", "camilo".toCharArray());
                    (user, pass.toCharArray());
        }
    };
    Authenticator.setDefault(au);
}
```

Fuente: el Autor.

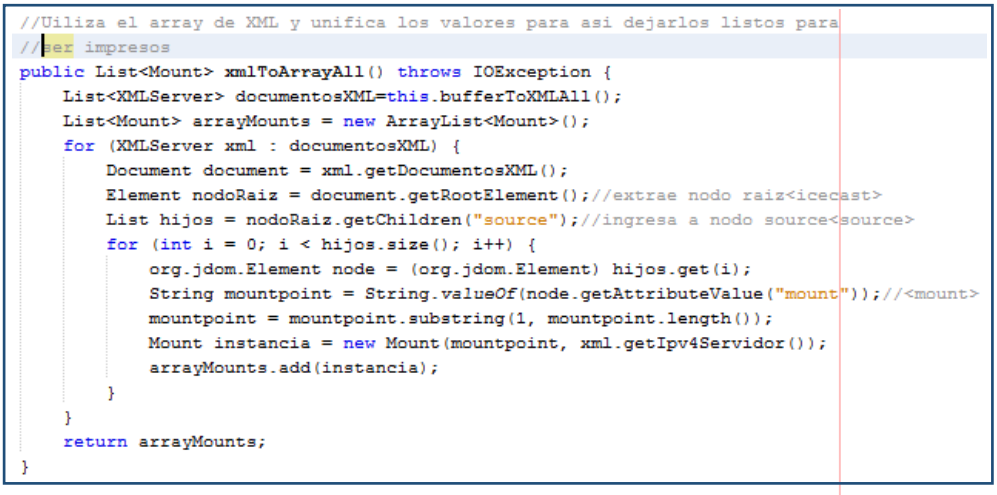
A continuación, se encuentra la función que permite utilizar los métodos anteriores para conectarse a la URL, obtener el stream de datos que retorna la conexión y hacer el casting hacia un objeto “SAXBuilder” que permite el procesamiento de un fichero XML. Para finalizar, el método crea una lista de objetos XMLServer para retornar los XMLs y las IPs a las que pertenecen.

Figura 32. Conversión de buffer a lista de XML

Fuente: el Autor.

Posteriormente se encuentra el método que recibe la lista de objetos “XMLServer”, extrae los XMLs y los procesa para transformar sus registros con la información de los canales en objetos tipo Mount, que serán organizados dentro en una lista.

Figura 33. Conversión XML a arreglo Server



```
//Uiliza el array de XML y unifica los valores para asi dejarlos listos para
//ser impresos
public List<Mount> xmlToArrayAll() throws IOException {
    List<XMLServer> documentosXML=this.bufferToXMLAll();
    List<Mount> arrayMounts = new ArrayList<Mount>();
    for (XMLServer xml : documentosXML) {
        Document document = xml.getDocumentosXML();
        Element nodoRaiz = document.getRootElement();//extrae nodo raiz<icecast>
        List hijos = nodoRaiz.getChildren("source");//ingresa a nodo source<source>
        for (int i = 0; i < hijos.size(); i++) {
            org.jdom.Element node = (org.jdom.Element) hijos.get(i);
            String mountpoint = String.valueOf(node.getAttributeValue("mount"));//<mount>
            mountpoint = mountpoint.substring(1, mountpoint.length());
            Mount instancia = new Mount(mountpoint, xml.getIpv4Servidor());
            arrayMounts.add(instancia);
        }
    }
    return arrayMounts;
}
```

Fuente: el Autor.

Por ultimo ya se puede publicar el servicio mediante una interfaz extraída del método xmlToArray que utiliza la API JABX para serializar los objetos. RestEasy es el encargado de realizar la publicación del recurso y utiliza la notación xml como parámetros, en este caso se puede especificar que el servicio es tipo GET, retorna datos tipo JSON y se podrá acceder con la ruta "**dominio:8080/rest/icecast/mpoints**". Al crear esta interfaz, se está disminuyendo el acoplamiento y se utiliza el método para brindar el servicio, sin exponer los métodos de la clase, el método padre es el servicioAdministrativo.

Figura 34. Interfaz de servicio administrativo

```
@Path("/icecast")
public interface InterfazDeServicioAdministrativo {

    @GET
    @Path(value = "/mpoints")
    @Produces(MediaType.APPLICATION_JSON)
    List<Mount> servicioAdministracion() throws IOException;
```

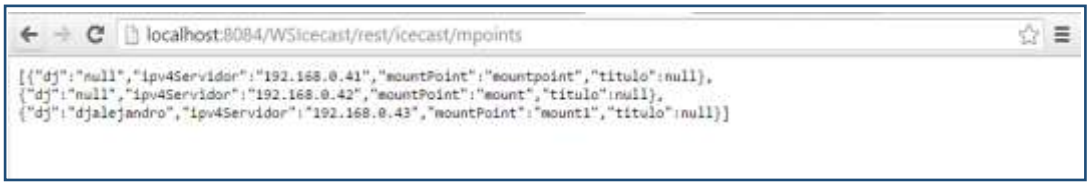
Fuente: el Autor.

A continuación, se mencionarán las etiquetas y los elementos JAXB agregados para imprimir el servicio, estos son:

- @GET, esta etiqueta define que la petición realizada por el cliente para usar este recurso, debe ser tipo GET.
- @Path(value = "/mpoints"), esta define la ruta del recurso y los parámetros que se enviaran en el método GET del cliente.
- @Produces(MediaType.*APPLICATION_JSON*), esta especifica el formato en el cual se imprimirán los datos.
- @PathParam, esta se utiliza para pasar los parámetros del servicio web al método del recurso como tal.

A continuación, se podrá observar cómo se imprime el servicio restful en un navegador que muestra la información de los puntos de montaje, ya que este soporta protocolo HTTP.

Figura 35. Servicio Mount Points en navegador



Fuente: el Autor.

- **Servicio Título Mount Point**

Este es el encargado de entregar el título del stream de un punto de montaje específico cuando una de las aplicaciones cliente este reproduciendo un canal y haga la petición. Es decir, se encarga de entregar el título de la canción que este escuchando el oyente para que este pueda conocer esa información.

Este método cuenta con una estructura similar al anterior servicio web, con la diferencia que esté realiza la conexión a un solo servidor debido a que necesita conocer el título de un solo punto de montaje.

Para iniciar, se puede conocer el método que realiza la conexión a un solo servidor. Es necesario tener en cuenta que esta función también utiliza los métodos de conexión a URL y de autenticación HTTPS que utiliza el método de Web Service anterior.

Al hacer la conexión a un solo servidor se extrae el stats.xml, hace casting a un objeto SAXBuilder, y estructura un solo XML, a diferencia del anterior WS que retorna una lista de objetos, este retorna solo un objeto.

Figura 36. Conversión buffer a un XML

```
//funcion que permite conectarse a un servidor y extrer los puntos de montaje
//para hacerles cast a un objeto XML y poder procesarlo
private Document bufferToXML() throws IOException {
    List<Mount> array = new ArrayList<Mount>();
    this.autenticationHTTPSIcecast("admin", "numidia21");//hace autenticacion
    URLConnection conexion = this.openconectionIcecast("http://"+this.ipv4Servidor+":8000/admin/stats.xml");
    BufferedReader in = new BufferedReader(new InputStreamReader(conexion.getInputStream()));//extrae flujo
    String linea;
    String xml = "";
    while ((linea = in.readLine()) != null) {
        xml = xml + linea;//lee flujo y crea String con lineas del XML
    }
    SAXBuilder constructorSAX = new SAXBuilder();
    Document xmlConstruido = null;
    try {
        xmlConstruido = constructorSAX.build(new StringReader(xml));//pasa string a documento XML
    } catch (JDOMException ex) {
        Logger.getLogger(ServicioVirtualStreaming.class.getName()).log(Level.SEVERE, null, ex);
    }
    return xmlConstruido;
}
```

Fuente: el Autor.

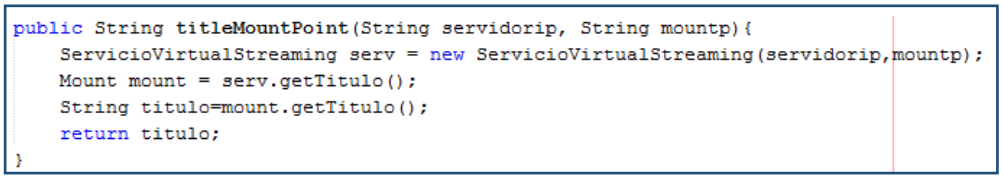
Para continuar se encuentra el método que permite procesar el XML y extraer el valor del título.

Figura 37. Método para obtener título de mount point

Fuente: el Autor.

Para continuar se encuentra el método de la capa de servicio que hace uso del método getTitulo.

Figura 38. Método para usar get titulo



Fuente: el Autor.

Para finalizar se encuentra la interfaz extraída de titleMountPoint que permite la impresión del servicio. Una de las diferencias de los servicios radica en que este utiliza dos parámetros, el punto de montaje y el servidor donde se encuentra, de esta manera el servicio sabe en donde consultar.

Figura 39. Interfaz de servicio web get titulo

Fuente: el Autor.

A continuación, se puede ver un ejemplo de cómo se imprime el servicio en un navegador, el cual recibe la dirección del servicio web por URL y mediante esta se pasan los parámetros, como se muestra en el ejemplo que se verá a continuación con un servidor local, “localhost:8080/rest/icecast/titulompoint/ 192.168.0.1/mountpoint.ogg”.

Figura 40. Método get título en navegador

Fuente: el Autor.

8.4.5.2. Cliente Web.

Esta aplicación HTML5 funciona de manera asíncrona, debido a que la comunicación con el back end se hace con la implementación AJAX de Jquery. Está aplicación está dirigida especialmente a los DJs, debido a que es el medio por el cual, se obtiene acceso a un servidor en la región deseada.

Recordando la interfaz especificada en el proceso de diseño y teniendo en cuenta su funcionalidad como tal, se explicará cómo se debe realizar la implementación de sus funcionalidades, que son:

1. Abrir un nuevo canal para DJ.
2. Ingresar y observar los datos para acceder al canal.
3. Monitorear los canales en operación.

- **Abrir un nuevo canal para DJ.**

Para proceder a obtener un canal, el DJ debe diligenciar el formulario presentado junto a todos los datos que en este se especifican.

Figura 41. Formulario de registro DJ

Fuente: el Autor.

Posteriormente a diligenciar el formulario y dar clic en el botón registrar, Jquery captura los datos de los inputs text y los selects mediante el identificador en el DOM, posteriormente se utiliza la función POST para el envió de los datos, en la cual se especifica la ruta del Servlet a donde se dirige y los parámetros que envía. A continuación, el código de esta implementación:

Figura 42. Método cliente para insertar DJ

Fuente: el Autor.

Como se puede observar en la imagen de la implementación de la función Jquery anterior, el servlet llamado es ServIDj y su función es recibir la información del formulario de inscripción del DJ y utilizar el dao DaoDj para realizar la inserción en la base de datos. A continuación, la implementación del servlet:

Figura 43. Captura de parámetros de formulario DJ

```
try {
    String nombreArtistico = request.getParameter("nombreArtistico");
    String email = request.getParameter("email");
    int regioParam = Integer.parseInt(request.getParameter("region"));
    int imagenParam = 1;
    int generoParam = Integer.parseInt(request.getParameter("genero"));
    String clave = request.getParameter("clave");
    DaoDj daodj = new DaoDj();
    DaoUser daouser = new DaoUser();
    DaoAsignacion daoasignacion = new DaoAsignacion();
    Imagen imagen = new Imagen();
    imagen.setIdimagen(imagenParam);
    Region region = new Region();
    region.setIdregion(regioParam);
    Generomusical genero = new Generomusical();
    genero.setIdgenero(generoParam);
    Dj dj = new Dj();
    int idDj = daodj.insertarDj(imagen, region, genero, nombreArtistico, email);
    if (idDj > 0) {
        dj.setIddj(idDj);
        if (daouser.insertarUsuario(dj, clave)) {
            int idAsignacion = daoasignacion.seleccionarAsignacionDisponible(regioParam);
            if (daoasignacion.actualizarAsignacion(idDj, idAsignacion)) {
                out.print("Registro exitoso, por favor inicie sesión");
            }else{
                out.print("Registro fallido");
            }
        }else{
            out.print("Registro fallido");
        }
    } else {
        out.print("Registro fallido");
    }
} finally {
    out.close();
}
```

Fuente: el Autor.

Por último, se puede observar el dao DaoDj que es el encargado de utilizar la función save de la factoría de sesiones de Hibernate para hacer la inserción del DJ a la base de datos, a continuación, su implementación:

Figura 44. Método backend para insertar DJ

```
public int insertarDj(Imagen imagen, Region region, Generomusical generomu, String nombreartistico, String email){
    SessionFactory sf = HibernateUtil.getSessionFactory();//crea una instancia de la factoria de sesiones
    Session session= sf.openSession();//crea un objeto para abrir la session
    session.beginTransaction();
    int id=0;
    String respuesta;
    Dj dj = new Dj();
    dj.setImagen(imagen);
    dj.setRegion(region);
    dj.setGeneromusical(generomu);
    dj.setNombreartistico(nombreartistico);
    dj.setEmail(email);
    try{
        id =(Integer)session.save(dj);
        session.getTransaction().commit();
        session.close();
    }catch(Exception e){
        id=0;
    }

    return id;
}
```

Fuente: el Autor.

- **Ingresar y observar los datos para acceder al canal.**

Luego de recibir una notificación de inserción exitosa por parte del sistema, se puede proceder a realizar el log in en la sección superior con el mail y la contraseña especificada.

Figura 45. Formulario de Log In

Fuente: el Autor.

Luego de hacer el log in, se desplegará la información retornada en la misma capa, anteriormente utilizada por el formulario del registro.

Figura 46. Datos de mount point en app web

Fuente: el Autor.

Los valores del mail y la clave son capturados mediante el DOM con Jquery e igualmente utiliza la función POST para enviar los parámetros a un servlet y asi recibir la información requerida por el DJ y poder realizar la transmisión.

Figura 47. Función cliente para Log In

```
function logIn() {
    usuario = $('#txtMailLog').val();
    clave = $('#txtClaveLog').val();
    if (usuario == "" || clave == "") {
        alertify.error("Por favor digite el usuario y la clave");
    } else {
        $.post('ServLogIn', {usuario: usuario, clave: clave},
        function (respuesta) {
            if(respuesta =="No esta registrado"){
               alertify.success(respuesta);
            }else{
                loadPanelDj(respuesta);
            }
        }).error(function () {
            alertify.log('No hay conexión');
        }
        );
    }
}
```

Fuente: el Autor.

Como se observa en el parámetro de la función POST, este hace llamado a ServLogIn, que recibe los parámetros, los valida mediante el dao DaoUser y retorna los datos en formato JSON al cliente web.

Figura 48. Captura de parámetros para formulario Log In

```
try {
    DaoUser daousuario = new DaoUser();
    String usuario = request.getParameter("usuario");
    String clave = request.getParameter("clave");
    List data = daousuario.logInEstructurado(usuario, clave);
    Iterator iterator = data.iterator();
    HashMap map = new HashMap();
    if (iterator.hasNext()) {
        Object[] pair = (Object[]) iterator.next();
        map.put("ipv4", pair[0]);
        map.put("puerto", pair[1]);
        map.put("mountpoint", pair[2]);
        map.put("clave", pair[3]);
        out.println((JSONObject) JSONSerializer.toJSON(map));
    }else{
        out.print("No esta registrado");
    }
} finally {
    out.close();
}
```

Fuente: el Autor.

La función utilizada por el DaoUser es LogInEstructurado, este realiza una consulta a la base de datos, teniendo como paramento el mail y la contraseña, si este usuario existe en la base de datos es porque anteriormente el Dj había realizado el registro y ya tenía asignado un punto de montaje, el cual puede ser retornado. A continuación, la implementación.

Figura 49. Método Log In de back end

```
public List logInEstructurado(String user, String clave){
    SessionFactory sf = HibernateUtil.getSessionFactory();//crea una instancia de la factoria de sesiones
    Session session = sf.openSession();//crea un objeto para abrir la session
    session.beginTransaction();
    String query = "select S.ipv4, S.puerto, P.titulo, P.contrasena "
            + "from Dj D, Asignacion A, Puntodemontaje P, Servidor S, Usuario U "
            + "where D.iddj=A.dj "
            + "and D.iddj=U.dj "
            + "and A.puntodemontaje=P.idpunto "
            + "and A.servidor=S.idservidor "
            + "and U.email=:email "
            + "and U.clave=:clave";
    Query q = session.createQuery(query);
    q.setParameter("email", user);
    q.setParameter("clave", clave);
    List resultList = q.list();
    return resultList;
}
```

Fuente: el Autor

Después de que el Log in es exitoso y el back entrega los datos solicitados al front end, estos son mostrados mediante otra función Jquery que los añade a la capa especificada.

Figura 50. Método callback para Log In

```
function loadPanelDj(respuesta){
    var obj = jQuery.parseJSON(respuesta);
    alertify.success(obj.ipv4+", "+obj.mountpoint);
    var contenidoUsuario=$('#contenidoUsuario');
    contenidoUsuario.empty();
    $('#tituloRegistro').text("Datos de acceso");
    var contLista=$('<div/>', {
        'id':'datosDeacceso'
    });
    var listaDatos = $('<ul>/');
        listaDatos.append("<li>Servidor: "+obj.ipv4+"</li>");
        listaDatos.append("<li>Puerto: "+obj.puerto+"</li>");
        listaDatos.append("<li>Punto de montaje: "+obj.mountpoint+".ogg</li>");
        listaDatos.append("<li>Clave: "+obj.clave+"</li>");
    contLista.append(listaDatos);
    contenidoUsuario.append(contLista);
}
```

Fuente: el Autor.

- **Monitorear los canales en operación.**

Esta funcionalidad, permite a los DJs monitorear su canal para que conozcan si su trasmisión se está realizando de manera exitosa o no y garantizar que los oyentes pueden accederla.

Figura 51. Canal en panel de monitoreo

Fuente: el Autor.

Los procedimientos de la función consisten en, conectarse con el servicio web de Mount Points mediante la función AJAX de Jquery y hacer la petición de los canales que están transmitiendo.

Figura 52. Metodo Ajax para obtener lista de canales

```
function getMountPoints() {
    jQuery.ajax({
        type: "GET",
        url: "http://54.207.63.75:8080/wsicecast/rest/icecast/mpoints/",
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        success: function (data, status, jqXHR) {
            estructuraCanalaes(data);
        },
        error: function (jqXHR, status) {
            alert('No hay canales transmitiendo!!!');
        }
    });
}
```

Fuente: el Autor.

Después de obtener la lista de canales, otra función es la encargada de mostrarlos organizadamente para que el DJ pueda saber si su canal está o no en transmisión. La implementación de esta es:

Figura 53. Función que lista canales en front end

