75.61 – Taller de Programacion III

Trabajo Práctico 2: Online Radio

1er cuatrimestre 2018

1ra entrega – 12/04/18

Padron: 95470

Nombre: Gabriel Gayoso

Email: ga-yo-so@hotmail.com

Facultad de Ingeniería

Universidad de Buenos Aires

Introducción

Este trabajo consiste en el diseño, desarrollo y testeo de un sistema de Radio Online. Como base se tienen una serie de requerimientos que se deben cumplir:

- Requerimientos Funcionales:
    - Se debe proveer acceso vía internet a ciertos programas radiales.
    - Los programas de radio son en vivo y los usuarios pueden decidir escuchar uno o varios a la vez mediante distintos dispositivos.
    - Existe un límite de 3 canales de radio conectados por usuario, siendo que se considera proveer un servicio ilimitado en el futuro para usuarios *premium*.
    - Las estaciones de radio pueden enviar el contenido de la emisión al sistema, y así disponibilizarlo al público.
    - Los administradores del sistema pueden consultar estadísticas de uso, incluyendo:
        - Cantidad de usuarios conectados por radio.
        - Usuarios con mayor cantidad de horas de reproducción.
- Requerimientos No Funcionales:
    - Se estima una cantidad de usuarios concurrentes muy elevada en todo momento.
    - Debido al reducido mercado de radios, se espera contar con una cantidad reducida de emisoras.
    - Se debe almacenar una entrada de log indicando conexión y desconexión de cada usuario a una determinada radio.
    - La programación debe permitir la distribución de sus componentes y la comunicación de información mediante colas persistentes.
    - El monitoreo de los flujos de información es clave para garantizar la performance del sistema.

Objetivos

La naturaleza del negocio planteado es de constante cambio. Su funcionamiento gira alrededor de una base de usuarios que probablemente este en constante crecimiento, por lo que el sistema debe ser escalable.

Las reglas del negocio no están completamente definidas. En un futuro pueden surgir nuevas oportunidades: usuarios pagos, escucha de transmisiones archivadas, etc. Por eso el sistema también necesita ser flexible a cambios futuros. Para que esto funcione además de flexibilidad en el diseño es necesario contar con documentación completa y explicativa del estado del sistema.

Obviamente para que la base de usuarios crezca es necesario proveer facilidad y velocidad en el uso del sistema. Si bien esto está muy vinculado a la aplicación Cliente que no es el foco del trabajo, se debe

garantizar la robustez ante fallas de todo el sistema para que no se perjudique la experiencia del usuario a causa de esto.

Para lograr la mejor claridad en la visualización y análisis de flujos de información en el sistema, se optó por utilizar en el mismo el servicio de colas y mensajería RabbitMQ.

Diseño

A partir de los requisitos y objetivos planteados, es necesario traducir esto a una serie de funcionalidades a desarrollar, que serán brindadas a los agentes externos al sistema que serán sus usuarios, las radios que transmitan, y los administradores que analicen las estadísticas pedidas.

