75.61 – Taller de Programacion III

Trabajo Práctico 2: Online Radio

1er cuatrimestre 2018

2da entrega – 26/04/18

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. El proyecto surge de una serie de requerimientos funcionales y no funcionales. A partir de estos y de la naturaleza del negocio se desarrollo una serie de casos de uso requeridos, y la estructura de la información que se necesita persistir. A partir de esta información se organizaron las entidades de procesamiento para proveer las funcionalidades requeridas, tratando de que fueran separables y escalables. A continuación se presenta una explicación del desarrollo realizado, acompañado de los diagramas necesarios para proveer distintas vistas 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 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.

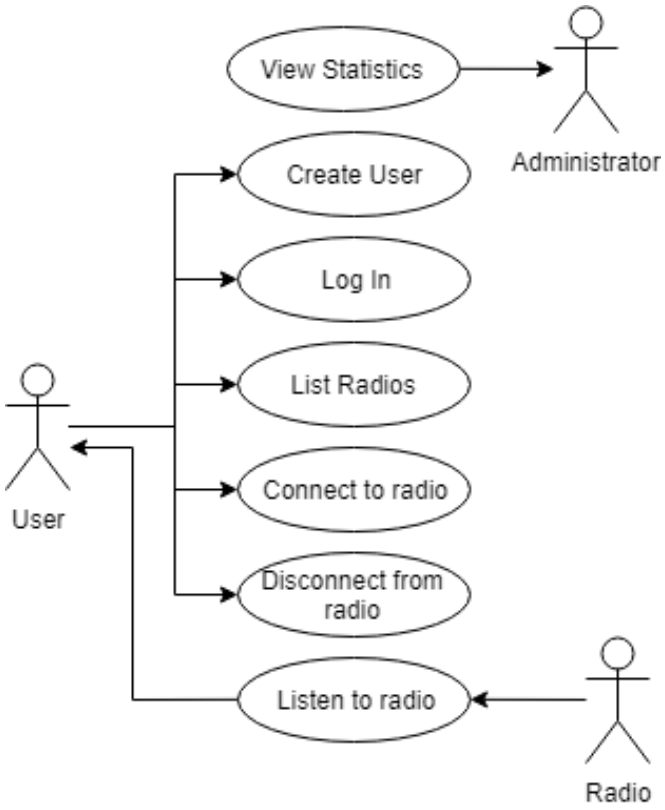

Figura 1: Diagrama de casos de uso

Las acciones que realizará el usuario serán: registrarse, iniciar sesión, pedir una lista de las radios disponibles, conectarse a una radio, escuchar la transmisión y desconectarse de una radio.

En la implementación realizada no se desarrollaron las funcionalidades de crear usuario ni listar radios.

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 profundiza en las siguientes secciones con énfasis en que el sistema sea lo más separable, distribuible y escalable posible.

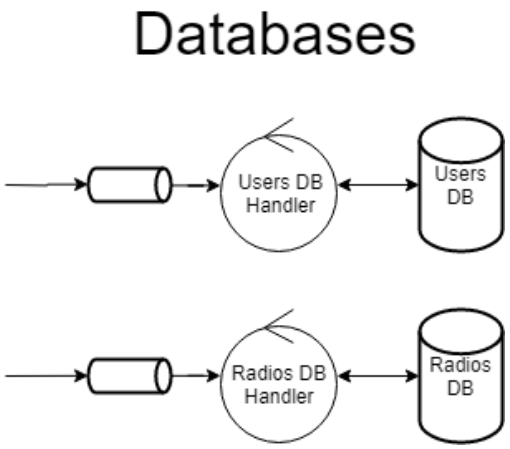

Figura 2: Bases de datos de radios y usuarios

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 y 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.