```
function estructuraCanalaes(data) {
    var listaCanales=$(".listaCanales");
    listaCanales.empty();
    var canal;
    $.each(data, function (i, item) {
        canal=("<div class='canal col-xs-10 col-sm-11' mount='"+item.ipv4Servidor+
                ":8000/"+item.mountPoint+".ogg'"+"dj='"+item.dj+"'>"+item.dj+
                "</div><div class='canal col-xs-2 col-sm-1' mount='"+item.ipv4Servidor+
                ":8000/"+item.mountPoint+".ogg'"+"dj='"+item.dj+"'><div class='iconoPlay'></div></div>");
        listaCanales.append(canal);
    });
}
```

Fuente: el Autor.

8.4.6. Implementación y configuración de tecnologías para proyecto app móvil.

Para desarrollar el proyecto HTLM5 se va a utilizar como base el conocido framework Apache Cordova y las librerías nativas OGG vorbis para soportar la codificación del stream.

8.4.7. Desarrollo de aplicación android.

En esta sección se va a proceder a hacer la descripción de la implementación de los diferentes componentes necesarios para cumplir con los requerimientos de la aplicación, de modo que sea funcional y cumpla con el objetivo planteado. Se registrará la adición de los plugins necesarios, el desarrollo de la capa HTML5, la capa nativa y la manera como estos envían mensajes entre sí.

8.4.7.1. Creación de proyecto Apache Cordova.

Para iniciar con el desarrollo de la aplicación, se debe proceder a lanzar un proyecto con Apache Cordova mediante consola, el comando utilizado para esta finalidad es: `cordova create streamingphongap com.tg.StreamOgg`. Primero se procede a especificar la utilización de Cordova, posteriormente el segundo parámetro define la función create para crear un nuevo proyecto, el tercer parámetro es el nombre del proyecto y por último el dominio De la aplicación.

Después de crear el proyecto, se puede proceder a agregar las plataformas en las cuales se implementará la aplicación, en este caso Android, para agregarla se debe usar el comando: `cordova platform add Android`.

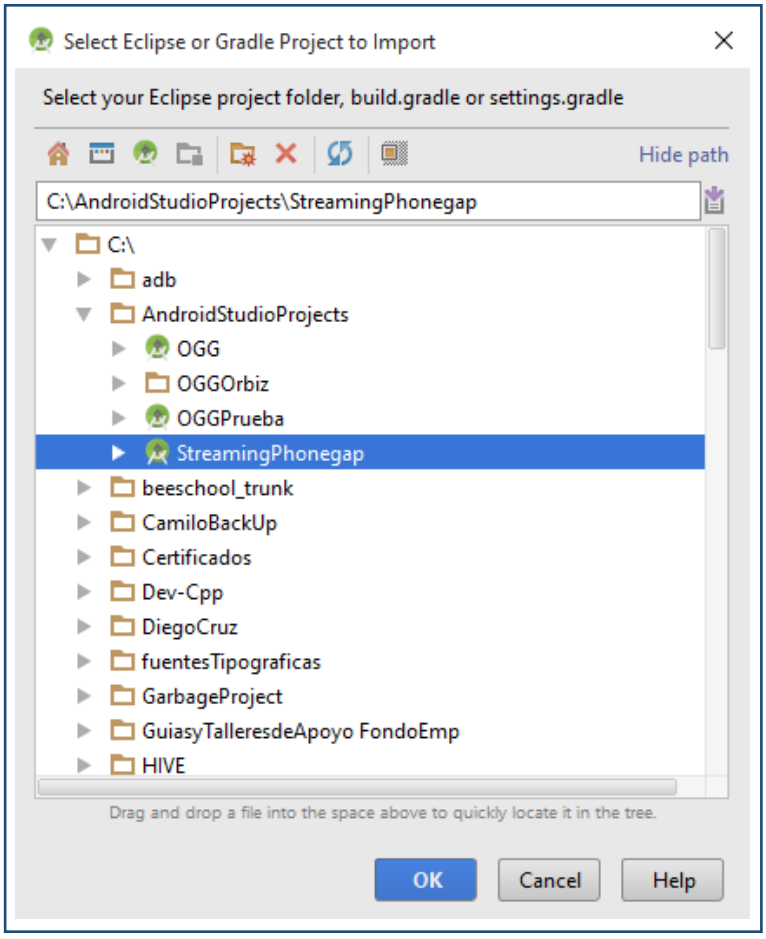
Según los requerimientos de la aplicación, se debe agregar el plugin de splasscreen para poder predeterminar la imagen que se presentar al cargar la aplicación, este se agrega mediante el comando: `cordova plugin add cordova-plugin-splashscreen`.

Por último, se utiliza el comando: `cordova build` para construir el proyecto y así crear el esqueleto HTML5 que contiene HTML, CSS y JS **(Apachecordova.com, 2015)**.

8.4.7.2. Apertura de proyecto en Android Studio

Después de crear el proyecto con cordova y construirlo, se puede proceder a abrir Android Studio y dirigirse a la opción File->New->Import Proyect, seleccionar la ruta del proyecto y oprimir “OK”.

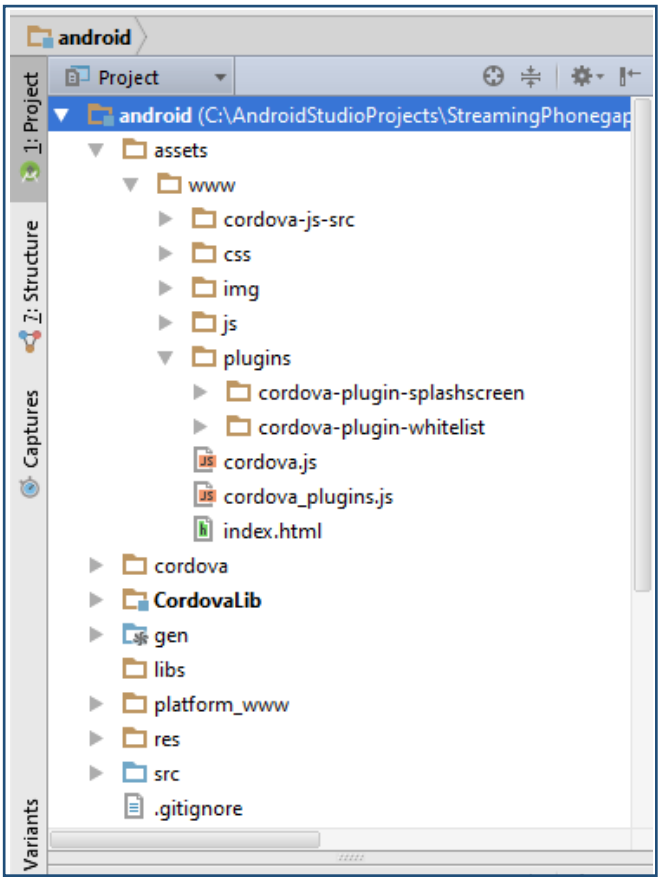
Figura 54. Seleccionar proyecto en Android Studio



Fuente: el Autor.

Ahora se puede proceder a observar el proyecto con el index.html y el index.js, además de los plugins instalados como el splashscreen para la pantalla de inicio y el whitelist para permitir acceso a dominios externos.

Figura 55. Árbol de proyecto en Android Studio

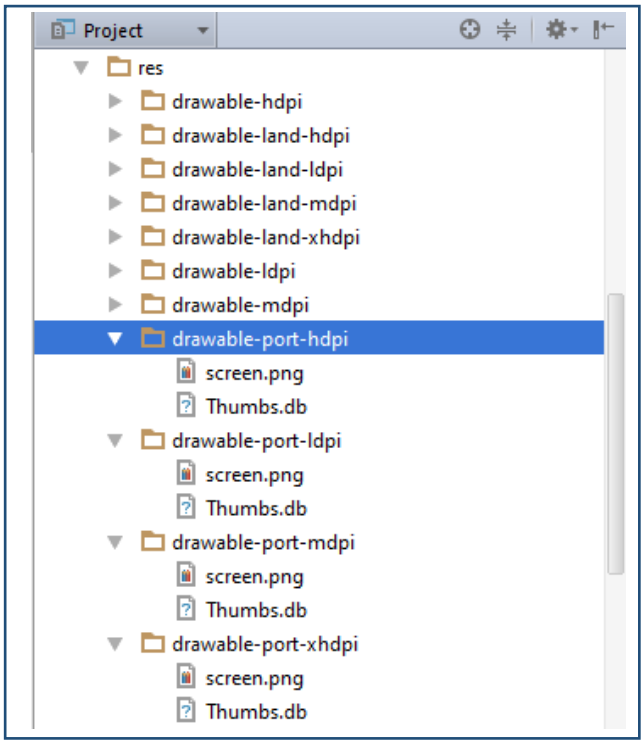


Fuente: el Autor.

8.4.7.3. Definir imagen de pantalla inicial.

Para definir las imagenes del inicio de pantalla, se debe proceder a cambiarlas en las carpetas con prefijo “drawable port” ya que la aplicación debe ser bloqueada en orientación vertical desde el Android Manifest. Para garantizar que la imagen sea mostrada en la mayor cantidad de dispositivos móviles, se debe cambiar todos únicamente las pertenecientes a los tamaños de los port que se ven desplegados en la siguiente imagen.

Figura 56. Ficheros de Splash Screen

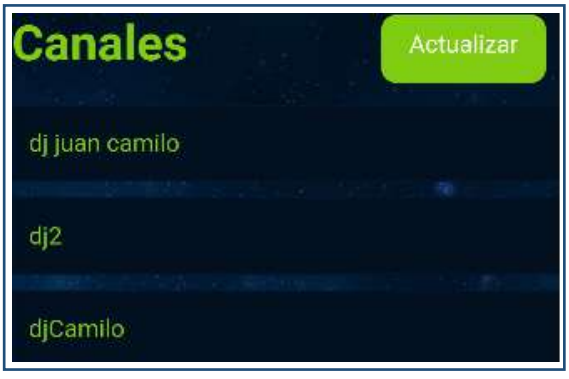


Fuente: el Autor.

8.4.7.4. Sección de carga de canales de streaming.

Recordando el segundo frame planteado en el diseño, el cual especifica que la aplicación debe poder listar los canales de streamig para que el usuario pueda seleccionar uno, se va a conocer la función Jquery que se conecta al servicio web que entrega todos los puntos de montaje y los lista en una capa HTML.

Figura 57. Lista de canales en app Móvil



Fuente: el Autor.

Es necesario tener en cuenta que la función usada es la misma que se utiliza anteriormente para listar los canales en la aplicación web, a diferencia que esta se conecta a la IP elástica de la máquina virtual donde se ejecuta el servidor Tomcat, la función es:

Figura 58. Método para obtener canales desde app móvil

```
var lastTheme="";
function listenTitleMountP(servidor,mountp){
    jQuery.ajax({
          type: "GET",
          url: "http://54.207.63.75:8080/wsicecast/rest/icecast/titulompoint/"+servidor+"/"+mountp,
          dataType: "text",
          success: function (data, status, jqXHR) {
          newTheme=data;
          if(newTheme==lastTheme){
          }else{
            lastTheme=newTheme;
            $('#titstreamtext').append(newTheme+"<br><br>");
          }

          },
          error: function (jqXHR, status) {
                          alert('error');
          }
    });
}
```

Fuente: el Autor.

Como se puede observar en el script, esté lista los canales en una capa con identificador titstreamtext que se encuentra en el index.html, el cual tiene la siguiente estructura en su cuerpo:

Figura 59. Estructura de la vista de app Móvil

```
<body>
    <div id="contenedor">
            <div id="cabeceraCanales">
                <div id="titCanales">
                    <h1>Canales</h1>
                </div>
                <div id="btnActualizarLista">
                    Actualizar
                </div>
            </div>
            <div id="listaCanales">

            </div>
        <div id="reproductor">
            <div class="acciones">
                <div id="btnVolver">
                    <div class="flechaBtnVolver">&larr;</div>
                </div>
            </div>
            <div class="infoDj">
                <h2 id="titDj"></h2>
            </div>

            <div class="titStream"><span id="titstreamtext"><span></div>
        </div>
    </div>
</body>
```

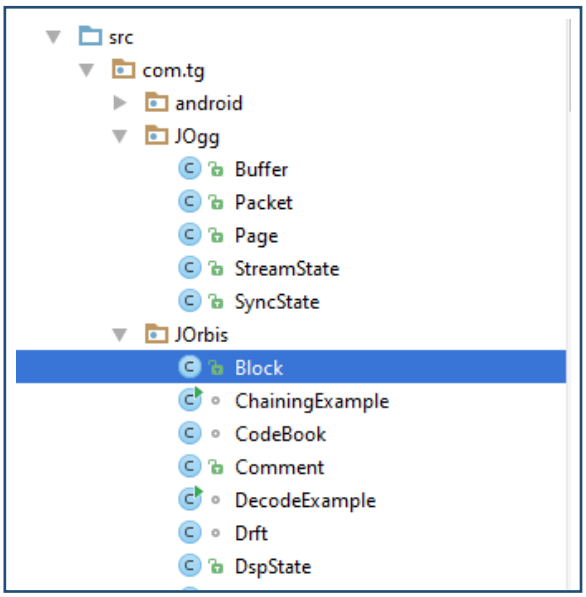
Fuente: el Autor.

El botón actualizar hace un llamado a la misma función Jquery que anteriormente fue presentada, de manera que se evitan redundancias y se le permite al usuario conocer los nuevos canales que están transmitiendo.

8.4.7.5. Implementación y utilización de plugin OGG para cordova.

Para realizar el plugin que contenga la función de la reproducción del flujo de datos es necesario implementar las librerías JOrbis en la sección del código nativo del proyecto, ya que la API de audio de HTML5 no brinda soporte a streaming OGG en implementaciones cordova. Las libreia de JOrbis pueden conseguirse en la página de jcraf https://www.jcraft.com. Al descargar las librerías, estas de deben agregar al proyecto en la carpeta source como se ve en la imagen que se observa a continuación:

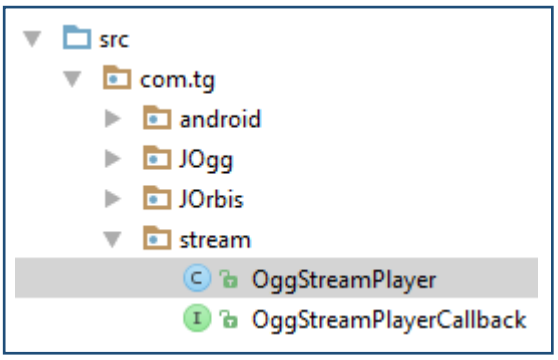
Figura 60. Libreria Jorbis y JOgg



Fuente: el Autor.

Posterior a agregar la librería requerida para la reproducción nativa, se proceder a descargar la clase OggStreamPlayer, que utiliza la librería JCraf para realizar la reproducción mediante el canal de salida de audio del harware, la descarga se puede realizar desde el perfil de GitHub de radzio **(Radzio on Amazon WS, 2015)**. Esta librería también debe ser agregada en la carpeta de fuentes de Android.

Figura 61. Libreria OggStramPlayer



Fuente: el Autor.

Para finalizar la implementación nativa del plugin, se debe proceder a hacer la clase PlayerOgg y el método Run, que utiliza las clases anteriores y es llamado desde la capa de HTML5 para realizar la reproducción nativa, a continuación, se observa la clase:

Figura 62. Método Nativo para reproducir OGG

```
public class PlayerOGG extends CordovaPlugin {

    private OggStreamPlayer player;

    public PlayerOGG(){
        player=new OggStreamPlayer();
    }

    private void run(String mpoint, CallbackContext callbackContext) {
        if(mpoint != null && mpoint.length() > 0) {
            this.playMyStream(mpoint);
            callbackContext.success(mpoint);
        }else{
            callbackContext.error("Expected one non-empty string argument.");
        }
    }
    private void playMyStream(String mount){
        player.playAsync(mount);
    }
}
```

Fuente: el Autor.

La clase PlayerOgg extiende a CordovaPlugin para que esta pueda ser visualizada por la capa HTML5 y así permitir el acceso a sus métodos, además de permitir la instanciación de la clase CallbackContext que se encarga de enviar una respuesta a una petición realizada desde JavaScript, la parte más importante de la petición es el parámetro que se envía, el cual contiene la URL del punto de montaje que se quiere reproducir.

Después de implementar la capa nativa, se va a desarrollar la capa de HTML5 que permite hacer una petición desde la aplicación cliente, en donde se especifica la clase y el método requerido, además de los parámetros estructurados en un formato JSON.

El primer script a realizar es el encargado de la comunicación entre JS y JAVA mediante la instanciación del método context del objeto Cordova , con el cual se estructura la petición a la capa nativa.

Figura 63. Objeto JS para llamar función nativa

```
var mountpoint = {
  mount: function(successCallback, errorCallback, mpoint){
    cordova.exec(successCallback,
        errorCallback,
        "PlayerOGG",
        "run",
        [{
          "mpoint":mpoint
        }]
    );

  }
}
module.exports = mountpoint;
```

Fuente: el Autor.

Para utilizar este script, se debe permitir al usuario seleccionar el punto de montaje de la lista retornada, para que de esta manera JQuery detecte el evento sobre el canal elegido, extraiga la URL del punto de montaje e instancie un objeto de mountpoint que envié el mensaje al reproductor OGG. El método utilizado se dispara cuando el usuario da un click sobre una capa que pertenezca a la clase “canal” y contenga el atributo “ip” con la dirección del flujo de datos del servidor.

Figura 64. Método JS para seleccionar canal

```
$(document).on( "click", ".canal", function() {
    var ip=$(this).attr('ip');
    var mount=$(this).attr('mountp')+".ogg";
    var mountPoint="http://"+ip+":8000/"+mount;
    var djMountP = $(this).attr('dj');
    $('#cabeceraCanales').css('display','none');
    listenMountEvent(ip,mount);
    playMountP(mountPoint,djMountP);

});
```

Fuente: el Autor.

Por último y no menos importante, se debe declara el plugin en el fichero config.xml del proyecto, en el cual se determinará el componente de la capa HTML5 y el componente de la capa nativa además de sus rutas correspondientes.

Figura 65. Plugin definido en config.xml

```
<plugin xmlns="http://apache.org/cordova/ns/plugins/1.0"
    id="OggStreamPlayer"
    version="0.1.0">

    <name>PluginStreamOgg</name>
    <description>Listening oga stream</description>
    <license>Apache 2.0</license>
    <keywords>cordova,streamoga</keywords>


    <js-module src="www/js/mountpoints.js" name="mountpoint">
        <clobbers target="mountpoint" />
    </js-module>

    <!-- android -->
    <platform name="android">
        <config-file target="config.xml" parent="/*">
            <feature name="PlayerOGG">
                <param name="android-package" value="com.tg.android.PlayerOGG"/>
            </feature>
        </config-file>

        <source-file src="src/com/tg/stream/PlayerOGG.java" target-dir="src/com/tg/stream" />
    </platform>
</plugin>
```

Fuente: el Autor.

8.4.7.6. Listado de canciones en tiempo real.

Al momento el que el usuario escoge un canal, la aplicación reproduce el punto de montaje y lleva al usuario a una interfaz gráfica en la cual se verá la lista de canciones que ha reproducido el DJ, incluyendo la actual. La interfaz incluye un botón que permite retornar a la capa que contiene la lista de canales disponibles.

Figura 66. Vista de canal en reproducción

Fuente: el Autor.

Teniendo en cuenta el servicio restful de “Titulo Mount Point” que retorna el título del tema actual de un solo punto de montaje en reproducción, se va a realizar una petición al servicio cada 3000 milisengundos, mediante JQuery Ajax para actualizar constantemente la lista de temas si el título no ha sido incluido.

Figura 67. Método JS para actualizar canal

```
var lastTheme="";
function listenTitleMountP(servidor,mountp){
    jQuery.ajax({
          type: "GET",
          url: "http://54.207.63.75:8080/wsicecast/rest/icecast/titulompoint/"+servidor+"/"+mountp,
          dataType: "text",
          success: function (data, status, jqXHR) {
          newTheme=data;
          if(newTheme==lastTheme){
          }else{
            lastTheme=newTheme;
            $('#titstreamtext').append(newTheme+"<br><br>");
          }