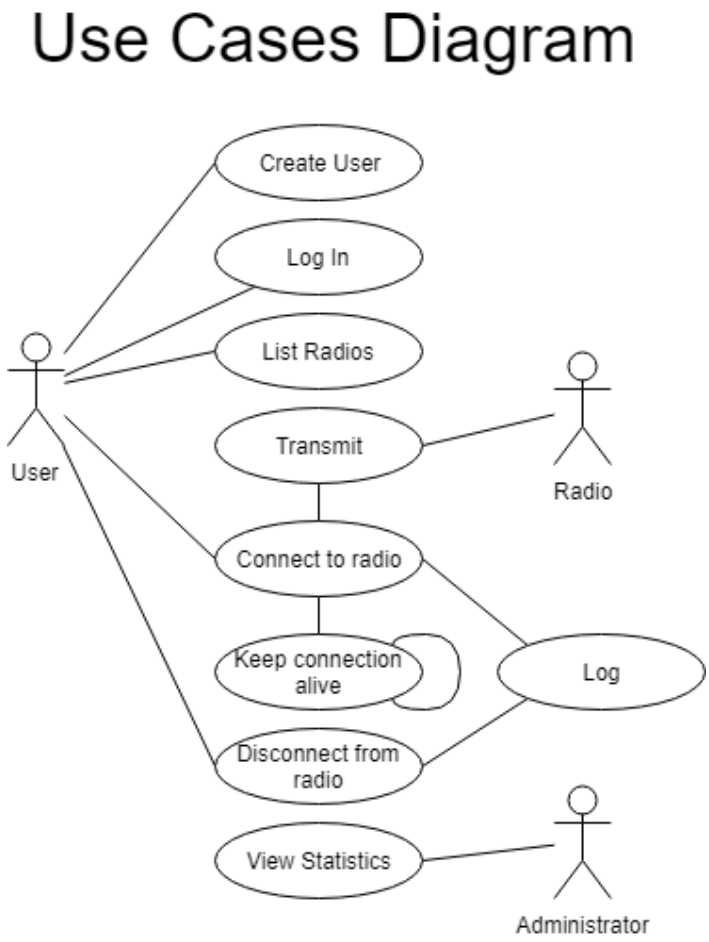

Figura 1: diagrama de casos de uso

En el diagrama de casos de uso podemos ver estas funcionalidades. El diagrama es bastante auto explicativo, las acciones (y el flujo normal) que realizará el usuario serán: registrarse, iniciar sesión, pedir una lista de las radios disponibles, conectarse a una radio y desconectarse de una radio. Además se puede ver una funcionalidad interna que valía la pena exponer: mantener viva la conexión a una radio. Dependiendo de la estrategia de resolución esto puede ser automático, pero vale la pena destacar su

importancia en un sistema donde el correcto registro de las conexiones activas de un usuario determina la posibilidad o no de nuevas conexiones.

Analizando en profundidad estas funcionalidades, se puede empezar a distinguir responsabilidades separables en ellas:

- Es necesario registrar y distinguir las distintas acciones del cliente para despachar a quien corresponda las acciones internas a llevar a cabo en respuesta.
- Es necesario mantener un registro de usuarios, sus conexiones y el estado de vida de cada una.
- Es necesario mantener un registro de radios y las conexiones a las mismas.
- Es necesario establecer una conexión entre las radios y los usuarios para poder transmitir.
- Es necesario obtener y hacer disponible información sobre el estado del registro de usuarios y el registro de radios.

Esta división inicial se profundizo un poco más para permitir que el sistema sea lo más separable, distribuible y escalable posible. De esta forma se llega a los diagramas que se presentan a continuación.

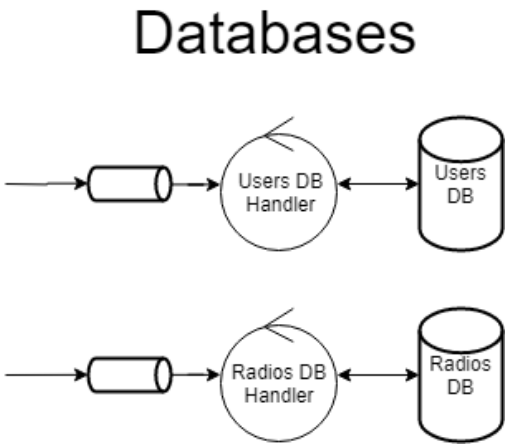

Figura 2: bases de datos de radios y usuarios

En principio resulta útil definir las dos entidades que almacenan el estado del sistema. El objetivo de esto es simular la presencia de una base de datos distribuida, accesible desde distintos nodos en la red. Por esta razón no es suficiente solo con un archivo sino que es necesaria la presencia de un manejador escuchando pedidos para interactuar con el archivo local.