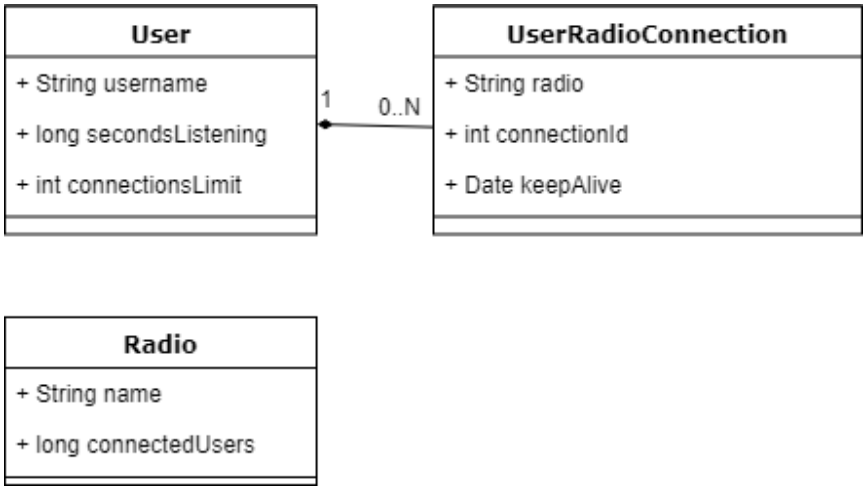

Figura 3: Registros de base de datos

Como se puede ver, se almacena una fila por usuario con los segundos que lleva escuchando y cuantas conexiones tiene permitidas. 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”.

Para las radios se almacena el nombre y la cantidad de conexiones activas.

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.

En la implementación realizada 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, como la sintonización en las radios tradicionales.

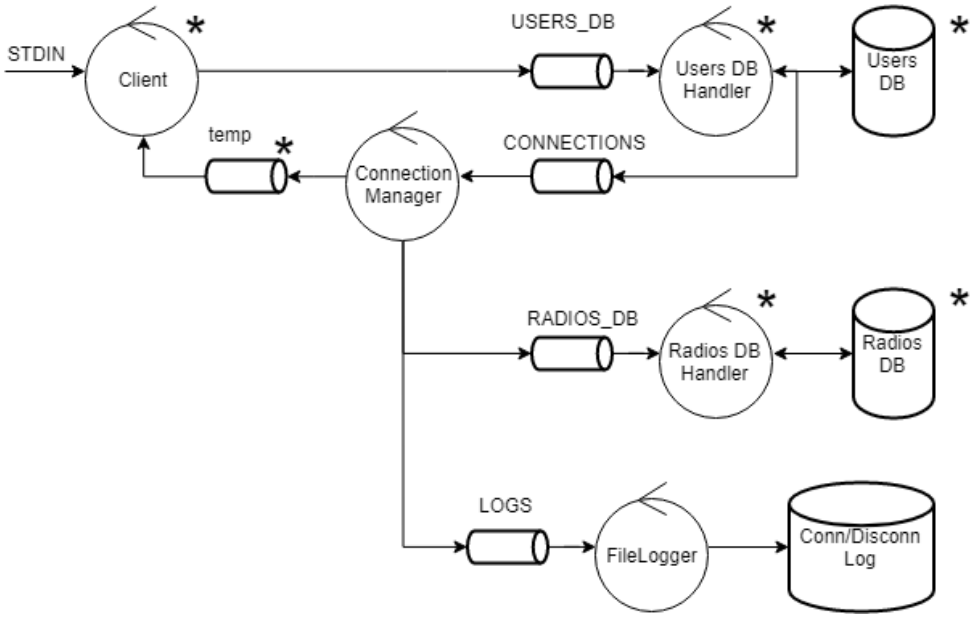

Figura 4: Diagrama de robustez – rama de Connection Manager

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. Corre en la computadora del usuario.

El proceso Client inicia la conexión a la radio encolando el pedido en un topic exchange, y se queda esperando en una cola temporal el resultado del pedido. El routing key del pedido es la primera letra del nombre de usuario.