          },
          error: function (jqXHR, status) {
                          alert('error');
          }
    });
}
```

Fuente: el Autor.

8.4.7.7. Botón para silenciar la transmisión.

Esta función tiene como finalidad permitir que el usuario quite el volumen del flujo de transmisión mediante un cambio de punto de montaje escuchado por el reproductor a uno no existente, al volver a oprimir el botón para retornar el audio, se volverá a escuchar el punto de montaje seleccionado por el usuario.

Figura 68. Función de interfaz para quitar el volumen.

Fuente: el Autor.

El método utilizado para cambiar a un punto de montaje vacío es el mismo utilizado para enviar un punto de montaje al reproductor nativo cuando el usuario selecciona un canal en la lista.

8.4.7.8. Botón para salir de la aplicación.

Una función que facilita la usabilidad de la aplicación consisten en permitirle al usuario cerrar la misma desde ella con el fin de proveer una ruta de salida que detenga los servicios consumidos y libere los recursos utilizados por esta, la función de cordova.js que se utiliza para este requerimiento es: navigator.app.exitApp().

Figura 69. Botón para salir de la aplicación

Fuente: el Autor.

8.5. RESULTADOS DE PROCEDIMIENTO

En los resultados se podrá conocer los registraos graficos de las aplicaciones con todas las funciones incluidas tanto en la aplicación web como en la aplicación móvil.

8.5.1. App web

A continuación, se encuentra el primer estado de la aplicación web en ejecución en el entorno de producción, en la cual se denotan las funciones de: formulario de registro de dj, formulario de ingreso, lista de canales en tiempo real, botón para actualizar la lista, reproductor de audio con control de apagado y encendido de volumen y el link para descargar la aplicación.

Figura 70. Estado 1 de aplicación web en producción

Fuente: el Autor.

La aplicación web es responsiva respecto a pantallas de 1200px y 768px, ya que en dispositivos móviles no es funcional la API de audio de HTML5 para streaming OGG y por este motivo es necesario ocultar la capa donde se listan los canales para evitar que el usuario se vea en un escenario en la cual sean visibles componentes no funcionales en sistemas operativos móviles.

Figura 71. Estado 1 de aplicación web en moviles

Fuente: el Autor.

8.5.2. App móvil.

El resultado final de la aplicación móvil permite observar dos pantallas, la primera en donde se listan los canales de DJs y le permite al usuario seleccionar uno para conectarse a este. Además, se puede encontrar el botón para actualizar la lista y otro para salir de la aplicación sin que continúe la reproducción.

Figura 72. Pantalla 1 de la aplicación móvil

Fuente: el Autor.

Posteriormente a escoger uno de los canales, se ingresará a una pantalla en la cual se verá el nombre del DJ, el botón para silenciar y retornar el audio de la transmisión, el botón para retornar a la pantalla principal y una capa en la cual se listarán en tiempo real los nombres de los temas musicales transmitidos en el canal.

Figura 73. Pantalla 2 de aplicación móvil

Fuente: el Autor.

8.6. DESARROLLO DE LA METODOLOGÍA DEL ESTUDIO.

En la siguiente sección se puede apreciar el desarrollo de los puntos planteados en la metodología del estudio para poder comprobar la hipótesis, teniendo en cuenta las variables planteadas en la misma.

8.6.1. Definir puntos de montaje.

Para el desarrollo de la prueba se utiliza 1 punto de montaje denominado mount.ogg en cada servidor (Tokyo, Frankfurt y Sao Pablo).

8.6.2. Definir clientes.

Los 6 clientes definidos para la prueba son:

1. Chrome Canary.
2. Mozilla Firefox.
3. Chrome estándar.
4. Samsung A3 serie 6.
5. Samsung J7.
6. LG G3 mini.

Es necesario conectar los clientes a una red wifi para evitar fallos en la prueba por falta de disponibilidad de transmisión de datos.

8.6.3. Definir fuentes de streaming.

Las fuentes seleccionadas para realizar el envió del flujo de datos a los servidores son:

1. Traktor DJ V2 Pro.
2. Virtual DJ V7 Pro.
3. Virtual DJ V8 Pro.

Es necesario conectar las fuentes a una red wifi para evitar fallos en la prueba por falta de disponibilidad de transmisión de datos.

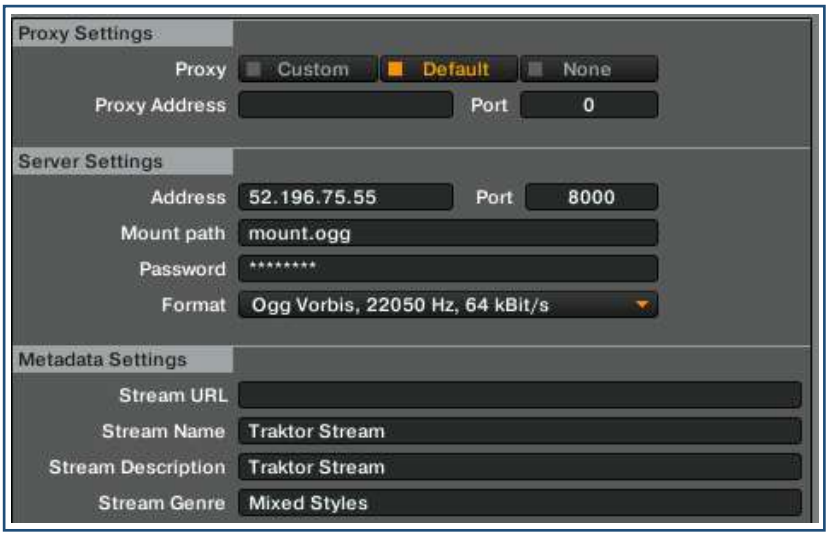
8.6.4. Diseño de prueba.

Después de definir las instancias de software utilizadas por clientes y fuentes, se debe proceder a realizar la configuración de los módulos RTP de cada fuente.

Para la prueba, Traktor DJ envía el flujo de datos a la instancia en Tokyo, la configuración del módulo se realiza en:

File-> Preferences->Broadcasting.

Figura 74. Panel RTP de Traktor 2

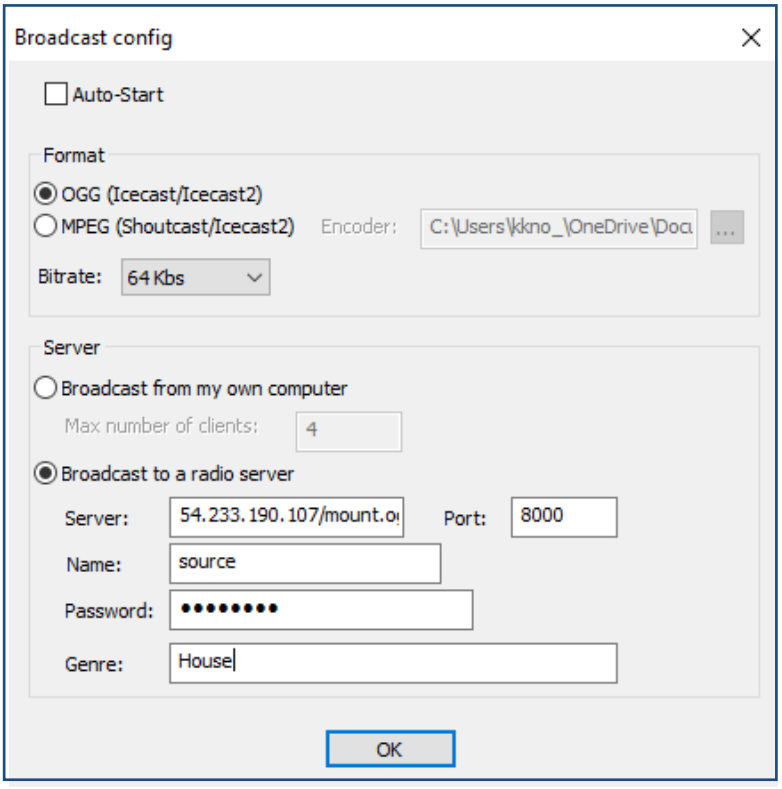


Fuente: el Autor.

Para la prueba, Virtual DJ 7 envía el flujo de datos a la instancia en Sao Pablo, la configuración del módulo se realiza en:

Record-> Config.

Figura 75. Panel RTP de Virtual DJ 7

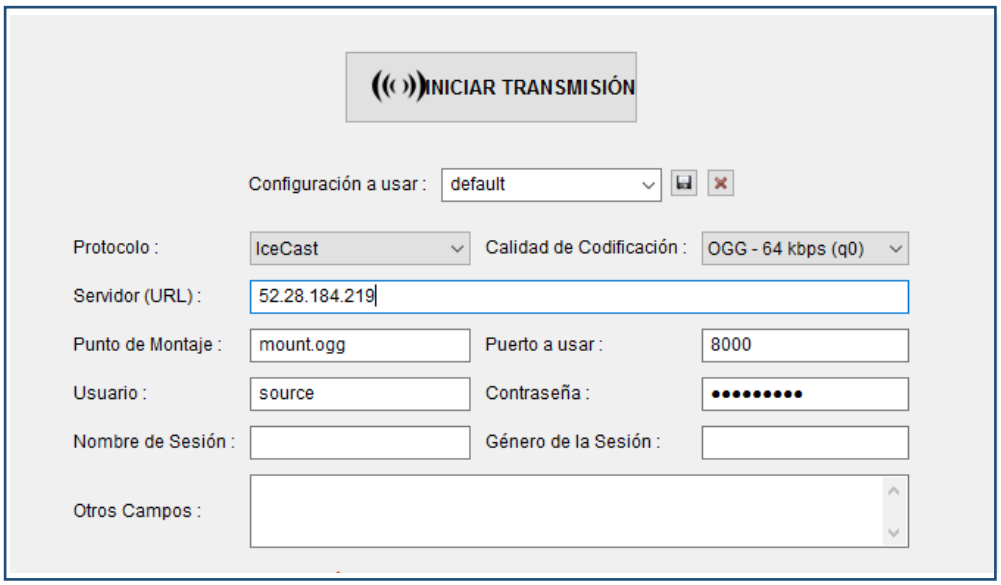


Fuente: el Autor.

Para la prueba, Virtual DJ 8 envía el flujo de datos a la instancia en Frankfurt, la configuración del módulo se realiza en:

Settings-> TRANSMITIR.

Figura 76. Panel RTP de Virtual DJ 8



Fuente: el Autor.

Al terminar de configurar los módulos RTP de cada software fuente, se procede a desplegar las aplicaciones clientes en los dispositivos móviles y mediante navegación, se debe llegar a la aplicación web en los PC para utilizar los canales indicados.

Así se realiza la configuración para el consumo del flujo de datos de los puntos de montaje:

Al punto de montaje mount.ogg de la instancia de Sao Pablo se conectan los clientes:

- Chrome Canary.
- Samsung A3 serie 6.

Al punto de montaje mount.ogg de la instancia de Tokio se conectan los clientes:

- Mozilla Firefox.
- Samsung J7.

Al punto de montaje mount.ogg de la instancia de Frankfurt se conectan los clientes:

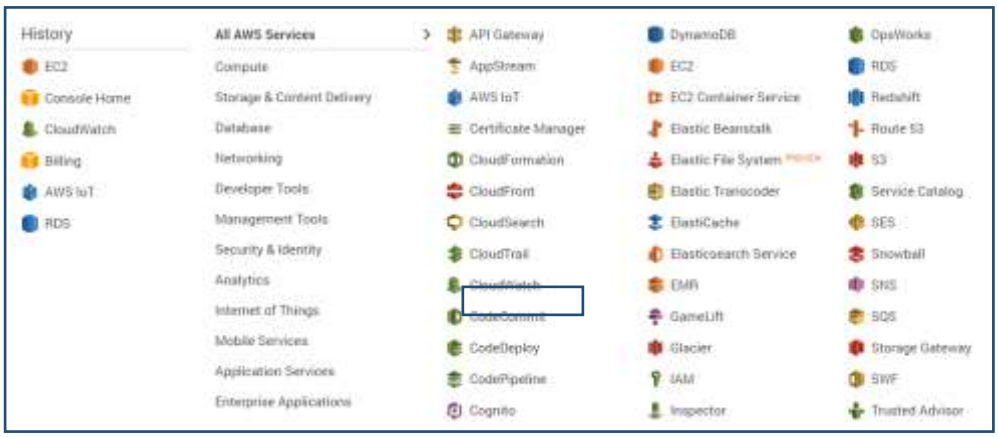
- Chrome estándar.
- LG G3 mini.

8.6.5. Configuración de Cloud Watch.

Después de iniciar la transmisión y conectar los clientes, se debe configurar un (1) Dashboard de CloudWatch para cada instancia, estos tableros permiten dos tipos de instrumentos, ungráfico para estadísticas y uno de texto para datos básicos y elementos de navegación entre URLs. En este caso se despliegan únicamente herramientas graficas ya que la finalidad es realizar un estudio del comportamiento de los recursos de la aplicación.

Para configurar cada panel es necesario ubicarse en la zona de disponibilidad y dirigirse al servicio de CloudWatch.

Figura 77. Servicio CloudWatch



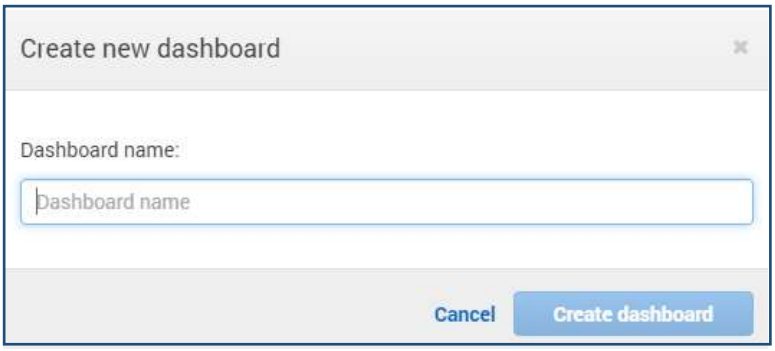
Fuente: el Autor.

Posteriormente a esto, se debe dar clic en la opción Dashboards-> Create Dashboar.

Figura 78. Opción de creación de dashboard

Al crear el tablero de instrumentos, este solicita el nombre que es definido por la zona de disponibilidad donde se está creando, es decir si se está configurando el servicio de Frankfurt, así se llamara el dashboard.

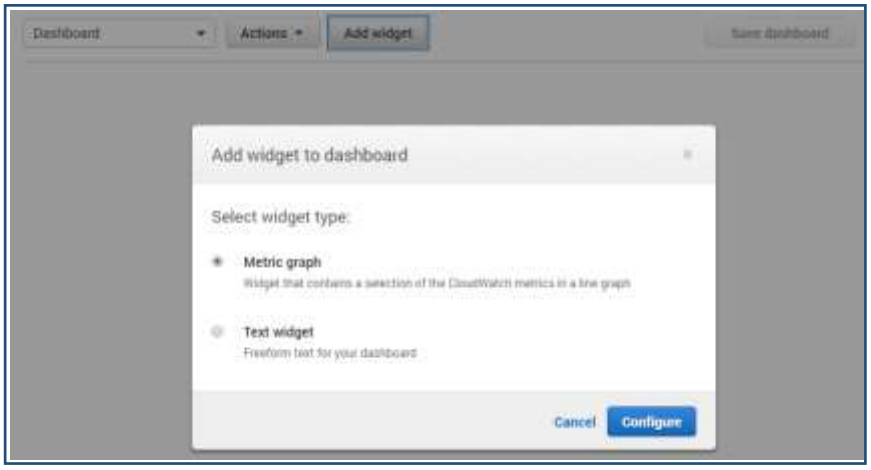
Figura 79. Nombre dashboard



Fuente: el Autor.

Después de definir el nombre, se deben crear las herramientas, para esto se puede dar clic en el botón “Add Widget” y se selecciona el instrumento, que como se especificó anteriormente es gráfica.

Figura 80. Seleccionar herramienta



Fuente: el Autor.

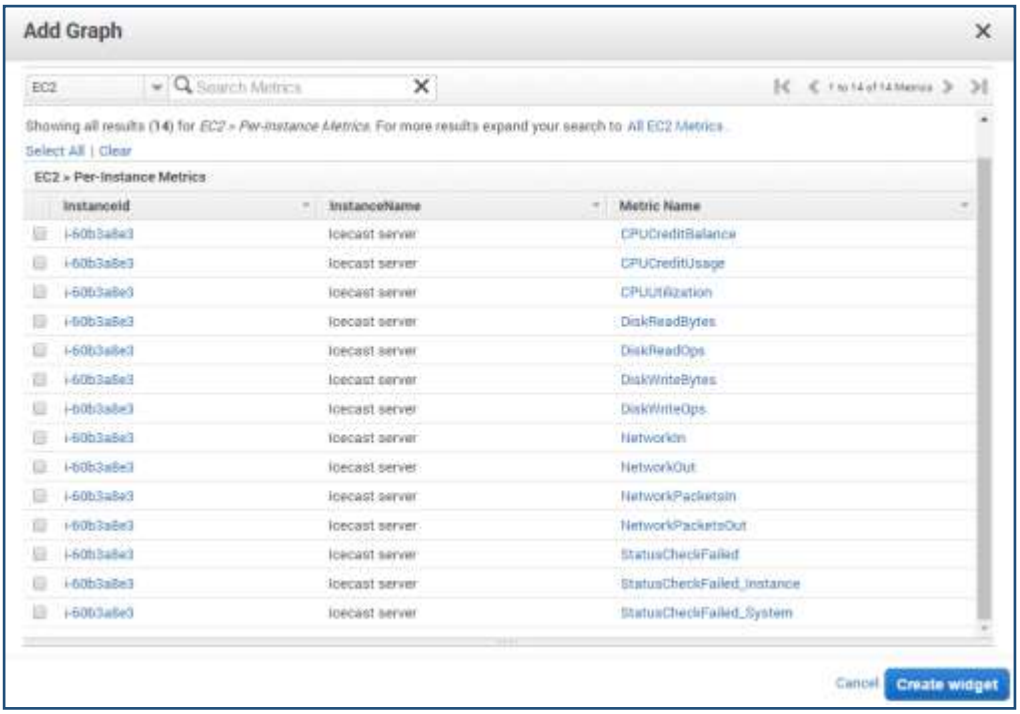
Al crear la herramienta, se proceder a escoger las métricas que se desean, ya sea por zona de disponibilidad o pre instancia específica, en este caso será para la instancia de streaming de cada server en el servicio de EC2.

Figura 81. Fuente de métricas

Fuente: el Autor.

Después de escoger la instancia a evaluar, se seleccionan las métricas que se desean utilizar.

Figura 82. Lista de métricas



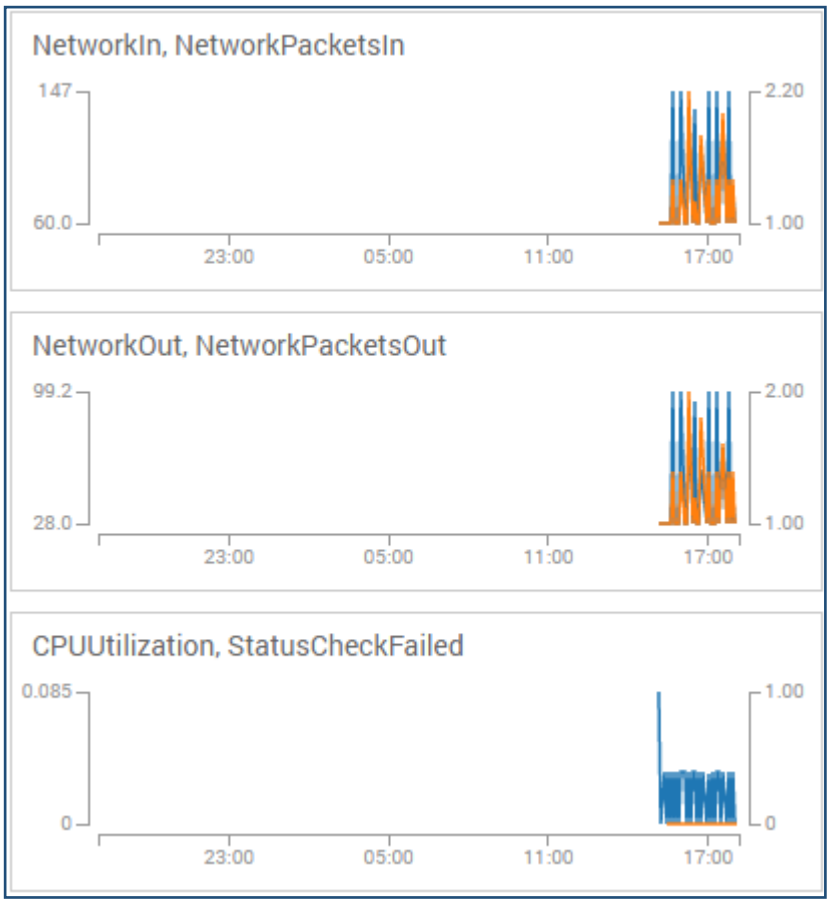
Fuente: el Autor.

Para los Dashboard de cada zona de disponibilidad se crean tres graficas que constan de:

- Análisis de tráfico entrante con las métricas de NetworkIn y NetworkPacketsIn.
- Análisis de tráfico saliente con las métricas de NetworkOut y NetworkPacketsOut.
- Por último, la herramienta de análisis de CPU y disponibilidad con CPUUtilization y StatusCheckFailed.

Así es la configuración de cómo debe verse cada Dashboard creado.

Figura 83. Estructura de herramienta gráfica



Fuente: el Autor.

Por último, se debe poder observar la lista de Dashboard creados, cada uno con los 3 tableros indicados para realizar la medición y poder conocer los resultados.

Figura 84. Lista de tableros de instrumentos

Fuente: el Autor.

8.6.6. Registro de resultados CloudWatch.

Para finalizar, se utiliza recortes en Windows 10 para imprimir los registros de cada instancia con un rango de 3 horas, en las cuales se está transmitiendo desde las fuentes y se permite que las aplicaciones clientes estén conectadas. Si se detecta una desconexión de algún cliente, se proceder a revisar la métrica de disponibilidad y la conexión de red del cliente para permitir reestablecerla.

- **Registros de Frankfurt.**

  A continuación, se observa el registro de las métricas de la instancia de Frankfurt para conocer los resultados.

Figura 85. Metricas de FrankFurt

Fuente: el Autor.

- **Registros de Sao Pablo.**

A continuación, se observa el registro de las métricas de la instancia de Sao Pablo para conocer los resultados.

Figura 86. Métricas de Sao Pablo

Fuente: el Autor.

- **Registros de Tokio.**

  A continuación, se observa el registro de las métricas de la instancia de Sao Pablo para conocer los resultados.

Figura 87 Metricas de Tokyo.

Fuente: el Autor.

Respecto a los datos registrados en las tres figuras anteriores, se puede deducir que, mientras las fuentes envíen datos y los clientes estén conectados, el comportamiento del tráfico de entrada vario entre 400 y 510 kilobytes durante toda la prueba.

8.6.7. Registro de memoria RAM del Sistema operativo.

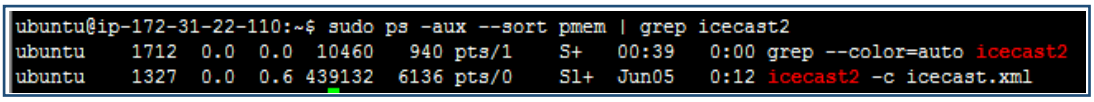
Como se especificó en la metodología, se observa el uso de memoria RAM del servidor Icecast 2 en el sistema operativo con el comando `ps -aux --sort pmem | grep icecast2`, el comando está divido en 5 partes que son:

- PS: muestra en pantalla información respecto a los procesos activos.

- AUX: estos parámetros permiten:
  - A: mostrar los procesos de otros usuarios
  - U: muestra el usuario y la hora de inicio.
  - X: muestran procesos no controlados por alguna terminal.
- SORT: organiza el contenido línea por línea.
- PMEM: ordena los procesos por los que más consumen CPU.

A continuación, se pueden observar el registro de memoria RAM del sistema operativo de la instancia de Frankfurt.

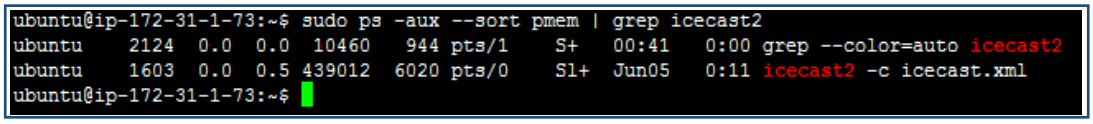
Figura 88. Registro de RAM, Frankfurt



Fuente: el Autor.

A continuación, se puede observar el registro de memoria RAM del sistema operativo de la instancia de Sao Pablo.

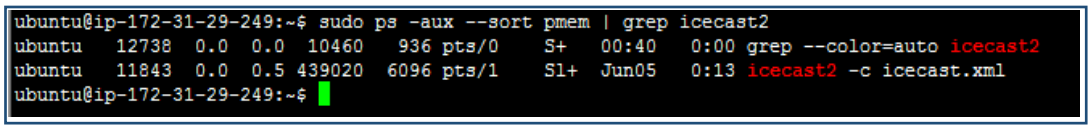
Figura 89. Sao Pablo



Fuente: el Autor.

A continuación, se puede observar el registro de memoria RAM del sistema operativo de la instancia de Tokyo.

Figura 90. Tokyo



Fuente: el Autor.

8.6.8. Interpretación de los resultados.

Según los datos registrados mediante los recortes de pantalla de las métricas y el consumo de recursos medidos, se va a realizar una interpretación acotando las variables dependientes e independientes definidas en la metodología.

La transparencia y la usabilidad de la aplicación cliente se da a partir del uso de la misma, ya que esta cumple su función de una manera simple e intuitiva sin que el usuario perciba las operaciones técnicas y lógicas que se desarrollan en los procesos que permiten obtener la dirección de los puntos de montaje mediante los servicios web, y así, sincronizarse con el flujo de datos de los servidores Icecast 2 para reproducirlos en el formato OGG Vorbis para que sea escuchados.