Los registros de la base de usuarios tienen la forma:

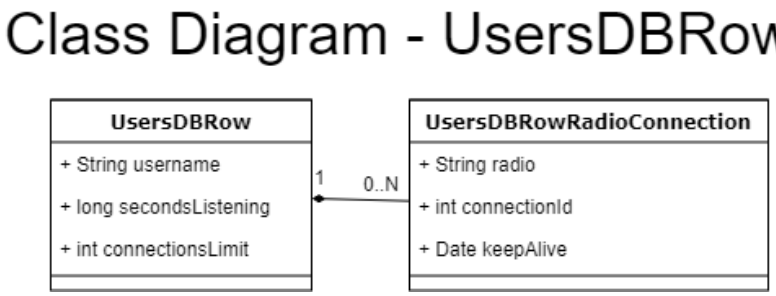

Figura 3: registros de base de datos de usuarios

Como se puede ver, se almacena una fila por usuario con los segundos que lleva escuchando y cuantas conexiones tiene permitidas (de modo que a futuro pueda cambiarse). Además se mantiene una lista de conexiones donde para cada una se guarda el nombre de la radio, el id de conexión para diferenciar entre distintos dispositivos, y la fecha en que se recibió el ultimo pedido de “keep alive”.

Las tareas del manejador de base de usuarios frente a una conexión entonces son:

- Si no existe el usuario, crearlo.
- Recorrer las conexiones y eliminar aquellas cuyo keep alive no sea reciente (configurable).
- Si queda lugar, establecer la nueva conexión.

Los registros de la base de radios tienen la forma:

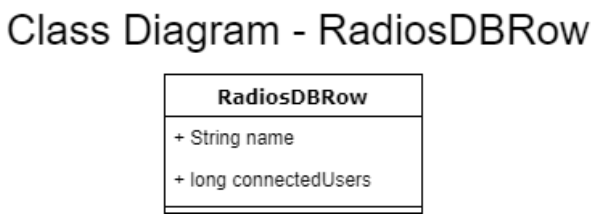

Figura 4: registros de base de datos de radios

Al momento de redactar este informe no hay un sistema de descubrimiento de radios y no se ingresan nuevas radios al iniciar o frenar ellas una transmisión, sino cuando un usuario intenta conectarse a una de ellas. Para conectarse un usuario debe conocer de antemano el nombre de una radio, o probar nombres. En caso de no existir, el cliente igual puede quedarse escuchando nada (hasta que una radio con ese nombre comience a transmitir). Algo así como la sintonización en las radios tradicionales.