De este exchange consumen mensajes varios UsersDBHandler cada uno ocupándose de un subset del espacio de routing keys posible. El manejador limpia las conexiones que hayan caducado de ese usuario y luego chequea que haya lugar para una nueva conexión. El resultado se responde al Connection Manager a través de una cola nombrada.

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 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.

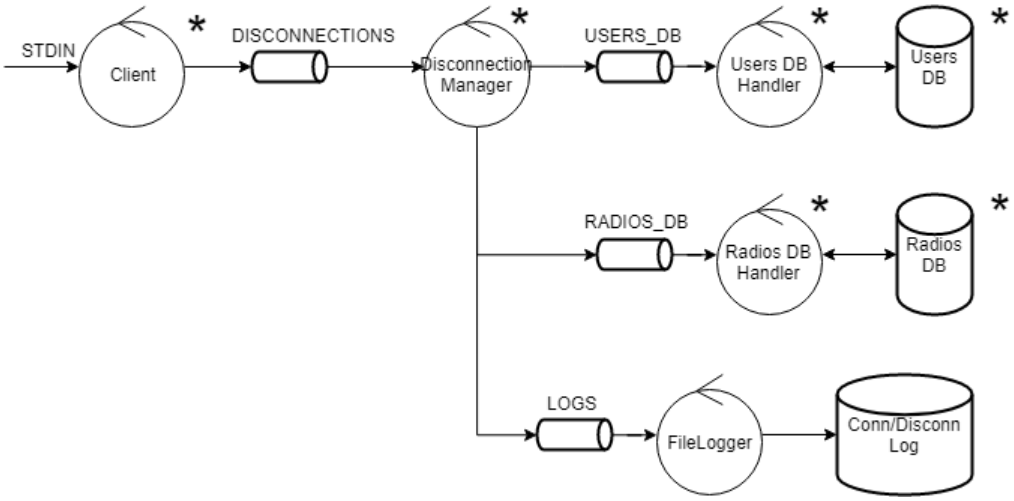

Figura 5: Diagrama de robustez – rama de Disconnection Manager

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.

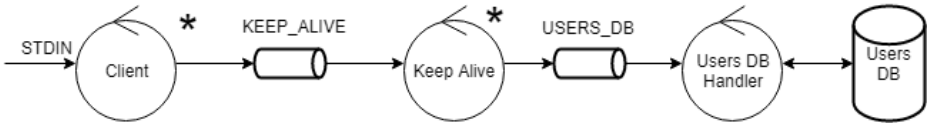

Figura 6: diagrama de robustez – rama de Keep Alive Manager

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

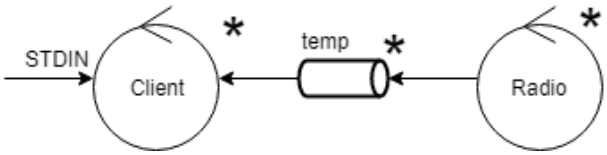

Figura 7: diagrama de robustez – rama de transmisión de radio

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.

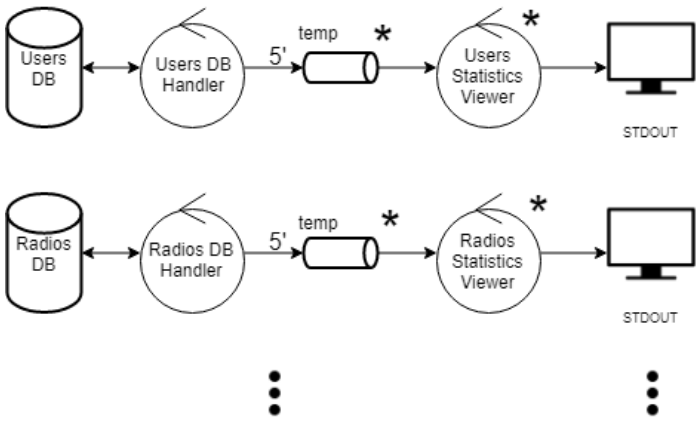

Figura 8: Diagrama de robustez – estadísticas

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:

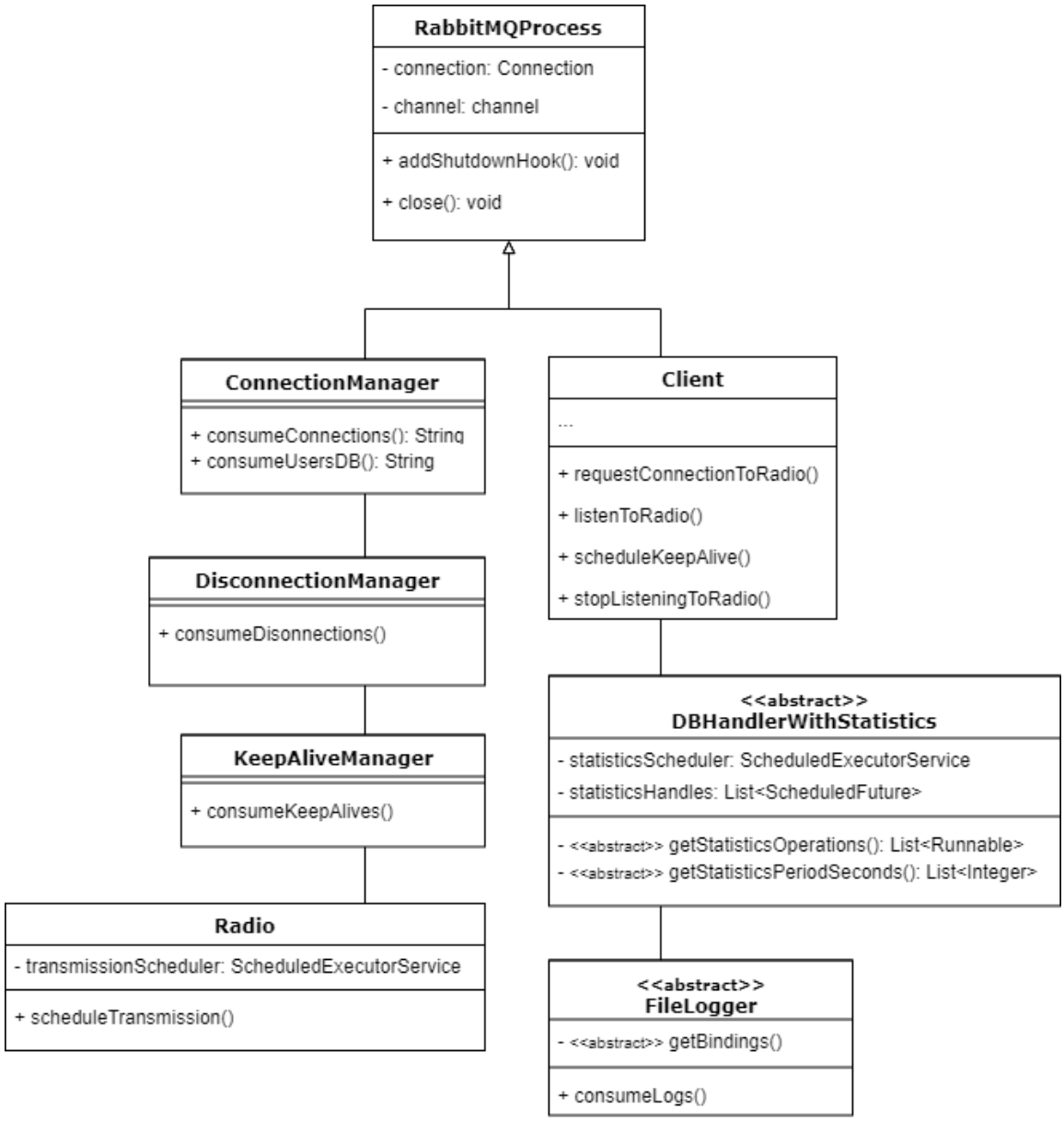

Figura 9: Diagrama de clases principales que corren como