Considerando el comportamiento de los dispositivos cuando están conectados a un punto de montaje, es importante destacar que su rendimiento no fue afectado en lo más mínimo, ya que mientras las métricas de NetworkOut y NetworkPacketsOut registraron conexión con los mismos, estos permitieron realizar otras funciones como ejecutar o instalar aplicaciones, navegar en internet o utilizar los datos en redes sociales y en aplicaciones de comunicación escrita de manera rápida y eficiente sin presentar retardos en la ejecución de sus interfaces gráficas y funciones principales. Además de permitir retornar a las aplicaciones clientes y navegar por los canales sin ningún inconveniente., de esta manera se puede deducir que el rendimiento de las mismas en Android y en la Web fue fluido sin causar interrupciones en otros procesos.

Considerando los registros tomados de la memoria RAM de los sistemas operativos en las instancias de EC2, se puede deducir que la utilización de la misma fue baja, ya que cumplieron con la funcionalidad sin consumir muchos recursos de la memoria volátil. En los registros tomados se puede observar que los servidores utilizaron entre 0.5% y 0.6% de un 100% de la memoria. Independientemente del uso de memoria, la métrica de CPUUtilization registró un uso de CPU de 3,7% de un 100%, se puede deducir que el uso de recursos de la unidad central de procesamiento fue considerablemente bajo.

Durante la prueba ejecutada y documentada, se observó que la métrica de disponibilidad del sistema StatusCheckFailed, mantuvo su valor en 0, de manera que el sistema no falló y los recursos estuvieron disponibles para la ejecución del servidor y el manejo de los flujos de datos.

La transmisión de datos fue exitosa y demostrada con las métricas de NetworkIn y NetworkPacketsIn que registraron un ingreso de datos por cada punto de montaje, de hasta 776 Bytes de ingreso. No acorde con el bitrate de 64kbps se considera que el servidor y la fuente abren una conexión del equivalente del ancho de banda disponible para el dispositivo que contiene el software fuente, la cual puede ser fragmentada en canales de 64kbps, pero solamente se utiliza uno para enviar el flujo de datos y hacer la transmisión.

Considerando la simetría de peticiones en una de red es importante considerar que, si una entrada de datos realiza un consumo de banda determinado, una petición para la salida de los mismos va a ser una aproximación relativa a los entrantes.

Confrontando las métricas de salida con las métricas de entrada, es importante observar que el registro tomado de las primeras duplica en Bytes y paquetes los de las segundas, ya que por cada fuente se conectaron dos clientes. Por ejemplo, si se considera que una entidad envía a un servicio Restful un fichero JSON de 100Kb mediante una acción tipo POST, al momento de hacer una solicitud del mismo fichero mediante una petición GET, el tamaño del JSON retornado debe ser de aproximadamente 100kb.

Con este comportamiento de red es fácil deducir que fue coherente el tráfico de datos de la aplicación, ya que por el tamaño de cada petición realizada se recibió una cantidad de datos relativo a los enviados por las fuentes.

9. PROYECCIONES FUTURAS

Para un proyecto futuro se proyecta que se puede hacer una integración con redes sociales de diferentes tipos para permitir una mayor interacción entre los DJs y los oyentes, para esta finalidad será necesario determinar el objetivo de la interacción de cada DJ con las diferentes redes sociales, de manera que se determine la finalidad con la cual es usada cada una, para así enriquecer la plataforma con más contenido de interés, generado en cada una de estas para lograr que los DJs puedan alimentar sus perfiles de manera automática.

Posteriormente a determinar el uso que se le puede dar a cada red social por parte de la aplicación de streaming, es necesario conocer el alcance permitido por las mismas plataformas, ya que la red social se puede ver sujeta a políticas restrictivas al momento de usar sus servicios web, con el fin de proteger la privacidad de sus usuarios y esto debe ser tenido en cuenta al momento de especificar un objetivo o requerimiento.

Desarrollar el código nativo de OGG Vorbis para otras plataformas incluidas en el soporte de apache cordova, de manera que se pueda entregar más alcance a la aplicación móvil sin restringirla únicamente a versiones del sistema operativo Android.

10. CONCLUSIONES

- Utilizar formatos de compresión OGG permite desarrollar eficientes sistemas de transmisión de contenido multimedia sin depender de licencias comerciales para el formato y los codecs que permiten la comprensión y descompresión en los servidores y en las aplicaciones clientes.
- Utilizar una plataforma en la nube como Amazon EC2 provee un alto nivel de elasticidad, ya que permite aumentar los recursos de transmisión, almacenamiento y procesamiento para dotar las aplicaciones de una infraestructura capaz de escalar en cuestión de minutos para manejar desde unos pocos a millones de usuarios.
- Utilizar HTML5 facilita el desarrollo multiplataforma debido a que, solo es necesario construir una API basada en Javascript para que pueda ser utilizada en las implementaciones realizadas ya sea móviles o de escritorio.
- Considerando el entorno y el contexto de los sistemas utilizados por clientes, fuentes y servidores, se encontró que existen opciones de software que proveen la funcionalidad necesaria para alguno de los nodos requeridos, de manera que, al implementar un sistema de trasmisión, se puede hacer uso de los programas existentes para evitar la codificación de los mismos.
- Utilizar una herramienta para analizar el comportamiento de un sistema prototipado, facilita la obtención de información necesaria para llevarlo a producción, debido a que se conocen las implicaciones respecto al uso de los recursos, y de esta manera se puede calcular el uso físico en un determinado estado de la aplicación, para así tomar decisiones de escalamiento en tiempo real.
- Realizar el análisis para obtener los requerimientos y casos de uso es importante para tener un conocimiento funcional que permita el posterior modelado del sistema para cumplir los alcances, sin incurrir en el diseño y desarrollo de funciones o procesos no contemplados o necesarios para cumplir los objetivos funcionales del sistema.
- La fase de diseño es vital para tener una base estructural en la cual se pueda plantear y planificar los componentes necesarios, y su nivel de comunicación para proveer al sistema con características como: la modularidad, la escalabilidad, la transparencia, la integridad, la Interoperabilidad y la fiabilidad. De manera que al ingresar en la fase de desarrollo se cuente con un planteamiento fiable para saber cómo realizar adecuadamente la aplicación.
- El desarrollo basado en la fase de análisis y diseño permite que se disminuyan los errores y se tenga claro el camino que se debe seguir respecto a la implementación de los módulos e interfaces que los

comunican, para evitar utilizar recursos valiosos en implementaciones no requeridas e innecesarias.

- Obtener conocimiento previo respecto al área a estudiar junto a las entidades e implicaciones encontradas en esta, es importante para poder definir una solución eficiente a un problema real y evitar planteamientos que no traten objetivamente las verdaderas adversidades existentes.

11. BIBLIOGRAFÍA

Á. Cobo, PHP y MySQL: Tecnología para el desarrollo de aplicaciones web, Editorial Díaz de Santos, S.A., 2005, 528 páginas, 9788479787066

***AIBN*,** Wha is a MIME Type, Disponible en (http://www.aibn.com/help/Learn/mimetypes.html). [Consultado el 25 de mayo del 2015]

Alegsa, Definición de acoplamiento, Disponible en (http://www.alegsa.com.ar/Dic/acoplamiento.php). [Consultado el 25 de mayo del 2015]

Amazon, Details, Internet, Disponible en (http://aws.amazon.com/es/ec2/details/). [Consultado el 20 de abril del 2015]

Apache Cordova, Create your first Cordova App, Internet, Disponible en (https://cordova.apache.org/docs/en/latest/guide/cli/index.html). [Consultado el 8 de Septiembre del 2015]

BeatBravo, About Us, Internet, Disponible en (https://beatbravo.com/about). [Consultado el 7 de abril del 2015]

CAMACHO Erick, Introducción a Maven, Disponible en (http://static1.1.sqspcdn.com/static/f/923743/15025126/1320942755733/Tutorial_de_Maven_3_Erick_Camacho.pdf?token=sBJ6Tx2H90qWXTSQLsam9AcB1e8%3D). [Consultado el 21 de mayo del 2015]

Colombiaredes, Hosting en Colombia redes, Internet, Disponible en (https://www.colombiaredes.net/). [Consultado el 7 de abril del 2015]

CONILL Ernest, Streaming Djs Beta… On air, Internet, 8 de mayo del 2013, Disponible en (http://streamingdjs.com/eng/blog/Streaming_DJs_new_online_djs_service#.VY2uUfl_Okp). [Consultado el 7 de abril del 2015]

Crunchbase, Livestream, Internet, Disponible en (https://www.crunchbase.com/organization/livestream). [Consultado el 7 de abril del 2015]

Crunchbase, Radionomy, Intenet, Disponible en (https://www.crunchbase.com/organization/radionomy). [Consultado el 7 de abril del 2015]

D. Rayburn, Streaming and Digital Media: Understanding the Business and Technology, Taylor & Francis, 2012, 264 páginas, 9781136032172

Definición, Difusión, Disponible en (http://definicion.de/difusion/). [Consultado el 25 de mayo del 2015]

Desarrollo Android, Develop Internet, Disponible en (https://developer.android.com/develop/index.html). [Consultado el 23 de mayo del 2015]

Findthecompany, Kal Soluciones Web Colombia SAS, Internet, Disponible en (http://fichas.findthecompany.com.mx/l/131305039/Kal-Soluciones-Web-ColombiaSas-en-Bogota). [Consultado el 7 de abril del 2015]

G. Coulouris, J. Dollimore, T. Kindberg and J. B. P. Junquera, Sistemas distribuidos: conceptos y diseño, Pearson Educación, 2001, 726 páginas, 9788478290499

I. Sommerville and M. I. A. Galipienso, Ingeniería del software, Pearson Educación, 2005, 687 páginas, 9788478290741

Icecast, Acerca de nosotros, Internet, Disponible en (http://icecast.org/). [Consultado el 18 de mayo del 2015]

Ihouseu, Paul Van Dyk Partners con Mixify to power conexión global, Internet, 7 de noviembre del 2014, Disponible en (http://ihouseu.es/news/paul-van-dyk-partnerscon-mixify-power-conexi%C3%B3n-global.html). [Consultado el 8 de abril del 2015]

J. Kurose and K. Ross, Redes de computadoras 5 ed, Pearson Addison-Wesley, 2010, 816 páginas, 9788478291199

J. M. Wargo, Apache Cordova API Cookbook, Pearson Education, 2014, 320 páginas, 9780133838596

J. T. Gironés, El Gran Libro de Android, Marcombo, 2013, 446 páginas, 9788426719973

J. Sandoval, RESTful Java Web Services: Master Core REST Concepts and Create RESTful Web Services in Java, Packt Publishing, Limited, 2009, 256 páginas, 9781847196477

L. Debrauwer and F. van der Heyde, UML 2: iniciación, ejemplos y ejercicios corregidos, Ediciones ENI, 2005, 273 páginas, 9782746027985

L.L. María Jesús, Meta Datos, Disponible en (http://www.hipertexto.info/documentos/metadatos.htm). [Consultado el 25 de mayo del 2015]

Listentomyradio, Acerca de nosotros, Internet, Disponible en (http://www.listen2myradio.com/lang/es/company-about-us.html). [Consultado el 7 de abril del 2015]

Livestream, What is livestream?, Internet, Disponible en (https://livestream.com/about). [Consultado el 7 de abril del 2015]