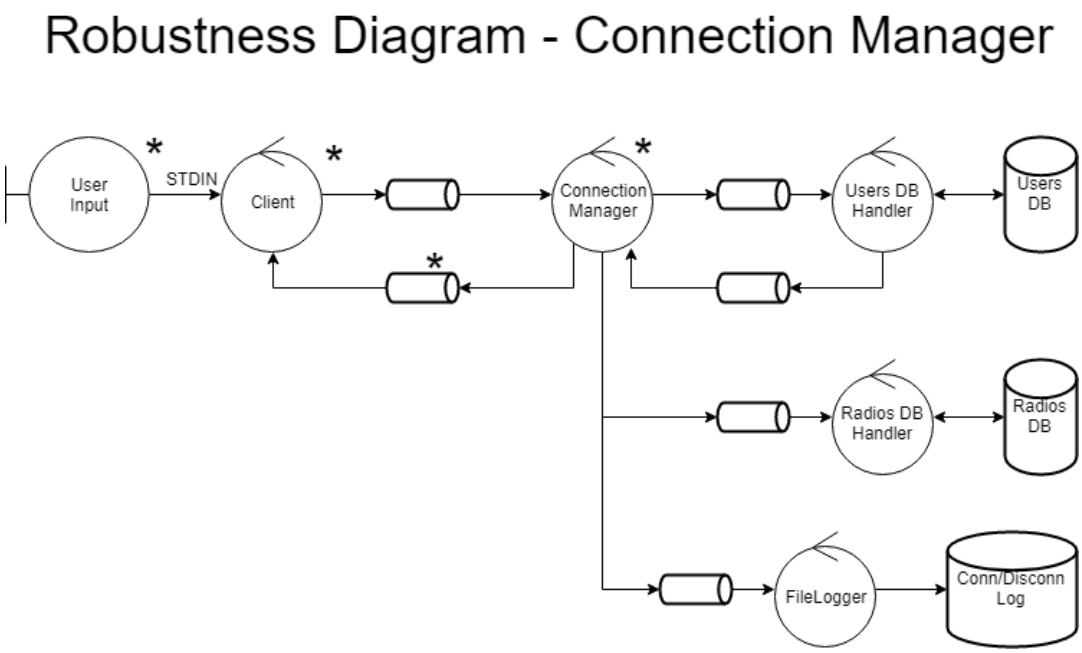

Figura 5: diagrama de robustez – rama de Connection Manager

La acción comienza con entrada del usuario, que elige a través del Client conectarse a una radio. Este proceso es el punto de acceso del usuario al sistema. Es una simple aplicación por consola que permite al usuario participar de los casos de uso descriptos anteriormente.

El proceso Client inicia la conexión a la radio encolando el pedido en una cola única de conexiones, y se queda esperando en una cola temporal el resultado del pedido.

De esta cola de conexiones consumen mensajes muchos Connection Managers. Frente a un pedido de conexión pasan el mismo al manejador de base de datos y esperan asincrónicamente una respuesta en una cola única (puede llegarle a otro Connection Manager distinto). El manejador de base de datos realiza sus tareas (que incluye el limpiado de conexiones viejas) y devuelve el resultado de toda la operación.

Si la conexión fue exitosa el Connection Manager avisa al manejador de base de datos de radios para que se actualice, al log para que la registre, y al cliente para que comience a escuchar la radio. Caso contrario se avisa al cliente que ya se lleno el límite de conexiones permitido para la cuenta, o el error que haya ocurrido. En cualquier caso se avisa al manejador de bases de radio y al file logger sobre cualquier conexión vieja cancelada en el paso anterior.

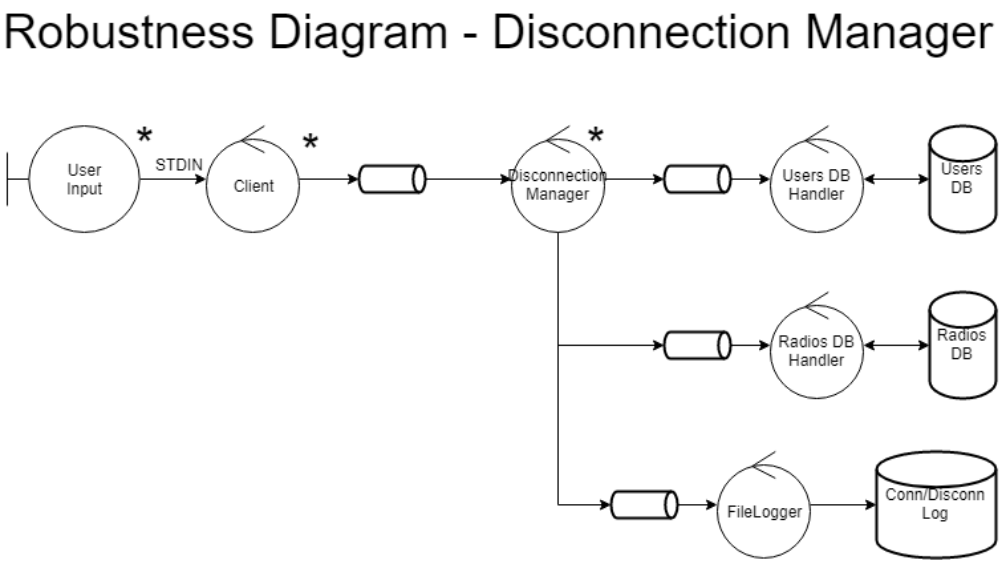

Figura 6: diagrama de robustez – rama de Disconnection Manager

Este camino es similar al anterior, excepto que no hay respuestas en ningún paso. El Client simplemente avisa de la desconexión por una cola única, y nuevamente algún Disconnection Manager procesa el pedido y avisa a ambas bases de datos y al log.