procesos separados

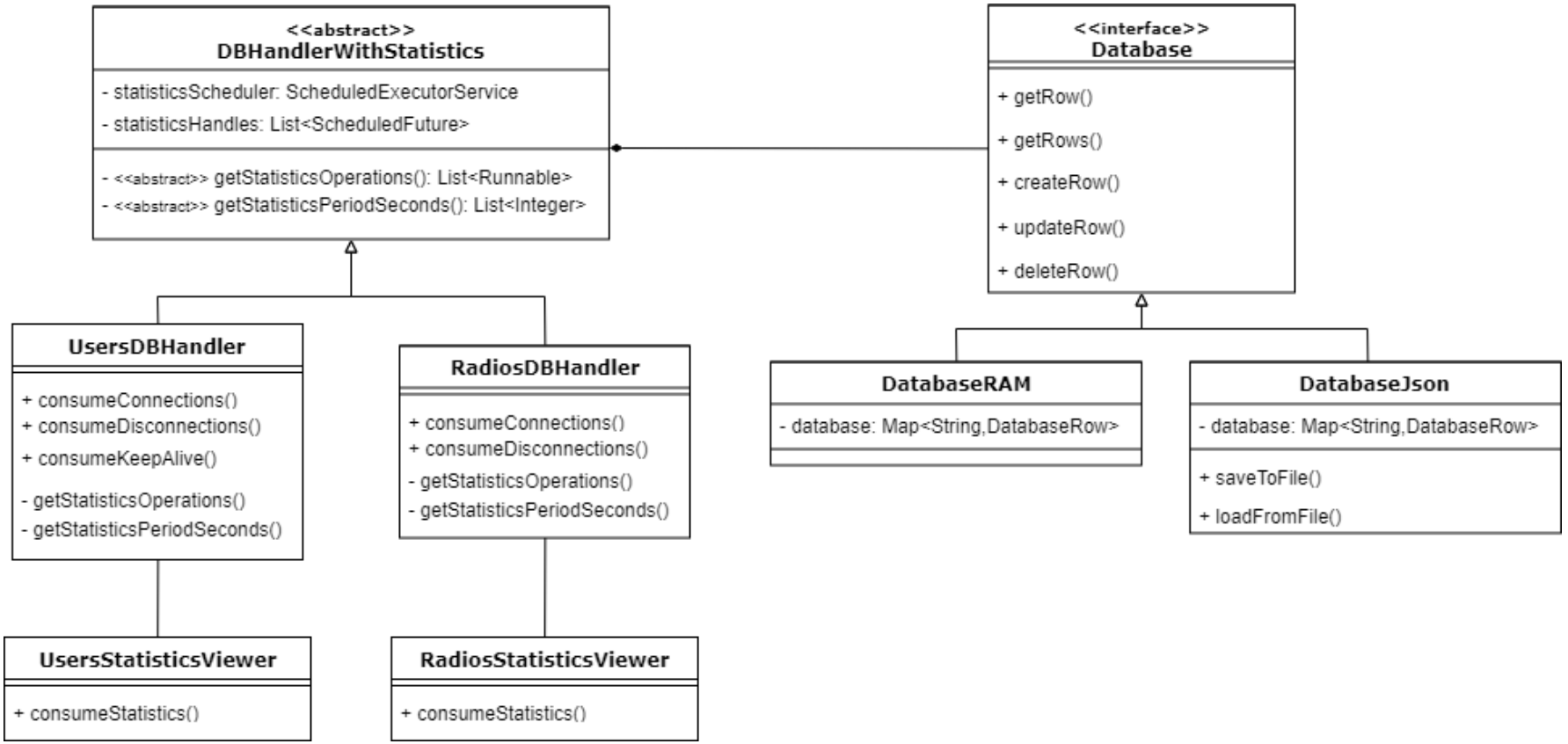

Figura 10: Diagrama de clases involucradas