M. Pietro, Transmisión en Internet: streaming de video y audio,Internet, Disponible en (http://www.grc.upv.es/docencia/tdm/practicas/P3.pdf), 2007[Consultado el 8 de abril del 2015]

Más adelante, ¿Que es un Host? - Definición de Host, Disponible en (https://www.masadelante.com/faqs/host). [Consultado el 25 de mayo del 2015]

Músicaelectrónica online, Historia del Dj. Nacimiento y evolución, Internet, Disponible en (http://www.musicaelectronicaonline.com/historia-del-dj-nacimiento-y-evolucion/). [Consultado el 7 de Agosto del 2015]

Network sorcery, RTP, Real-time Transfer Protocol, Internet, Disponible en (http://www.networksorcery.com/enp/protocol/rtp.htm). [Consultado el 25 de mayo del 2015]

PEÑAREDONDA José Luis, Apps.co mostro nueve de sus mejores prototipos, Internet, 13 de febrero del 2013, Disponible en (http://www.enter.co/culturadigital/colombia-digital/apps-co-mostro-nueve-de-sus-mejores-prototipos/). [Consultado el 7 de abril del 2015]

PIETRO SANCHEZ Juan Pablo, Lecciones aprendidas con streaming, Internet, Disponible en (http://www.streamingcolombia.co/blog/). [Consultado el 7 de abril del 2015]

Radionomy, Features, Internet, Disponible en (http://www.radionomygroup.com/en/companies/radionomy/). [Consultado el 7 de abril del 2015]

Radzio on Amazon WS, Android OGG Stream Player, Internet, Disponible en (http://radzio.github.io/AndroidOggStreamPlayer/). [Consultado el 12 de noviembre del 2015]

Richard C. Dorf, Broadcasting and Optical Communication Technology, Apostolis K. Salkintzis, 2006, 408 paginas, 9780203499986.

ROJAS Laura, De a rockola a la aplicación: conoce secret dj, Internet, 27 de Marzo del 2015, Disponible en (http://www.enter.co/chips-bits/apps-software/de-la-rockolaa-la-aplicacion-conoce-secret-dj/). [Consultado el 8 de abril del 2015]

ROJAS Laura, Un grupo de paisas lanzan aplicación que mezcla música y contenido, Internet, 14 de marzo del 2014, Disponible en (http://www.enter.co/cultura-digital/ciudadinteligente/un-grupo-de-paisas-lanzaaplicacion-que-mezcla-musica-y-contenido/). [Consultado el 8 de abril del 2015]

Sistemas heterogéneos**,** Sistemas heterogéneos, Disponible en (http://sistemasetereogeneos.blogspot.com/). [Consultado el 25 de mayo del 2015]

T. Groussard, JAVA 7: Los fundamentos del lenguaje Java, Ediciones ENI, 2012, 410 páginas, 9782746073180

Universidad del valle, Programación extrema, Disponible en (http://eisc.univalle.edu.co/materias/WWW/material/lecturas/xp.pdf). [Consultado el 24 de mayo del 2015]

Vorvis, FAQ, Internet, Disponible en (http://www.vorbis.com/faq/#what). [Consultado el 19 de mayo del 2015]

XIPH, What is, Internet, Disponible en (http://xiph.org/). [Consultado el 19 de mayo del 2015]

ANEXOS

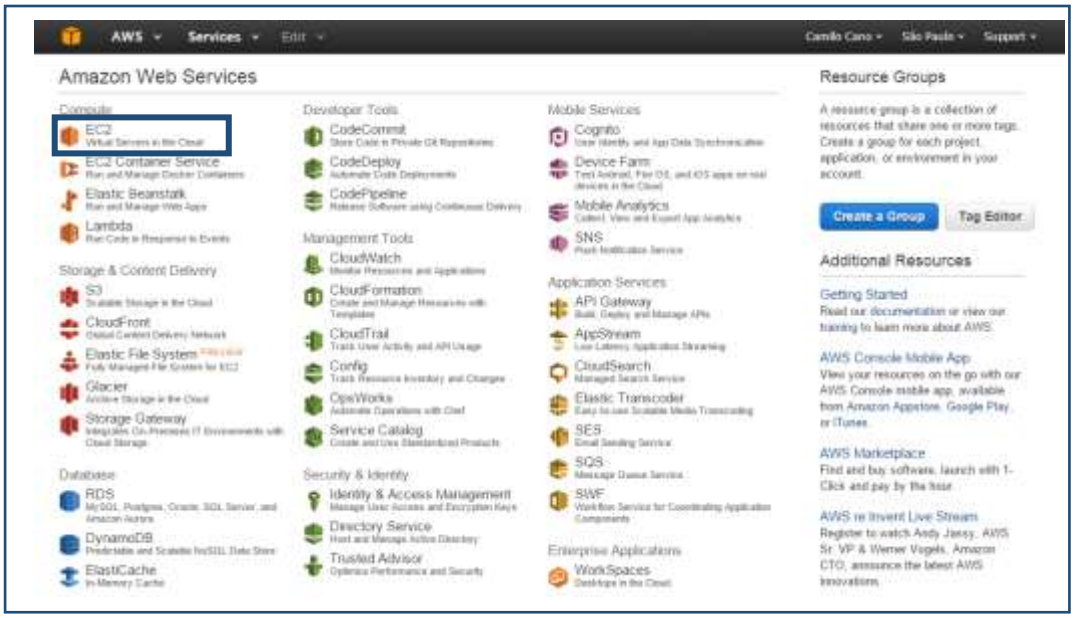
1. Anexo de Instalación de servicio virtual de streaming.

En este punto se van a conocer los pasos necesarios para crear el servicio virtual de streaming en la nube de Amazon EC2.

- **Configuración de Instancias en EC2.**

Para realizar la instalación del servicio, es necesario ingresar a la cuenta de servicio de computación en la nube de AWS y dirigirse a la plataforma de EC2, para utilizar este servicio.

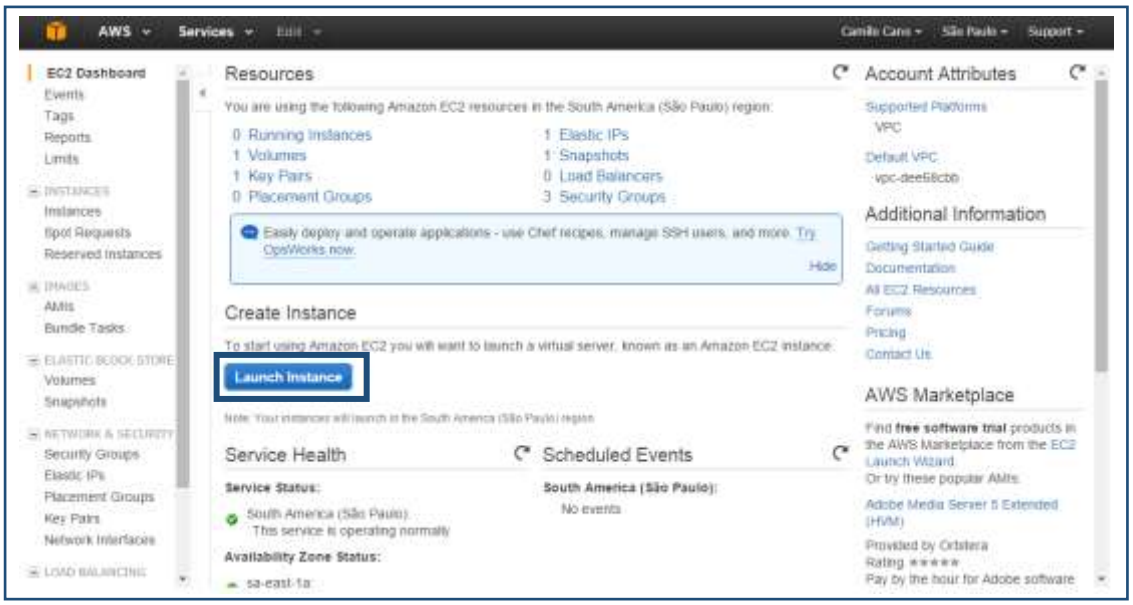
Figura 91. Ingreso a Ec2



Fuente: el Autor.

Posteriormente a escoger el servicio, se observará el panel de gestión de computación elástica en la nube y se procederá a la opción de “Launch Instance” para crear una nueva máquina virtual con el servicio estándar de streaming.

Figura 92. Lanzar instancia

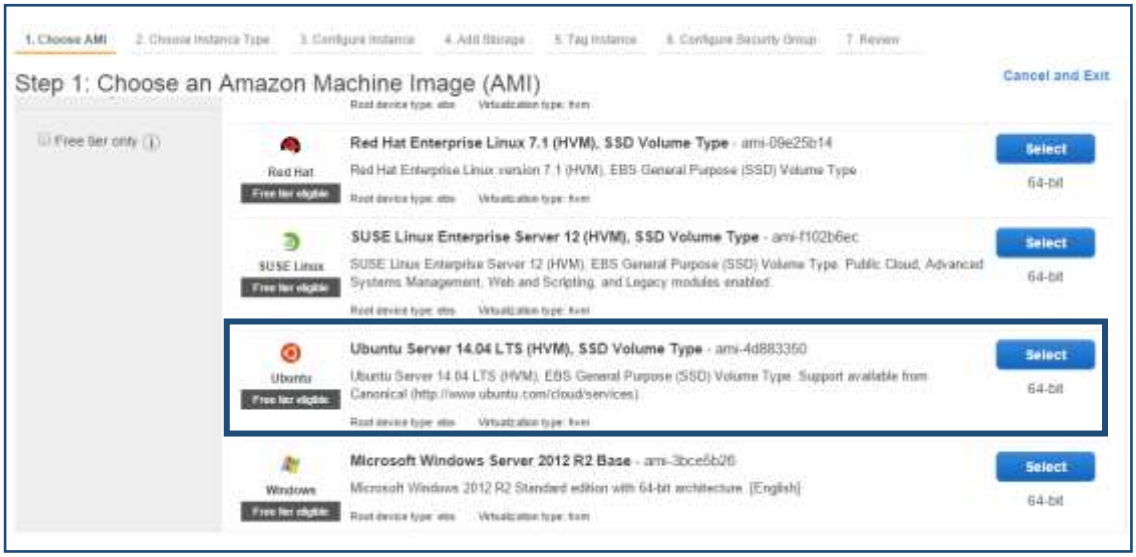


Fuente: el Autor:

Al ingresar a la opción mencionada, se inicia el proceso de creación de la máquina virtual para el servicio de transmisión de audio.

En el paso inicial, se debe escoger la Máquina Virtual de Amazon (AMI) que se desea utilizar, en este caso se utilizará la opción de Ubuntu Server 14.04 en su más reciente versión, que puede variar en función de la última distribución lanzada.

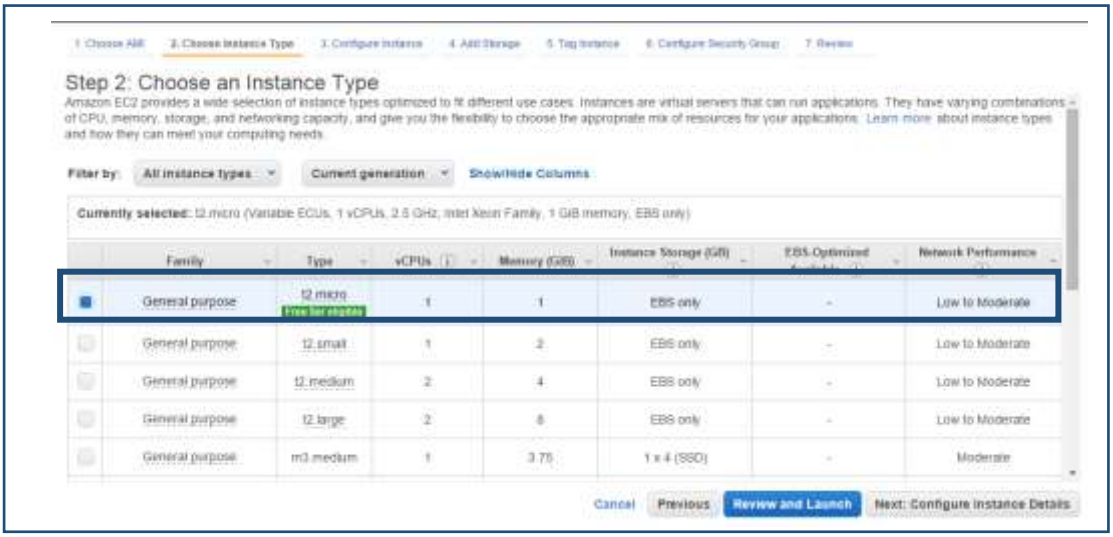
Figura 93. Seleccionar sistema operativo



Fuente: el Autor.

En el paso dos (2), se solicita que se escoja el tipo de instancia que se va a ejecutar, para este entorno de desarrollo se escoge t2.micro ya que provee 15 Gb de banda ancha compartida y ráfagas de CPU que son utilizadas cuando se requiere.

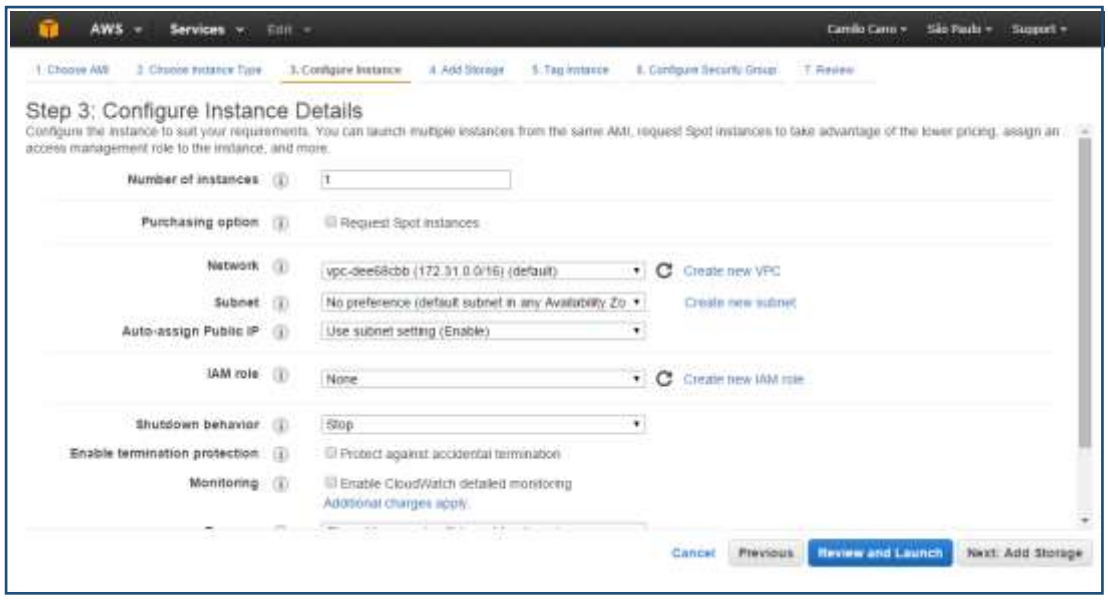
Figura 94. Escoger tipo de instancia



Fuente: el Autor.

EN el paso tres (3), se podrá configurar los requerimientos de red de la instancia, para este desarrollo no se modificará ningún valor y se continuará al siguiente paso.

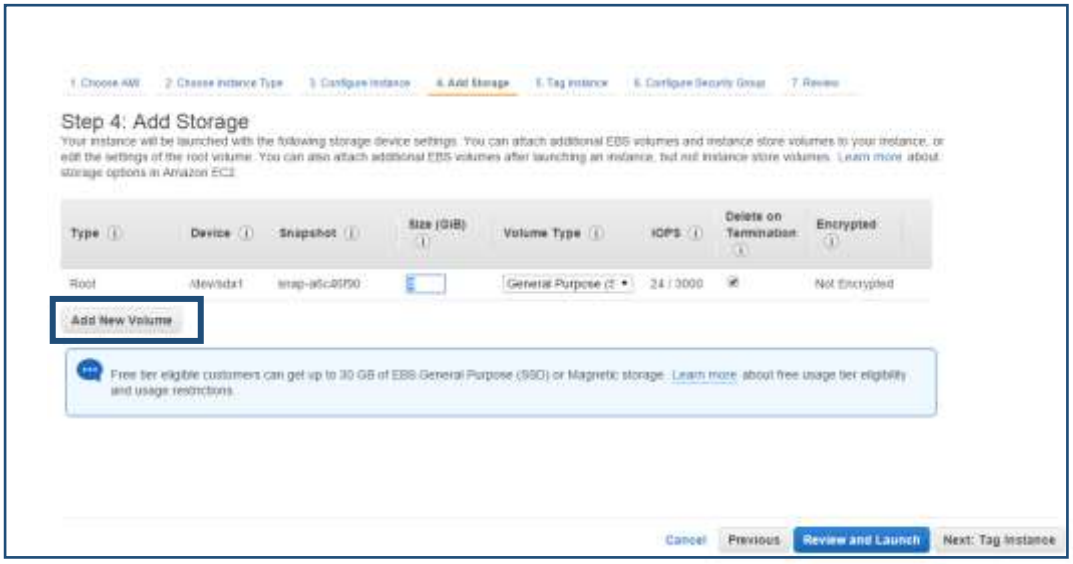
Figura 95. Configurar detalles de instancia



Fuente: el Autor.

En el paso cuatro (4), se debe configurar el dispositivo de almacenamiento, para esto, se utiliza la opción “Add New Volume”, se crear un dispositivo de almacenamiento EBS de 8 GB y se especifica que sea de propósito de almacenamiento general SSD.

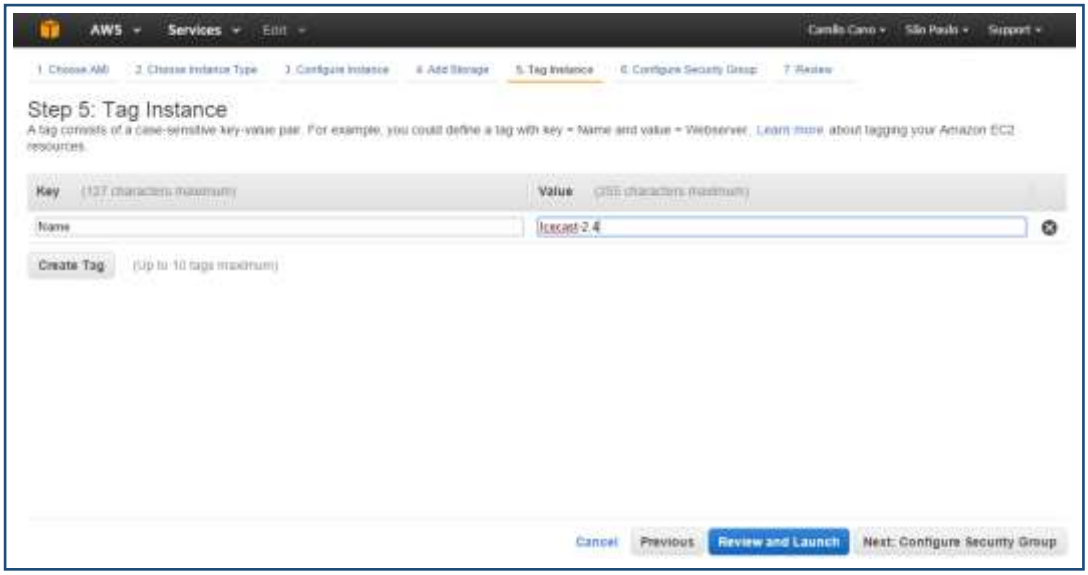
Figura 96. Agregar volumen de almacenamiento



Fuente: el Autor.

En el paso cinco (5), se procede a etiquetar la máquina virtual con el nombre que esta va a tener dentro del pool de AMIs, de esta manera podrá ser identificada.

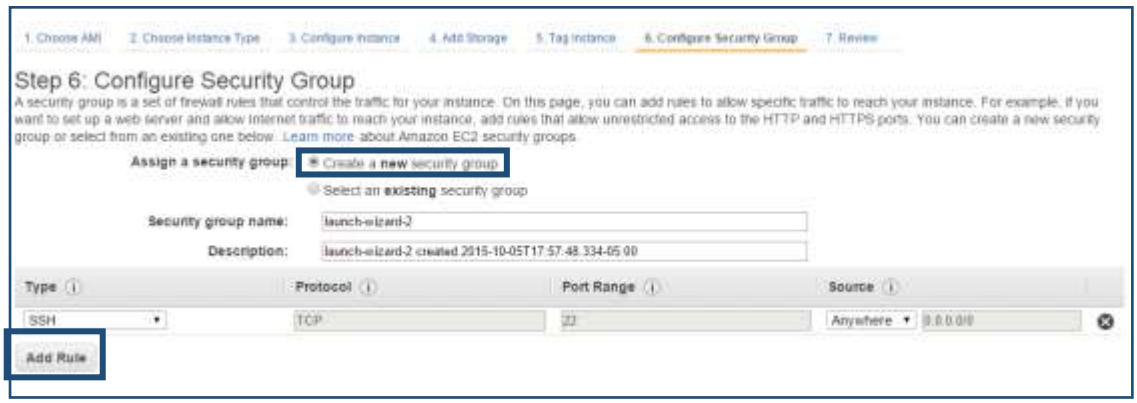
Figura 97. Nombrando instancia



Fuente: el Autor.

En el paso seis (6), se va a definir el grupo de seguridad para gestionar los permisos a los puertos e IPs para conexiones remotas, inicialmente se procede a crear un nuevo grupo con la opción “Create a new security group”, y con la opción “Add Rule” se podrán agregar las reglas para el control del tráfico de la instancia.

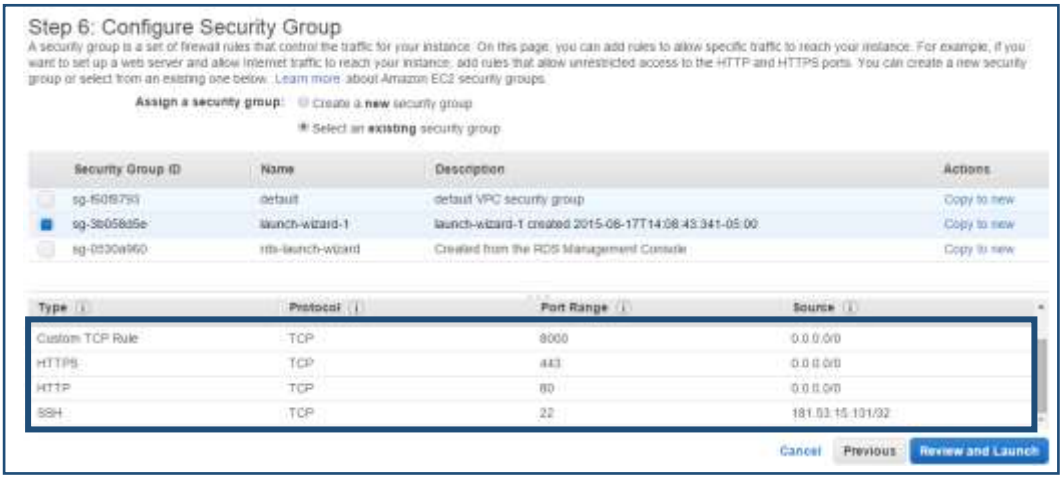
Figura 98. Configurando la seguridad



Fuente: el Autor.

En este caso se habilita el puerto 80, 443, 8000 y 22. En el valor source de: 80, 443 y 8000 se utiliza 0.0.0.0/0 para permitir el acceso del tráfico de toda máquina externa. Para el servicio SSH, se habilita puerto 22 y se define la IP para conexión segura.

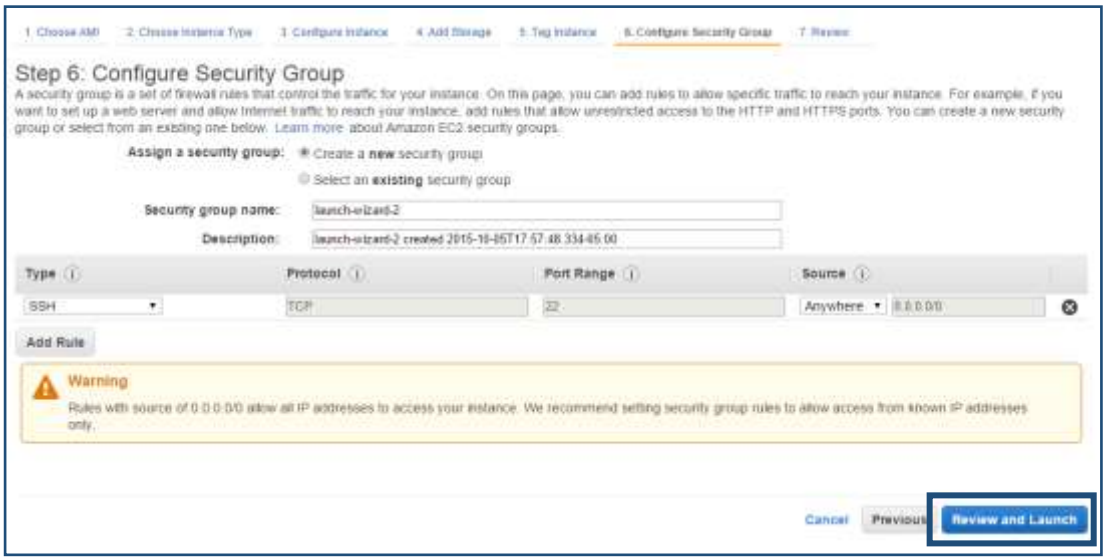
Figura 99. Configurar grupo de seguridad



Fuente: el Autor.

Por ultimo en el paso siete (7) se puede proceder a revisar la configuración establecida y realizar el lanzamiento de la máquina.

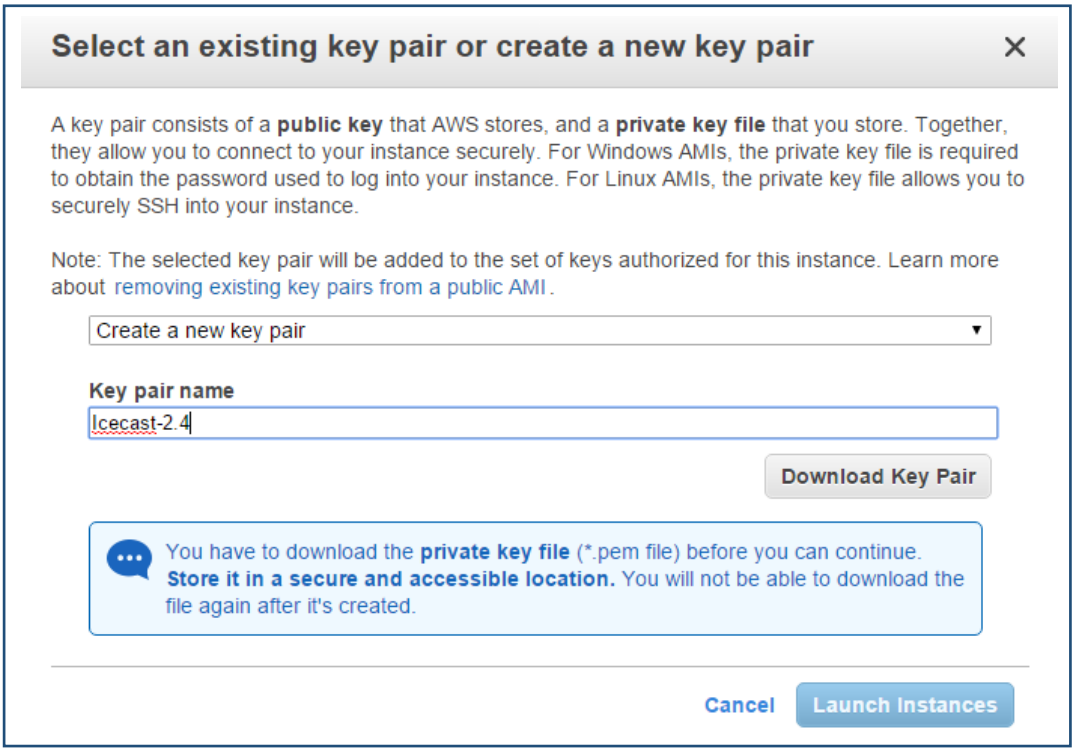
Figura 100. Revisar configuración de seguridad



Fuente: el Autor.

Al oprimir en la opción “Review and Launch”, se despliega un panel que permite crear la llave pública para hacer la conexión remota a la instancia. Es necesario tener en cuenta que este archivo debe ser celosamente almacenado, ya que es la llave para conectarse con la instancia mediante protocolo SSH.

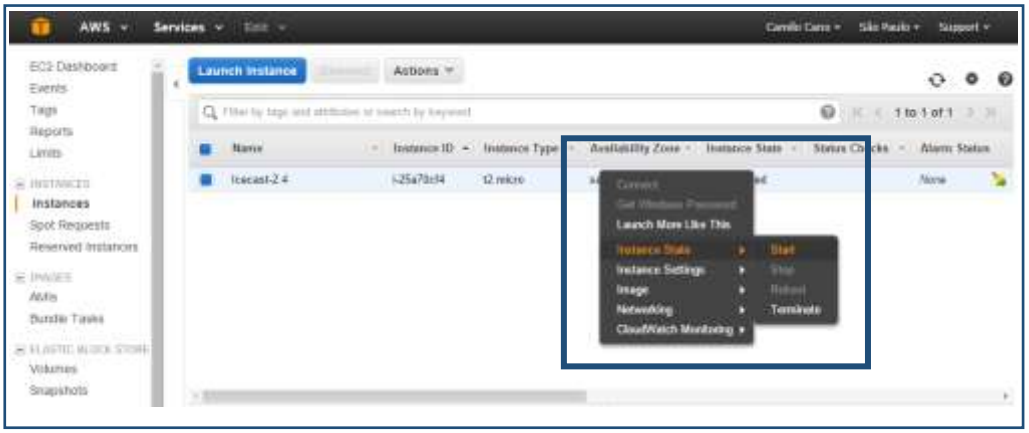
Figura 101. Crear Llave de acceso a Instancia



Fuente: el Autor.

Para realizar el lanzamiento de la maquina creada, es necesario dirigirse a la opción de “Instances”, dar clic derecho en la instancia y seleccionar Instance State => Start.

Figura 102. Iniciar instacia



Fuente: el Autor.

- **Configuración de IP elástica.**

Antes de realizar cualquier acción sobre el servidor, se debe detener la instancia, para asignar la IP elástica y así poder realizar la conexión mediante protocolo SSH para hacer público el acceso a cada servidor Icecast2.

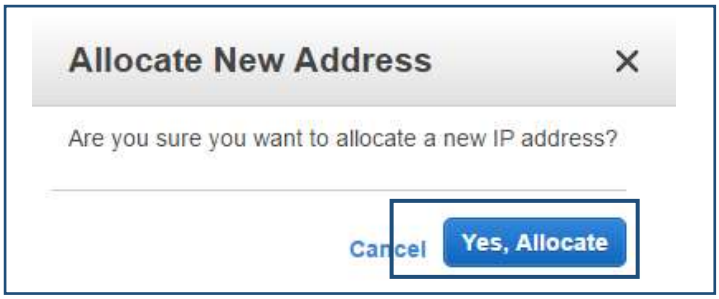
Para realizar esta configuración, se utiliza la opción Elastic IP y al ingresar a la misma se da clic en el botón “Allocate New Address”.

Figura 103. Obtener IP elástica

Fuente: el Autor.

Posteriormente se despliega una ventana que cuestiona si se está seguro que se desea adquirir una IP Elástica (ya que tiene un costo), a lo cual se debe dar que sí.

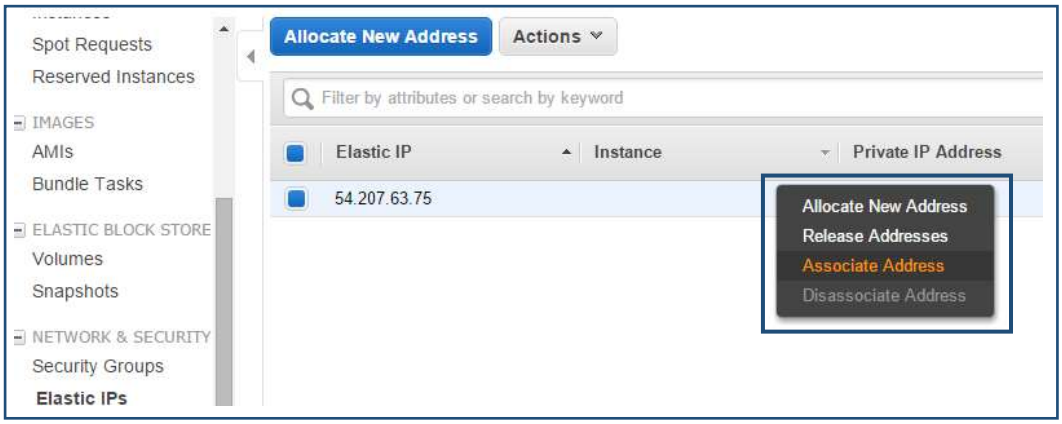
Figura 104. Confirmar IP elástica



Fuente: el Autor.

Posterior a esto, sale una confirmación de la asociación de la IP elástica a nuestra cuenta, para asignar la IP Elástica a la máquina virtual que se desea, se cierra la ventana de confirmación, se da clic derecho sobre la IP y se utiliza la opción “Associate Address”.

Figura 105. Asociar IP elástica



Fuente: el Autor.

Posteriormente se despliega una ventana de asignación, donde se da clic sobre la caja de texto “Instance”, y se podrán observar las instancias o máquinas virtuales creadas, entonces se debe escoger la que se desea hacer pública mediante la IP agregada.

Figura 106. Elegir ARN para asociar IP

Fuente: el Autor.

Al dar clic en “Asociate” se podrá observar que la asignación fue exitosa y se notara en la configuración de la IP elástica.

Figura 107. IP elástica asociada

Fuente: el Autor.

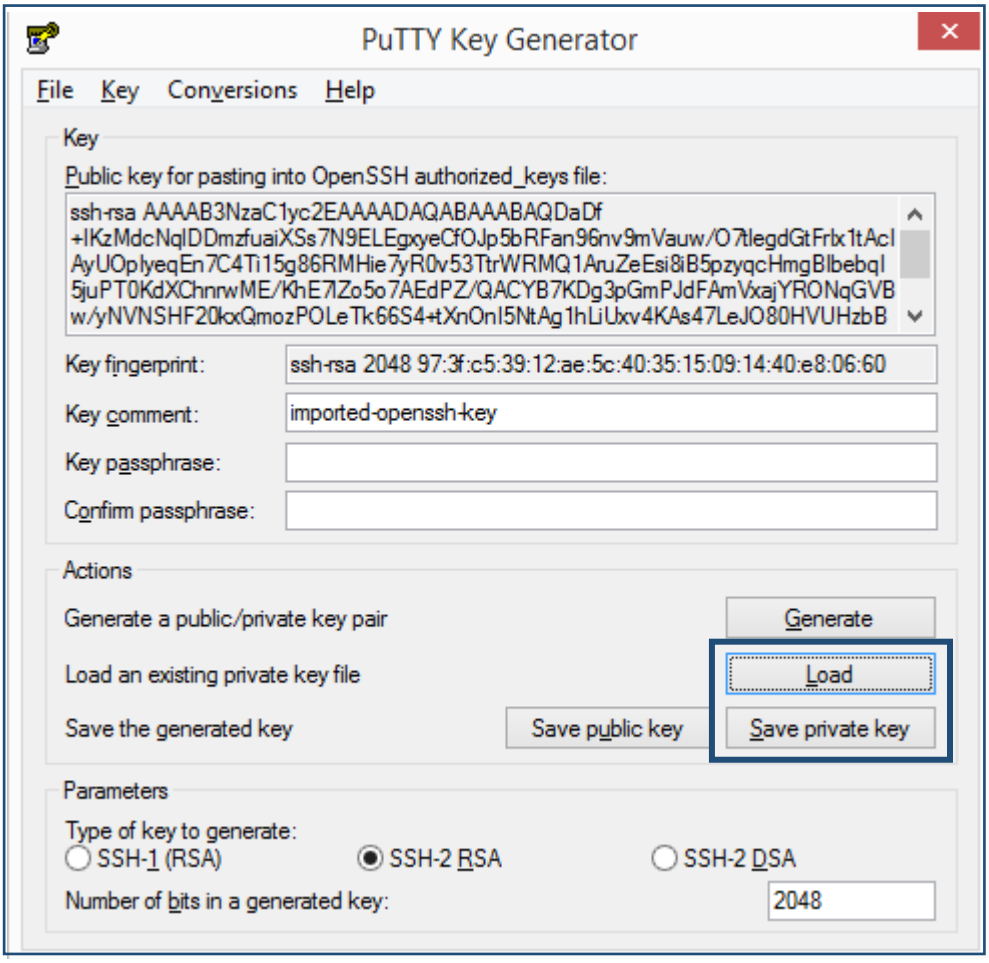
- **Conexión a instancia con Putty.**

Al ejecutar la máquina virtual se puede realizar la conexión a esta mediante dos métodos, uno web y otro mediante el cliente SSH conocido como Putty, el cual será el método utilizado.

Inicialmente se debe utilizar Puttygen, para convertir la clave de acceso de instancias con formato PEM a formato PPK, que será reconocido por el cliente SSH Putty.

Inicialmente se ejecuta Puttygen y se procede a utilizar la opción “Load” para cargar la llave en formato PEM, posteriormente se oprime el botón “save private key” para guardar la llave.

Figura 108. Cargar llave en Putty Key Gen



Fuente: el Autor.

Después de generar la clave, se puede proceder a utilizar Putty, al ejecutar el programa este solicita 3 elementos vitales para la conexión.

El primer elemento necesario es la IP elástica de la máquina virtual, este se localiza en la información general de la máquina virtual que se despliega al seleccionarla.

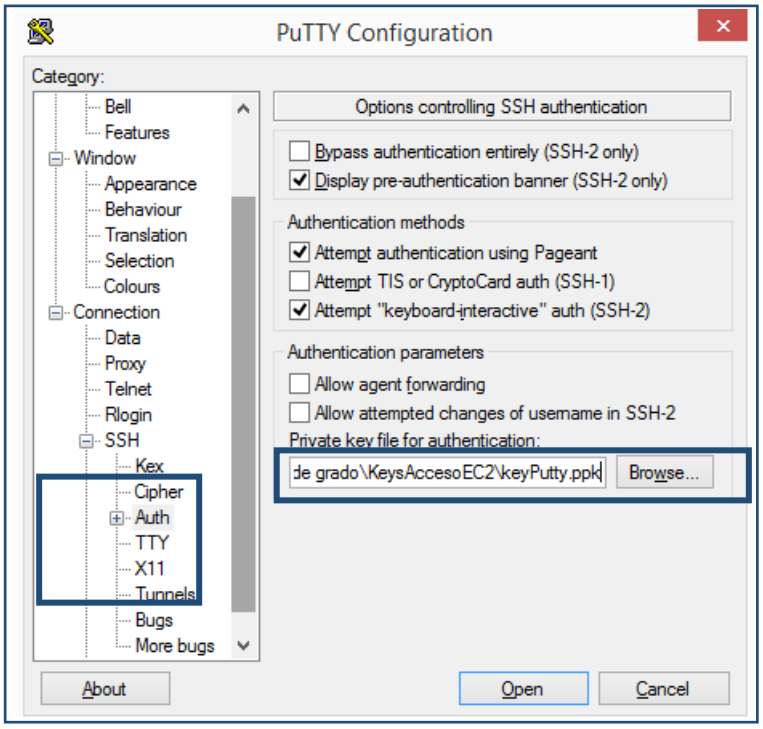
Figura 109. Observar IP elástica asignada

Fuente: el Autor.

Además de lo anterior, se necesita tener habilitado el puerto 22 en la máquina virtual y la llave privada en formato PPK.

Inicialmente se debe cargar la llave privada en Putty, esta opción se encuentra en la ruta Connection=>SSH=>Auth. Al dirigirse a esta ruta, el usuario se encuentra con una serie de opciones ya configuradas para la conexión, de este modo la única acción que debe realizar el usuario es cargar la ruta del archivo PPK, generado por Puttygen.

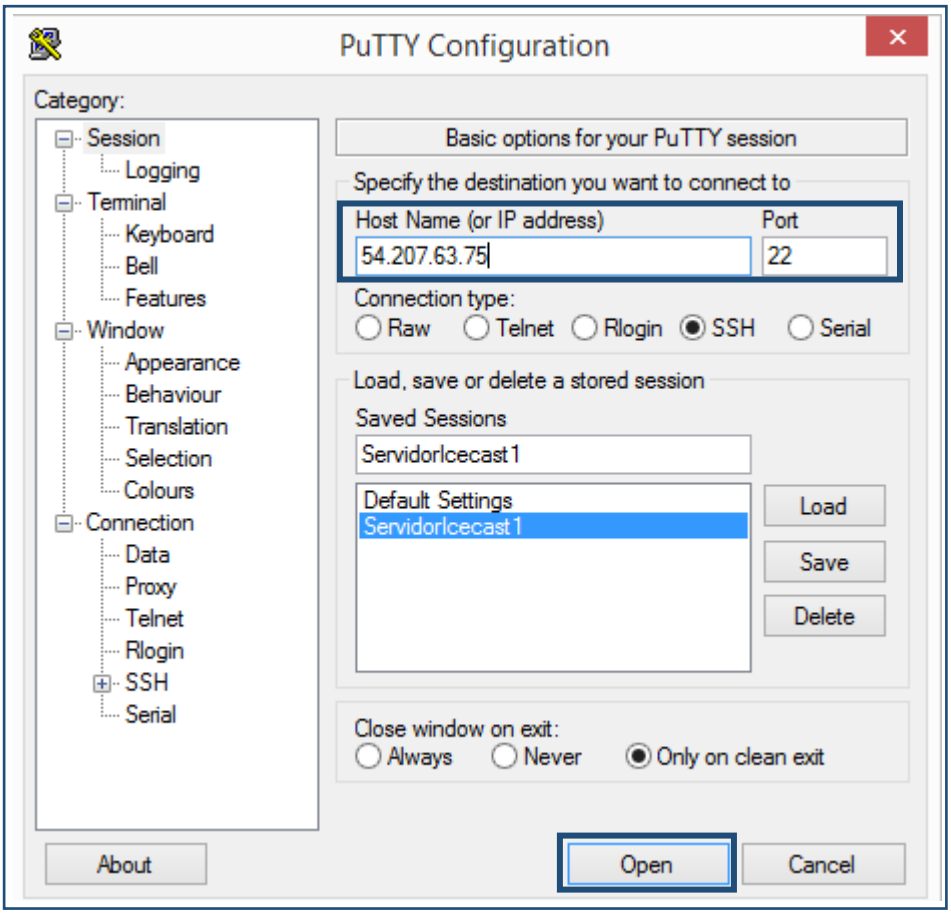
Figura 110. Cargar llave en putty



Fuente: el Autor.

Después de configurar la llave primaria, se dirige a “Session”, se diligencia el formulario con la IP Elástica del host y el puerto del protocolo SSH (22), por último, se podrá realizar la conexión con la máquina virtual haciendo clic en la opción “Open”. Es necesario tener en cuenta que el usuario del sistema operativo es “ubuntu” para todas las maquinas.

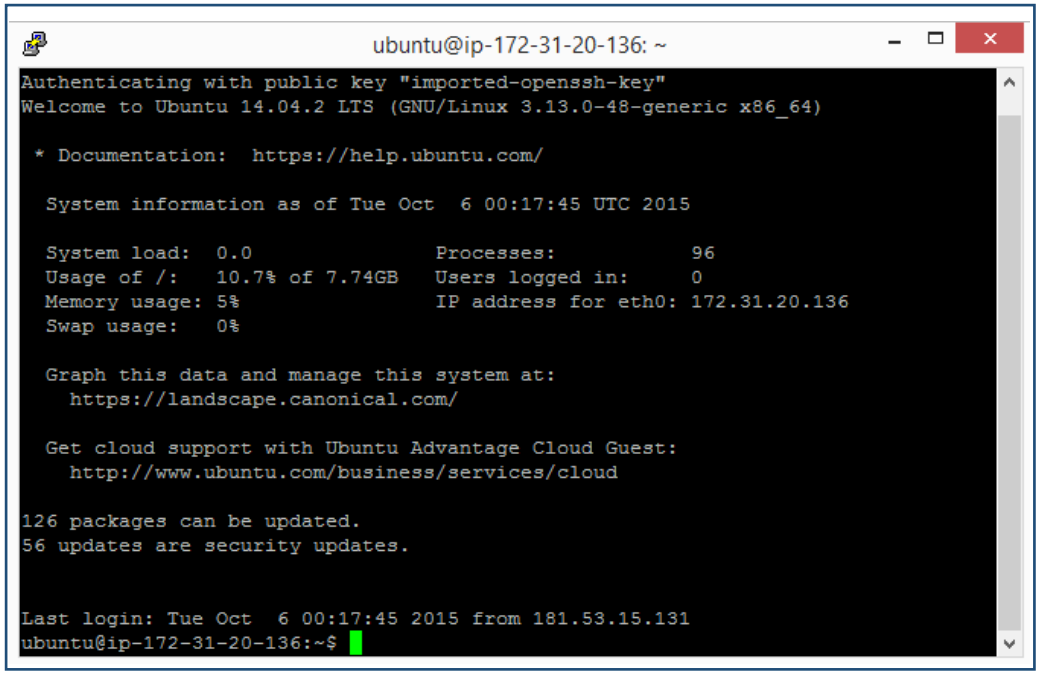
Figura 111. Definiendo datos de conexión a instancia



Fuente: el Autor.

Al establecer una conexión exitosa con la instancia mediante protocolo SSH, se podrá observar en consola las características de la instancia en ejecución. **(aws.amazon.com,2015).**

Figura 112. Conectado a instancia



Fuente: el Autor.

- **Instalación repositorios multimedia de Xiph.**

Para poder instalar la última versión de Icecast 2, es necesario agregar los repositorios oficiales para multimedia de Xiph, que se localizan en el siguiente link(https://wiki.xiph.org/Icecast_Server/Installing_latest_version_(official_Xiph_repositories), **2015**). En el anterior se podrá conocer las listas de repositorios oficiales para distribuciones Linux.

Teniendo en cuenta el tipo de AMI seleccionada, en este caso Ubuntu 14.04, se debe ejecutar una serie de comandos que a continuación se encontraran.

El comando que se verá a continuación, permite agregar la ubicación del repositorio en el archivo “Icecast.list”.