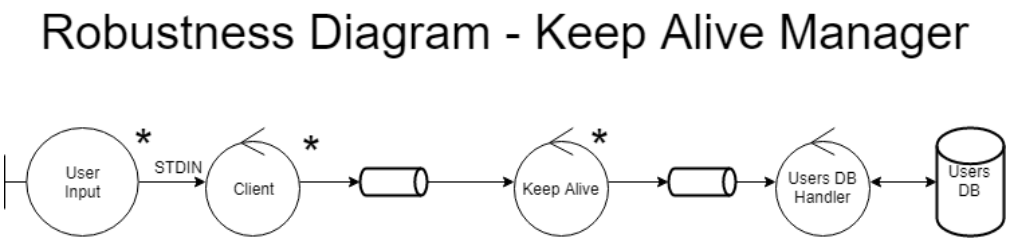

Figura 7: diagrama de robustez – rama de Keep Alive Manager

Este camino es similar pero más simple, ya que solo necesita interactuar con la base de datos de usuarios. Periódicamente mientras un Client está conectado a una radio envía mensajes para notificar que sigue ahí. Estos son pasados a la base de usuarios que actualiza este campo para esa conexión.

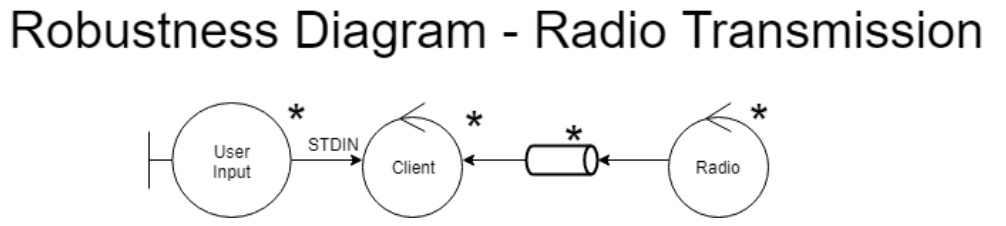

Figura 8: diagrama de robustez – rama de transmisión de radio

Aquí se puede ver la interacción entre el proceso Radio y el proceso Cliente. El proceso Radio debería ser entregado a todas las radios que deseen transmitir en el sistema, y configurado para transmitir su señal. La Radio transmite a cada Cliente conectado (a través de un Exchange de tipo fanout).

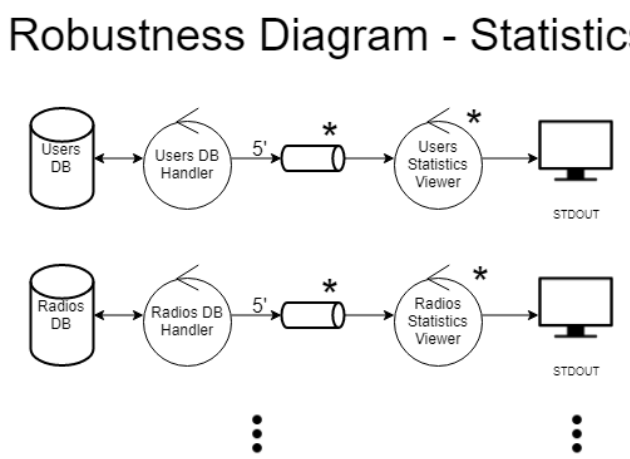

Figura 9: diagrama de robustez – estadísticas

Para la visualización de estadísticas, se optó por presentarlas periódicamente a través de colas (mediante en Exchange de tipo fanout). De este modo se puede conectar cualquier cantidad de Viewers del otro lado y recibir las estadísticas en todos los lugares que haga falta.