en persistencia de datos

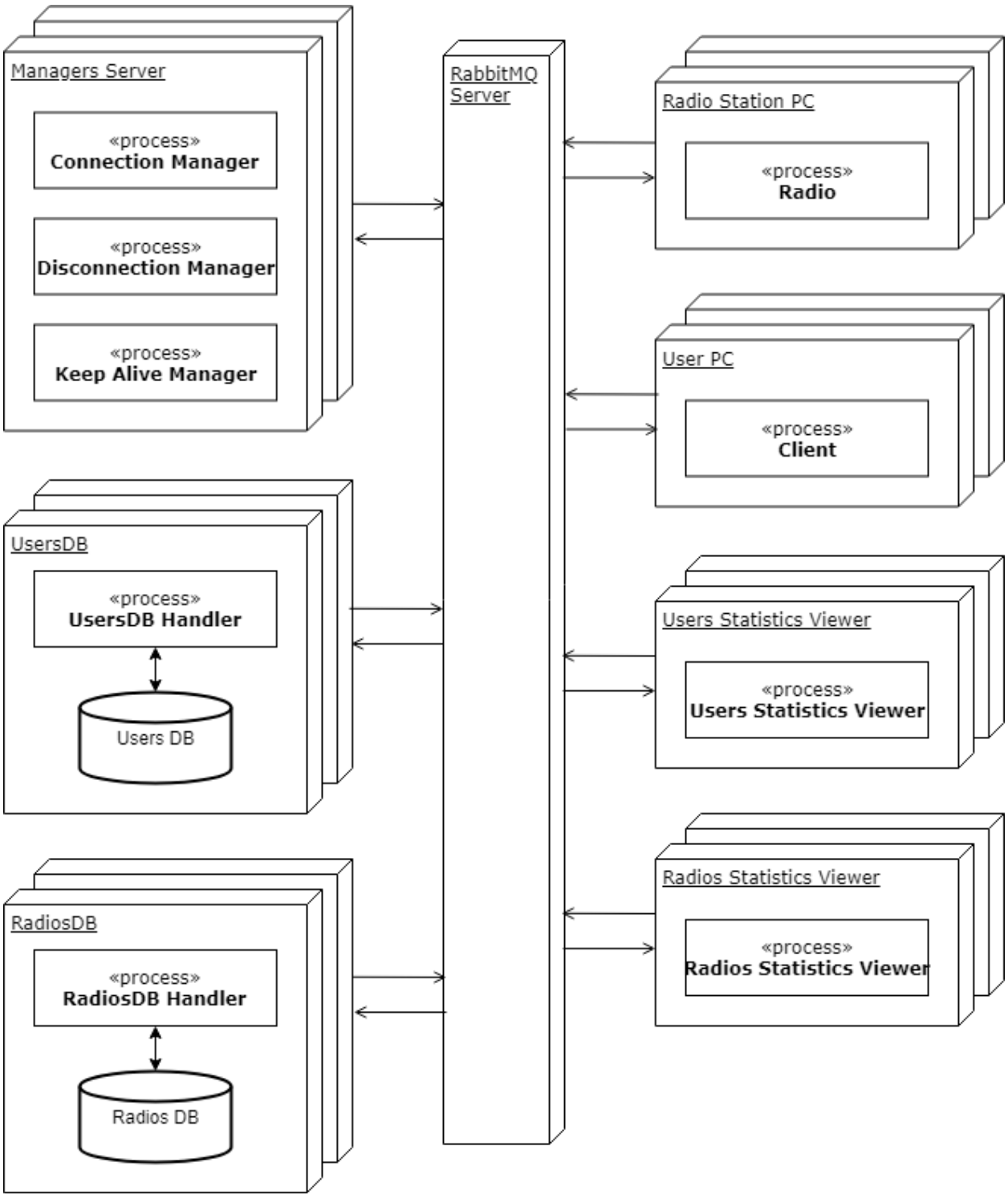

Figura 11: Diagrama de despliegue