```
Sudo sh -c "echo deb
http://download.opensuse.org/repositories/multimedia:/xiph/
xUbuntu_14.04/ ./ >>/etc/apt/sources.list.d/icecast.list"
```

Posteriormente se debe proceder a configurar la llave del repositorio, se comienza por descargar el archivo “multimedia-obs.key” desde (http://icecast.org/multimedia-obs.key, **2015**).

Ahora se puede proceder a ejecutar el siguiente comando con el fichero descargado:

```
gpg multimedia-obs.key
```

Ya se puede proceder a importar la llave de repositorio con:

```
sudo apt-get add multimedia-obs.key
```

Para finalizar se actualizan los índices del sistema operativo con:

```
sudo apt-get update
```

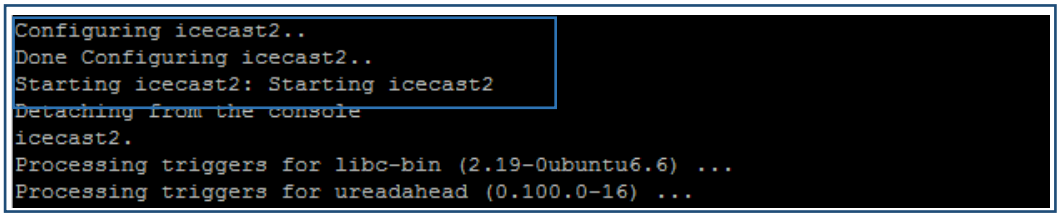
- **Instalación y configuración de Icecast2.**

Para comenzar la instalación de Icecast2 en Ubuntu, se debe ejecutar el siguiente comando:

```
sudo apt-get install icecast2
```

Para tener certeza que se instaló correctamente, se debe observar que las líneas señaladas en la figura 113 se encuentran al final de la instalación.

Figura 113. Confirmación de instalación de Icecast 2



Fuente: el Autor.

Ahora se debe detener el servicio de Icecast2 con el comando: `service icecast2 stop`, y volver a actualizar el sistema con el comando: `sudo apt-get update`.

Para proceder a configurar el servidor se debe saber que, por razones de seguridad, Icecast2 no puede ser ejecutado por el usuario root y por este motivo, se deben copiar las carpetas y archivos que permiten el funcionamiento, a la carpeta personal de un usuario sin privilegios de administración sobre el sistema operativo, en este caso se sabe que el usuario es “ubuntu”.

Para comenzar la configuración se debe determinar la ruta de instalación del fichero “Icecast.xml” con el comando:

```
Sudo find / -name “icecast.xml”
```

Al conocer la localización de este fichero, se puede ir a la carpeta que lo contiene, con el comando: `cd /PATH/icecast2`

Al encontrase en esta carpeta se va a observar el fichero icecas.xml, las carpetas admin y web que deben ser copiadas en la carpeta local del usuario “ubuntu”.

Antes de continuar, se debe utilizar el comando mkdir para crear una carpeta llamada Icecast en la ruta “/home/user”, donde se remplaza user por el usuario local.