Todos los procesos descriptos se comunican a través de colas administradas por un broker de RabbitMQ. A continuación se presenta un diagrama de clases que intenta mostrar esta organización principal de procesos:

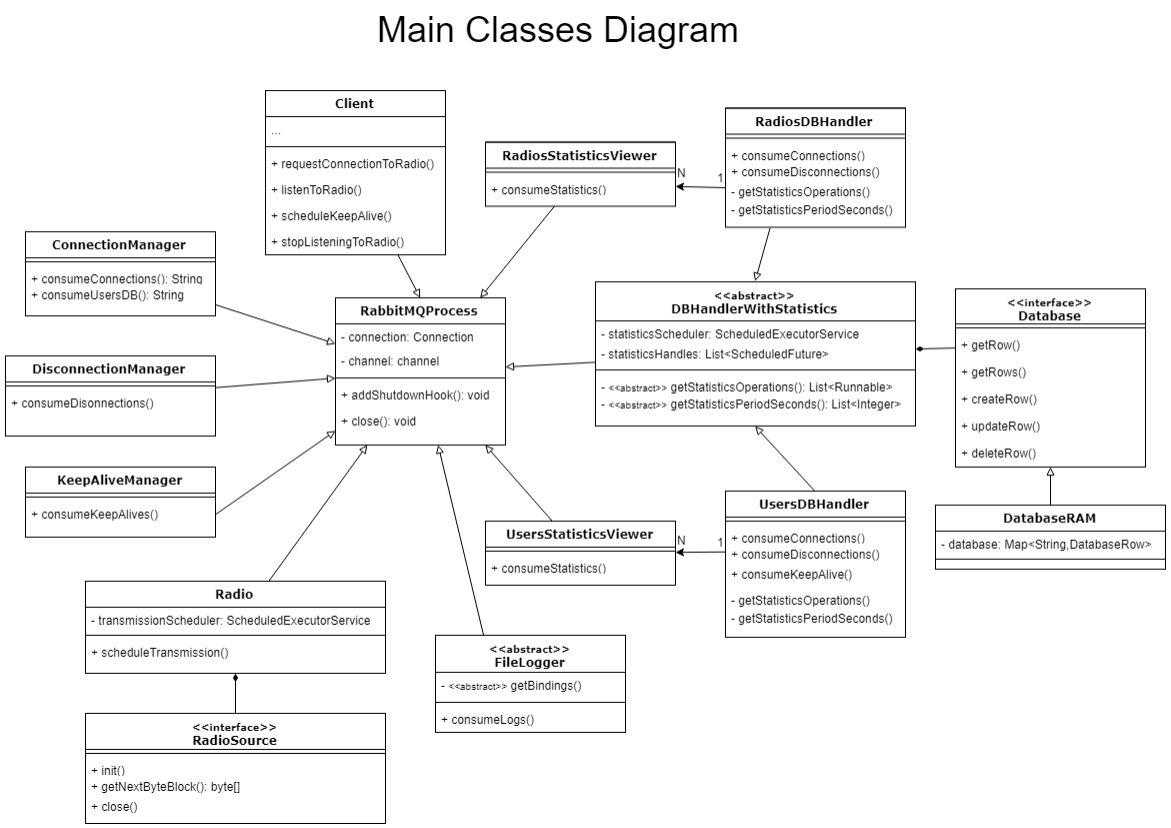

Figura 10: diagrama de clases principales (las que corren como procesos separados)

Cada uno de estos procesos es una parte distribuible del sistema y puede correr en un equipo separado del resto. Esto se intenta mostrar a partir del siguiente diagrama:

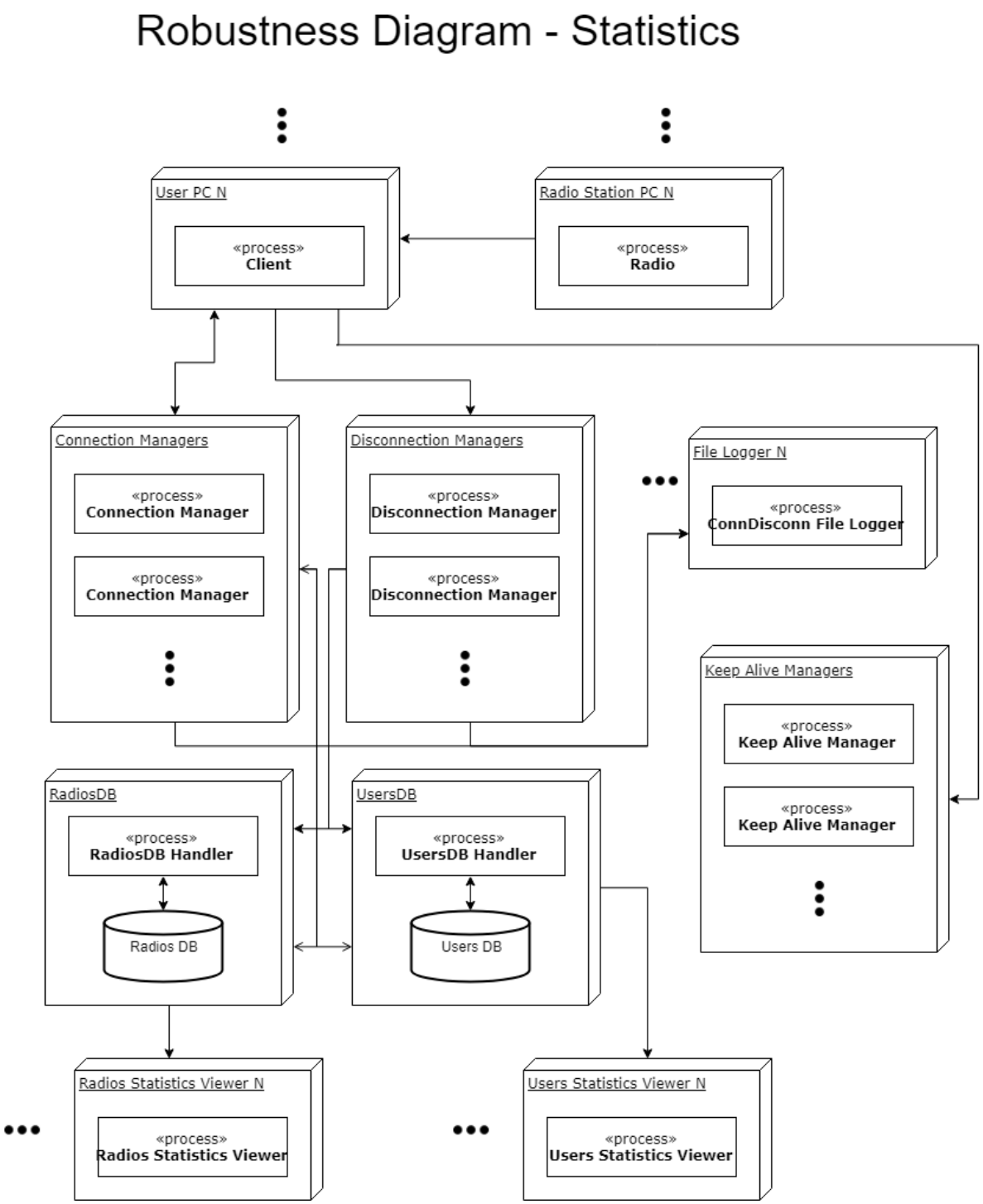

Figura 11: diagrama de despliegue