```
mkdir /home/ubuntu/icecast
```

Ahora si se debe proceder a copiar los ficheros, cambiar el usuario y el grupo de los mismos con los siguientes comandos:

```
sudo cp icecast.xml /home/ubuntu/icecast
sudo chown ubuntu:ubuntu /home/ubuntu/Icecast/Icecast.xml

sudo cp -R admin  /home/ubuntu/icecast
sudo chown -R ubuntu:ubuntu /home/ubuntu/Icecast/admin

sudo cp -R web  /home/ubuntu/icecast
sudo chown -R ubuntu:ubuntu /home/ubuntu/Icecast/web
```

Después de copiar las carpetas y cambiarles el usuario, se puede proceder a ver los ficheros con su respectivo dueño en la carpeta local del usuario.

Ahora con el comando mkdir se va a crear la carpeta logs y dentro de esta, los ficheros error.log y access.log con el comando nano.

El archivo más importante en la configuración es el icecast.xml, por este motivo se procederá a su modificación.

Un paso vital para utilizar el servicio es configurar las contraseñas de autenticación, para realizar esto, se edita el icecast.xml que se debe copiar a la ruta del usuario, para esto se utiliza el comando:

```
nano /home/ubuntu/icecast/icecast.xml
```

Las líneas que se deben editar son las necesarias para hacer funcionar nuestro servicio. Inicialmente se determinan la cantidad de conexiones que permite cada servidor, para este caso se van a determinar en la sección “limits” una cantidad de 8 fuentes, 100 oyentes por cada una, 100x8=800, es decir 800 clientes en total. Las otras opciones de esta sección se dejan predeterminadas.

```
<limits>
<clients>800</clients>
<sources>8</sources>
<queue-size>524288</queue-size>
<client-timeout>30</client-timeout>
<header-timeout>15</header-timeout>
<source-timeout>10</source-timeout>
<burst-on-connect>1</burst-on-connect>
<burst-size>65535</burst-size>
</limits>
```

La configuración necesaria de autenticación para acceder a las funcionalidades de administración y acceso se determina con los siguientes elementos:

- Contraseña para fuentes, que permite el acceso de las fuentes en general al servidor
- Contraseña para retransmisión, esta permite el acceso de un servidor esclavo para que envié todos sus Streams.
- Administrador, especifica el usuario del log in para acceder al tablero de instrumentos.
- Contraseña de administrador, es la clave para acceder al tablero de instrumentos

```
<authentication>
<source-password>clave</source-password>
<relay-password>clave</relay-password>
<admin-user>admin</admin-user>
<admin-password>clave</admin-password>
</authentication>
```

El sistema virtual estará compuesto por 3 servidores, cada uno tendrá un total de 8 puntos de montaje, cada uno con su título y contraseña específica, junto a una configuración que contiene:

- Punto de montaje, que especifica el título del mismo.
- Clave de acceso, es única para el punto de montaje al que pertenece y permite la conexión de una fuente al mismo.
- El máximo de clientes, es necesario recordar que cada punto de montaje tiene un máximo de 100 oyentes.
- El tamaño de bufer del punto de montaje.
- En public se hace público el punto de montaje con el valor de 1 para permitir que los clientes se conecten sin contraseña.
- El bitrate es 64 kbps, que es la tasa de compresión máxima de cada paquete en 1 segundo.

A continuación, se observará un ejemplo de la configuración de un (1) punto de montaje., utilizado en el sistema

```
<mount>
<mount-name>/example.ogg</mount-name>
<password>pass_example</password>
     <max-listerners>100<max-listerners>
     <burts-size>65536</burts-size>
     <public>1</public>
     <bitrate>64</bitrate>
</mount>
```

Por último, se configuran los Paths, que son las rutas de los ficheros que requiere Icecast2 para su correcto funcionamiento. Es necesario recordar que con anterioridad se copiaron estos ficheros a la carpeta del usuario “ubuntu” y se cambiaron, el usuario y grupo al que pertenecían.

- Basedir, en esta casilla se especifica el ejecutable del servidor, se deja predeterminado.
- La dirección de la carpeta que contiene los archivos error.log y Access.log.
- La ruta donde se ubican los ficheros del panel web cliente para cada servidor.
- Por último, se especifica la ruta donde se ubican los ficheros del panel web administrador para cada servidor.

```
<paths>
   <basedir>opt/Icecast/2.4.99.1/share/icecast</basedir>
   <logdir>/home/ubuntu/icecast/logs</logdir>
   <webroot>/home/ubuntu/icecast/web</webroot>
   <adminroot>/home/ubuntu/Icecast/admin</adminroot>
</paths>
```

Posterior a hacer la configuración se deben guardar los cambios, se debe localizar la ruta en la carpeta donde se copió el Icecast.xml, en este caso `/home/ubuntu/Icecast/` y se procede a iniciar la instancia y el servicio de streaming con el comando: `Icecast -c Icecast.xml`.

Para comprobar el funcionamiento se ejecuta cualquier navegador de internet, se digita la IP elástica de la instancia junto al puerto en la caja de texto de la URL, e inmediatamente se conectará con el servidor, si la instalación y configuración se realizaron correctamente.

Figura 114. Icecast 2 correctamente instalado

Fuente: el Autor.

- **Instalación de mysql 5.x.**

Para proceder a hacer la instalación del servidor MySQL, se utilizará la instancia sobre la cual se ejecuta el servidor Icecast2 en la Zona de Sao Pablo. Para comenzar se debe iniciar la instancia y hacer la conexión mediante Putty como se ha realizado anteriormente.

Posterior a conectar con la instancia, se deben ejecutar los siguientes comandos para instalar el servidor MySQL.

```
sudo apt-get install mysql-server mysql-common
```

- **Instalación de tomcat 7.**

La instalación de tomcat7 se realizará en la instancia de la zona de Sao Pablo, antes esta se debe iniciar y se actualizarán los índices con el comando `apt-get update`. Posteriormente se debe proceder a instalar tomcat7 con el siguiente comando:

```
sudo apt-get install tomcat7
```

Posteriormente se debe instalar el jdk de JAVA con el comando `sudo apt-get install default-jdk`.

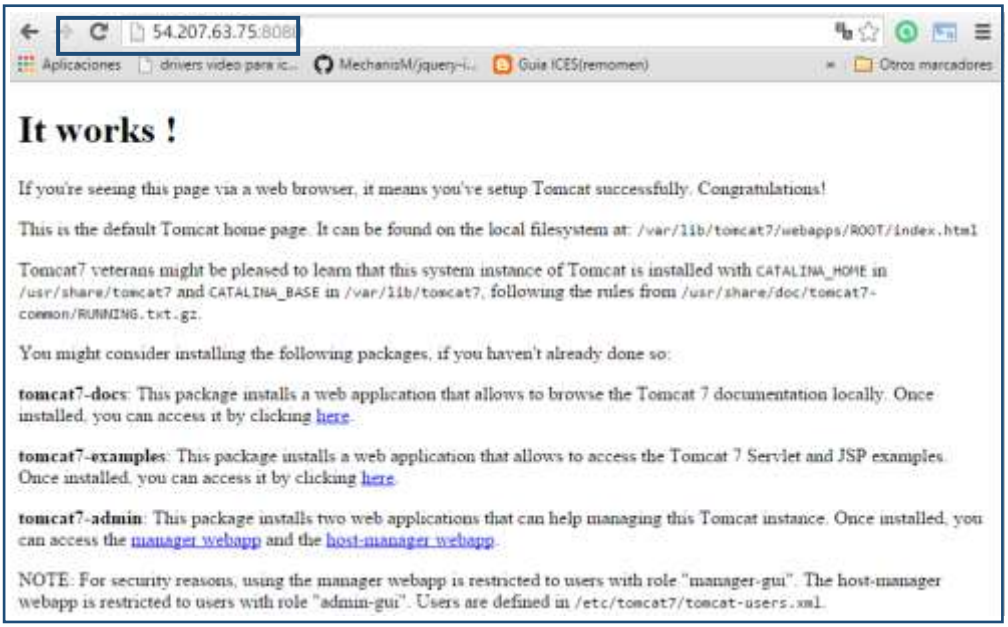
Ahora se debe proceder a editar el archivo bashrc con el comando `sudo nano ~/.bashrc`. Al editar el archivo, se deben agregar las siguientes líneas al final del archivo:

```
export JAVA_HOME=/usr/lib/jvm/default-java
export CATALINA_HOME=/var/lib/tomcat7
```

Por último se recarga el bash con el comando: `~/.bashrc`, y se puede proceder a reiniciar el servicio con: `sudo service tomcat7 restart`.

Para comprobar el funcionamiento, se despliega un navegador y se ingresa la IP de la instancia de Sao Pablo junto con el puerto 8080 por el cual escucha tomcat7.

Figura 115. Inicio de Tomcat en navegador



Fuente: el Autor.

- **Implementación y configuración de tecnologías para proyecto web.**

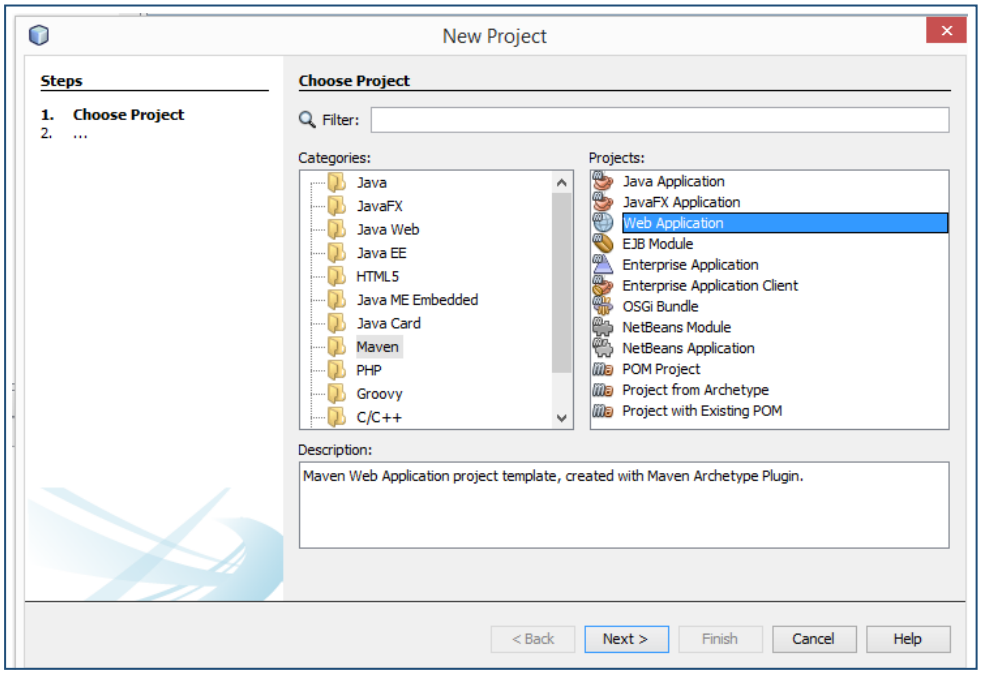
Para desarrollar el proyecto eficientemente se van a usar librerías y frameworks conocidos en los diferentes componentes web del sistema, que son:

- **Back end**
  - Maven
  - Hibernate
  - EasyRest
- **Aplicación web cliente**
  - Bootstrap
  - JQuery

Creación de proyecto Maven.

Para comenzar, se debe crear el proyecto Maven que cuenta con un repositorio que facilita la importación de artefactos, es decir librerías junto a todas las dependencias requeridas para funcionar correctamente sin tener que realizar la importación manual de los JAR. Se ejecuta Netbeans y se crea el proyecto en: File->New Project->Maven->Web Application.

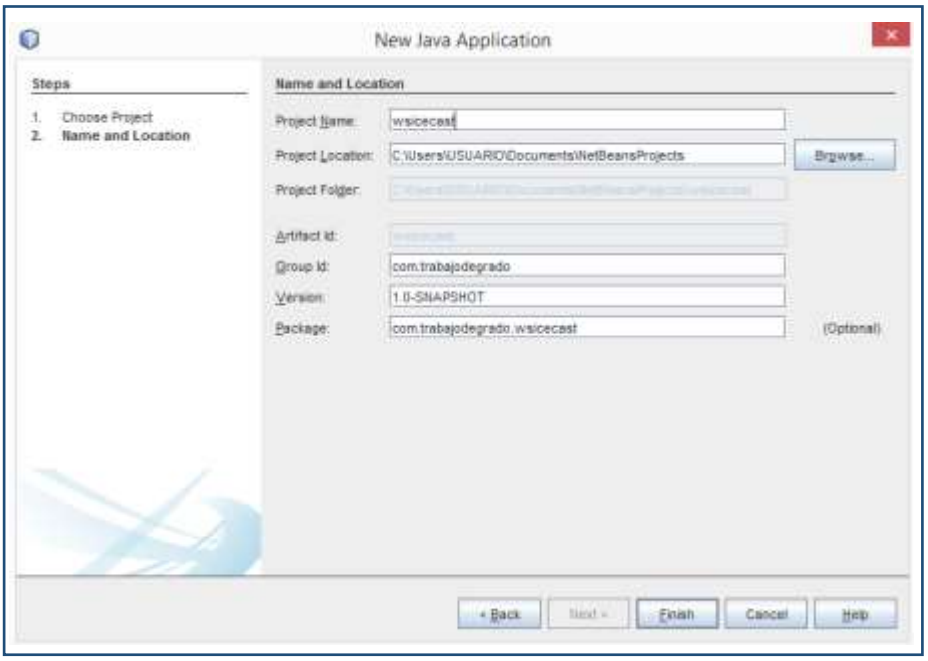
Figura 116. Lanzar proyecto Maven



Fuente: el Autor.

Para continuar, se debe definir el nombre del proyecto que será “WSIcecast” y el servidor que se va a utilizar, en este caso será Apache Tomcat EE 8 para el entorno local, que viene incluido en la instalación de Netbeans 8.

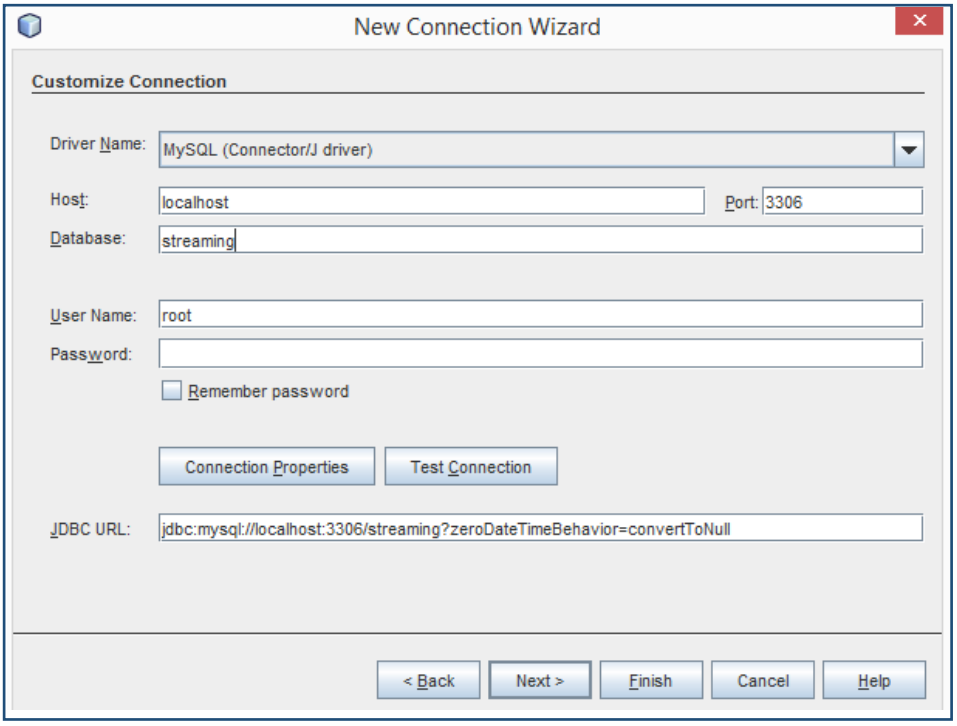
Figura 117. Configurar proyecto Maven



Fuente: el Autor.

Antes de proceder a configurar hibérnate, es necesario garantizar que la conexión a la base de datos está establecida, esto se hace mediante la opción services->Databases->New Connection. Al utilizar esa opción, se despliega un panel en el cual se deben especificar los datos de la conexión para el proyecto.

Figura 118. Conexión a MySQL



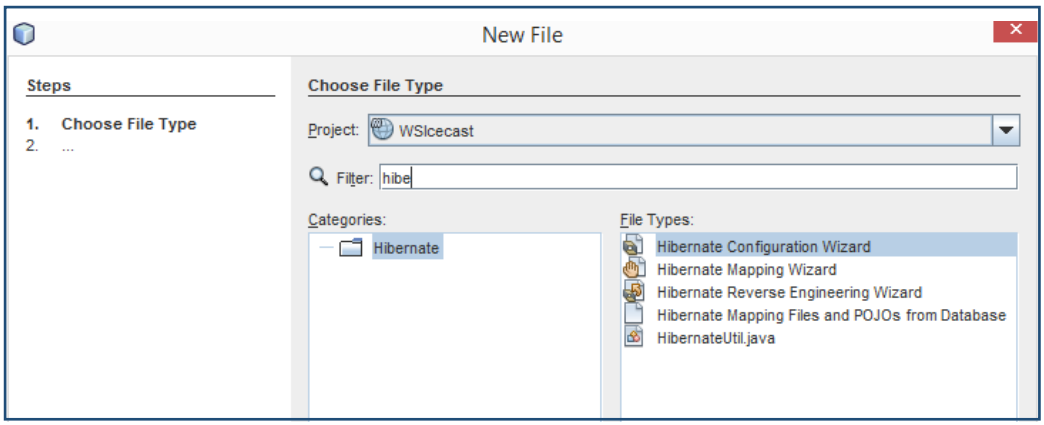
Fuente: el Autor.

Instalación y configuración de Hibernate y POJOs .

Hibernate es un framework JAVA, que permite mapear bases de datos relacionales de diferentes gestores como Oracle DB, MySQL o Postgresql, además facilita la realización de consultas mediante un lenguaje propio llamado HQL que soporta consultas para motores anteriormente mencionados.

Después de crear el proyecto y garantizar la conexión a la base de datos, se debe mapear está utilizando Hibernate. Primero se crea el asistente de configuración y el archivo hibernate.cfg.xml, para esto, se va al proyecto creado WSIcecast, se da clic derecho sobre el mismo y se utiliza la opción New->Other->Hibernate Configuration Wizard.

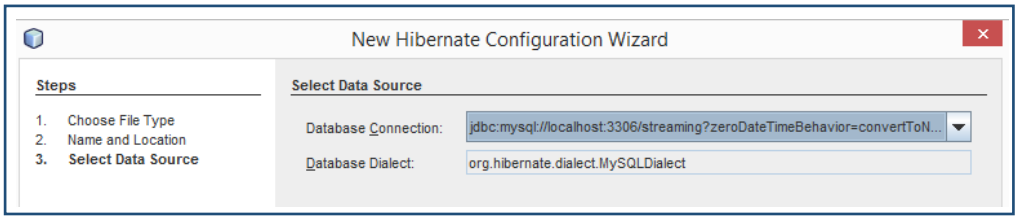
Figura 119. Creando Hibernate wizard



Fuente: el Autor.

Después de crear el asistente, se debe definir la conexión a la base de datos que se va a utilizar, se selecciona la que fue creada con anterioridad.

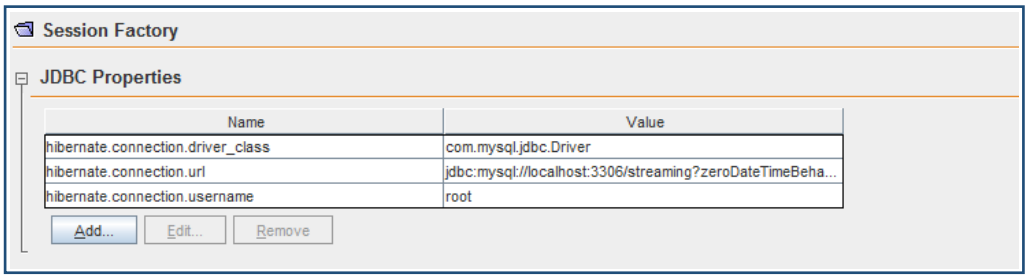
Figura 120. Definiendo conexion MySQL



Fuente: el Autor.

Después de crear el asistente, se debe proceder a observar el panel grafico que aparece instantáneamente.

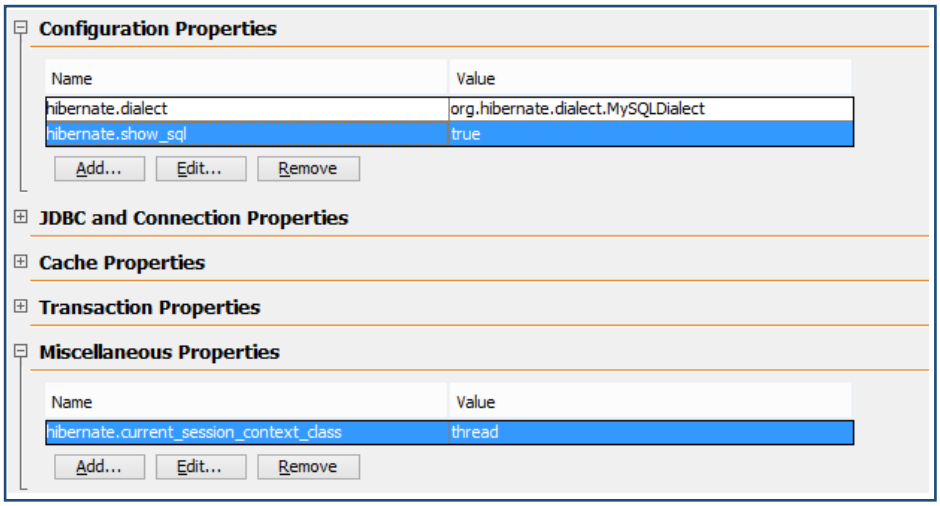
Figura 121. Interfaz gráfica de configuración Hibernate



Fuente: el Autor.

En el panel grafico se va a agregar la propiedad de show_sql y current_session_context_class para poder ve en consola las consultas construidas en SQL.

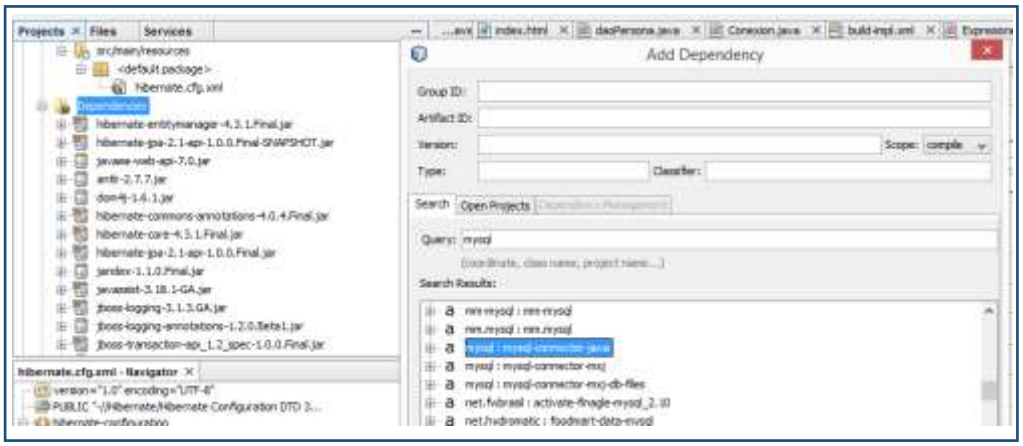
Figura 122. Propiedades de Hibernate agregadas en interfaz grafica



Fuente: el Autor

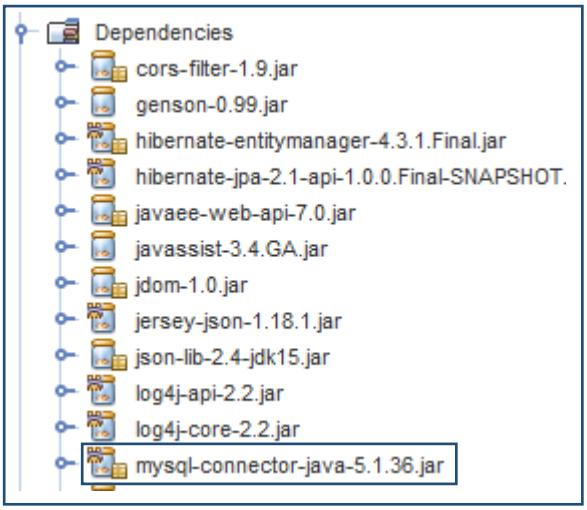
Después de modificar las propiedades, se debe proceder a agregar el conector de mysql en las dependencias Maven, para esto, se da clic derecho en dependencias->Add Dependency y se agrega el conector.

Figura 123. Agregando conecto MySQL



Fuente: el Autor.

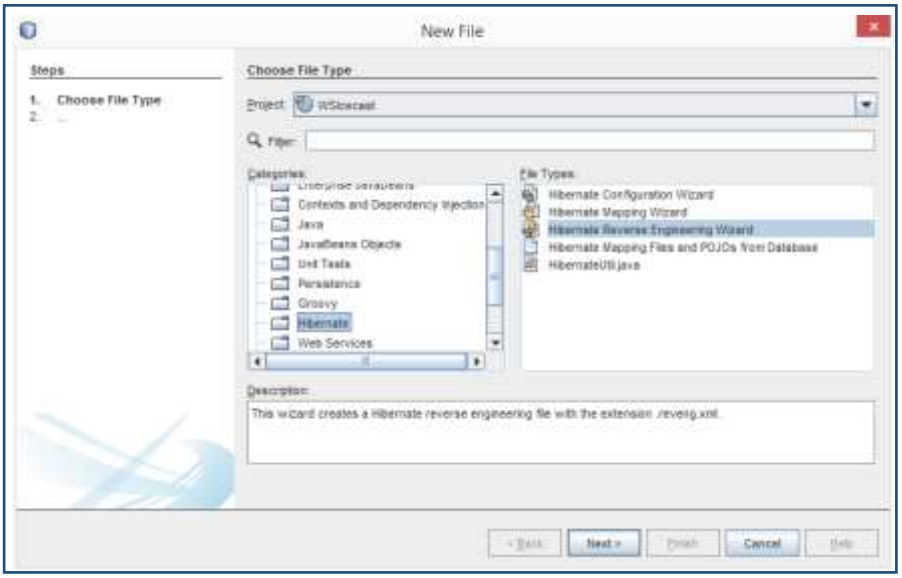
Figura 124. Dependencias Maven



Fuente: el Autor

Al agregar el conector de MySQL, se puede proceder a crear el fichero hibernate.reveng.xml, que permite realizar la ingeniería inversa a las tablas de la base de datos, necesaria para crear los POJO, que son las clases que representan cada una en el sistema y son utilizados para construir el HQL.

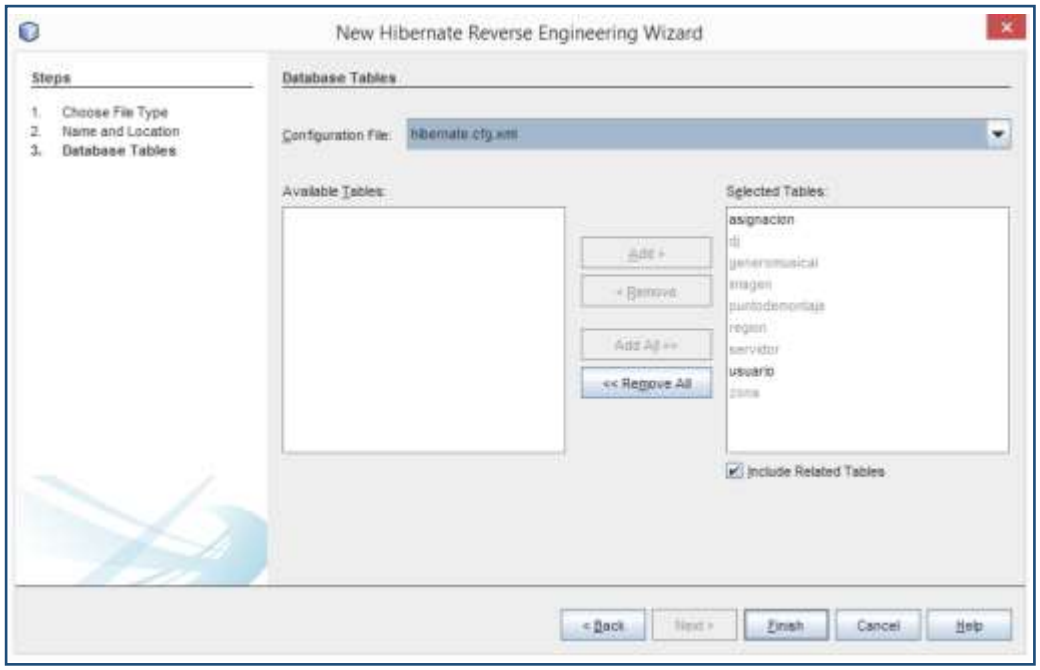
Figura 125. Creando fichero de ingeniería inversa



Fuente: el Autor.

Después de escoger el archivo, se debe poder observar las tablas de la base de datos para agregarlas al mapeo inverso.

Figura 126. Agregando tablas



Fuente: el Autor.

Después de adicionar las tablas y crear exitosamente el fichero hibernate.reveng.xml, se puede proceder a observar las tablas en el mismo.

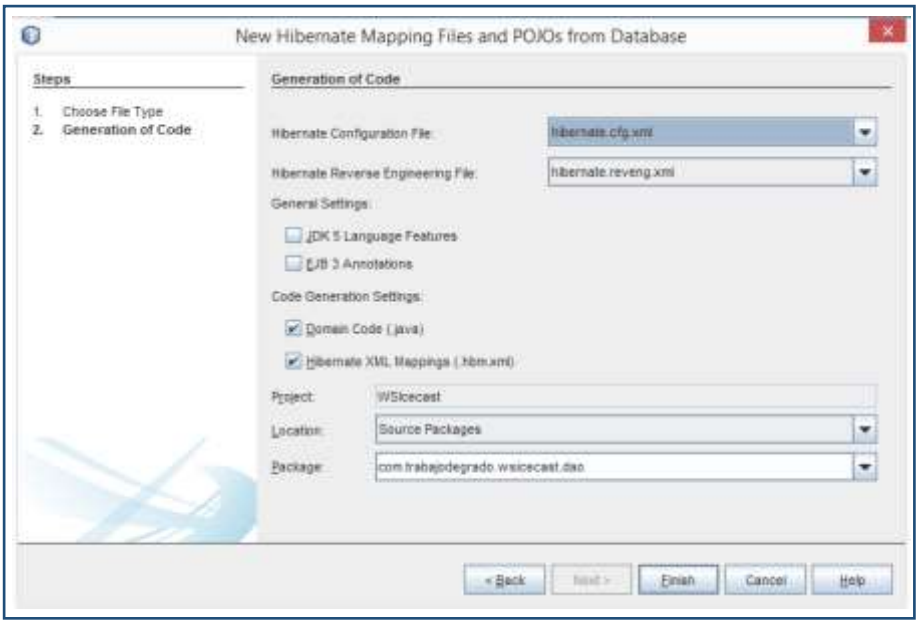
Figura 127. Fichero de ingeniería inversa en XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-reverse-engineering PUBLIC "-//Hibernate/Hibernate Reverse Engineering DTD 3.0//EN"
<hibernate-reverse-engineering>
  <schema-selection match-catalog="streaming"/>
  <table-filter match-name="imagen"/>
  <table-filter match-name="puntodemontaje"/>
  <table-filter match-name="asignacion"/>
  <table-filter match-name="dj"/>
  <table-filter match-name="generomusical"/>
  <table-filter match-name="region"/>
  <table-filter match-name="servidor"/>
  <table-filter match-name="usuario"/>
  <table-filter match-name="zona"/>
</hibernate-reverse-engineering>
```

Fuente: el Autor.

Ahora, para proceder a crear los POJOs, se va a crear un nuevo paquete llamado “pojo” y en este se generan, para esto, se da clic derecho en el paquete creado y se procede a la ruta New->Other->Hibernate Mapping Files and POJOs from Database.

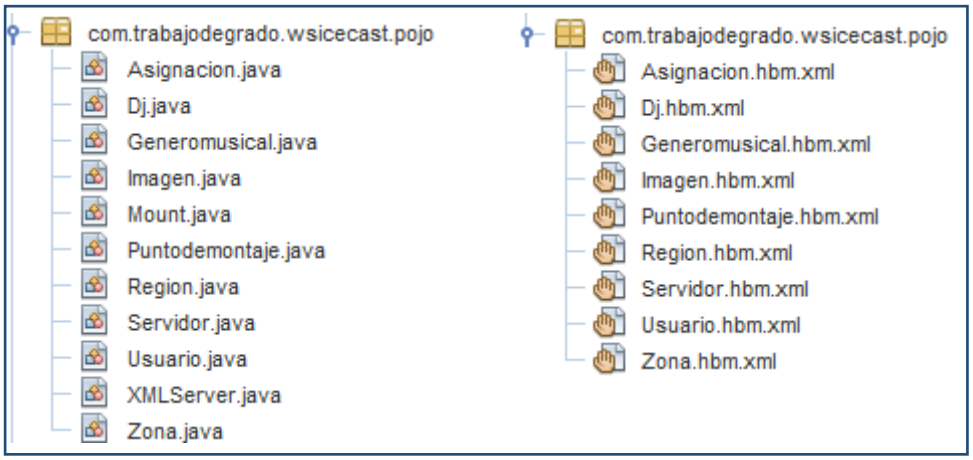
Figura 128. Opción para mapear las tablas



Fuente: el Autor.

Ya se puede proceder a ver la lista de POJOs relativos a las tablas de la BD con sus respectivos mapas en XML.

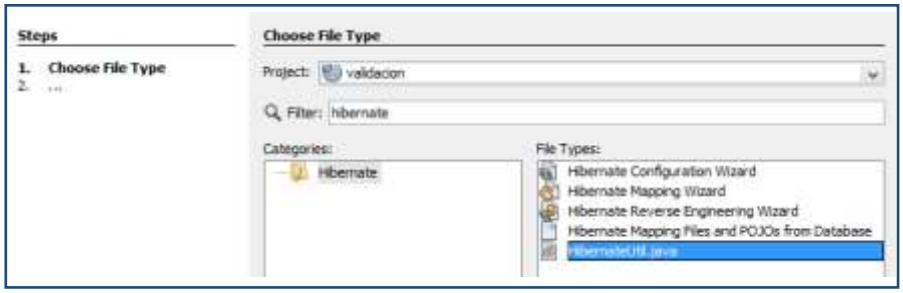
Figura 129. Ficheros de tablas mapeadas



Fuente: el Autor.

Después de realizar el mapeo inverso de la base de datos, se importa el archivo HibernateUtil.java, que cumple la labor de ser una fábrica de sesiones para realizar las consultas a la base de datos. Inicialmente se crea otro paquete llamado “dao” y se importar la clase HibernateUtil, haciendo clic derecho sobre el proyecto y dirigiéndose a la ruta New->Other->HibernateUtil.java.

Figura 130. Crear Hibernate Util



Fuente: el Autor

Instalación y configuración de RestEasy.

RestEasy en una librería JAVA, que facilita la publicación de servicios RestFul a partir de POJOs y clases mapeadas mediante la notación JAXB.

Maven posee un archivo de configuración llamado pom.xml, en este se configuran las dependencias que son utilizadas en el proyecto, para iniciar, se deben importar las dependencias requeridas de RestEasy, pero antes de la importación, se debe configurar la conexión al repositorio de JBOSS.

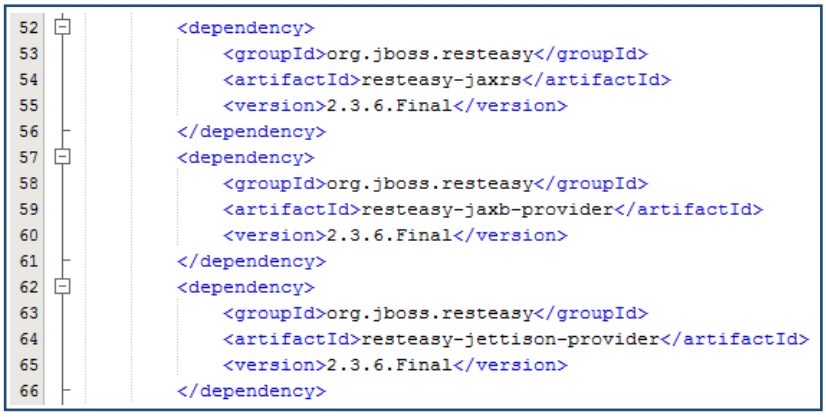
Figura 131. Repositorios de JBOSS en pom.xml

```
<repositories>
    <repository>
        <id>jboss-public-repository-group</id>
        <name>JBoss Public Repository Group</name>
        <url>http://repository.jboss.org/nexus/content/groups/public/</url>
    </repository>
    <repository>
        <id>unknown-jars-temp-repo</id>
        <name>A temporary repository created by NetBeans for libraries and jars it could not identify.
        <url>file:${project.basedir}/lib</url>
    </repository>
</repositories>
```

Fuente: el Autor.

A continuación, se muestran las dependencias que deben ser agregadas al pom.xml para importar RestEasy al proyecto.

Figura 132. Dependencias de RestEasy en pom.xml



```
<dependency>
    <groupId>org.jboss.resteasy</groupId>
    <artifactId>resteasy-jaxrs</artifactId>
    <version>2.3.6.Final</version>
</dependency>
<dependency>
    <groupId>org.jboss.resteasy</groupId>
    <artifactId>resteasy-jaxb-provider</artifactId>
    <version>2.3.6.Final</version>
</dependency>
<dependency>
    <groupId>org.jboss.resteasy</groupId>
    <artifactId>resteasy-jettison-provider</artifactId>
    <version>2.3.6.Final</version>
</dependency>
```

Fuente: el Autor.

Después de configurar el archivo pom.xml con las dependencias, se debe crear el archivo web.xml para configurar el despliegue de la aplicación. En este se puede configurar como el nombre y el punto raíz de nuestra aplicación, cuándo se encuentra en ejecución; también se define el dispacher que se utiliza para distribuir las peticiones y el tiempo que debe durar cada sesión.

Figura 133. Configuración de despliegue de aplicación

Fuente: el Autor.

Después de configurar el despliegue, se puede proceder a configurar las clases que servirán de modelo para los elementos del servicio web. Esto se hace con la notación JAXB de java, que permite definir qué elementos se utilizaran como modelos XML para la entrega de los datos. Esta implementación es requerida para hacer la publicación, ya sea en formato XML o JSON.

2. Anexo de instalación de entorno de aplicaciones cliente

En este anexo se podrá conocer la instalación y configuración de las dependencias requeridas para configurar correctamente el entorno de desarrollo de las aplicaciones cliente.

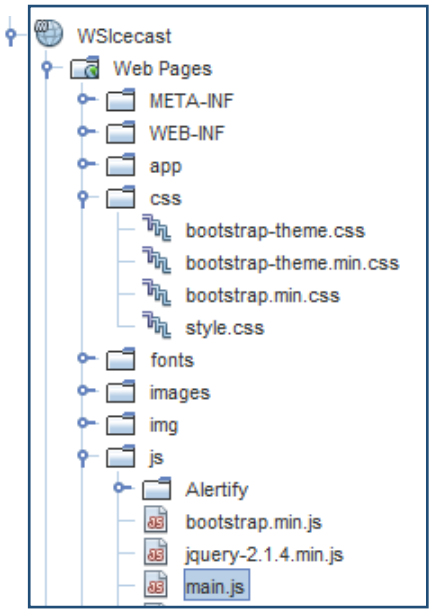
- **Instalación y configuración de Bootstrap y JQuery.**

Bootstrap es un framework de CSS que facilita el maquetado de aplicaciones HTML5 mediante una estructura tipo malla, que permite la adaptación de la aplicación de front end a los diferentes tamaños de pantallas que existen hoy en día, además de poseer una estructuración visual predeterminada de los elementos más utilizados como en la maquetación.

Teniendo en cuenta el estándar moderno de una aplicación HTML5, donde se divide la presentación de la estructura y la lógica, se debe proceder a cargar los archivos requeridos que son: **bootstrap.min.css y bootstrap-theme.min.css** en la capa de presentación.

Después de incluir los archivos CSS de bootstrap, se deben agregar los ficheros JavaScript para su correcto funcionamiento, además de agregar la librería de JQuery, que facilita un método para la interacción con el DOM mediante identificadores y atributos, este también es el encargado de hacer las peticiones al servidor con métodos AJAX, Post y Get.

Figura 134. Frameworks en proyecto HTML5



Fuente: el Autor.

Por último, se deben agregar los links de los ficheros CSS en el encabezado de la estructura HTML con la etiqueta link y los JS antes de finalizar el cuerpo con la etiqueta script, en todos los archivos HTML que hagan uso del framework.

Figura 135. Enlaces CSS en proyecto HTML5

```
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <meta name="viewport" content="width=device-width, user-scalable=no, initial-scale=1.0, maxi
    <link rel="stylesheet" href="css/bootstrap.min.css">
    <link rel="stylesheet" href="css/style.css">
    <link href="js/Alertify/alertify.bootstrap.css" rel="stylesheet" type="text/css"/>
    <link href="js/Alertify/alertify.core.css" rel="stylesheet" type="text/css"/>
    <link href="js/Alertify/alertify.default.css" rel="stylesheet" type="text/css"/>
</head>
```

Fuente: el Autor.

Es importante recordar que se debe agregar JQuery al proyecto junto al archivo principal de javascript y definir su respectivo link como se muestra a continuación.

Figura 136. Enlaces JS en proyecto HTML5

```
    <script src="js/jquery-2.1.4.min.js"></script>
    <script src="js/bootstrap.min.js"></script>
    <script src="js/jquery.jplayer.min.js"></script>
    <script src="js/Alertify/alertify.min.js"></script>
    <script src="js/main.js"></script>
</body>
```

Fuente: el Autor.

- **Instalación de Apache Cordova**

Para definir un proyecto Android basado en HTML5 es necesario implementar un framework que, mediante webview soporte HTML, CSS y JS, además de permitir que el último se comunique con la plataforma nativa para potencializar su funcionalidad haciendo uso de los sensores y los periféricos, para esto se utilizará apache Cordova.

Antes de iniciar con la instalación, es necesario tener en cuenta que el sistema operativo debe tener instalado Node JS. Posteriormente a garantizar que el sistema cuenta con esta librería, se va a proceder a ejecutar el comando: `npm install -g cordova.`