SINCRONIZACIÓN DE DATOS DE USUARIO EN REDES DE PRÓXIMA GENERACIÓN

EIVAR EDER ARMERO LUNA

DIEGO FERNANDO RODRIGUEZ CHAMORRO

ANEXO A

INSTALACIÓN Y CONFIGURACIÓN DE SEQUOIA

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Servicios Avanzados de Telecomunicaciones
Popayán, Septiembre de 2008

ANEXO A

INSTALACIÓN Y CONFIGURACIÓN DE SEQOIA

TABLA DE CONTENIDO

LISTA DE FIGURAS

1. INTRODUCCIÓN

En este documento se describe el proceso de instalación de SEQUOIA en su versión 2.10.9 así como la estructura de directorios y los archivos de configuración. Esto último se presenta de forma genérica, ya que varía de acuerdo con las necesidades que se tengan.

Para realizar este documento se utilizó como base la documentación propia de esta herramienta, disponible en el sitio web de Continuent en la siguiente URL: http://sequoia.continuent.org/Manuals .

2. GENERALIDADES DE SEQUOIA

SEQUOIA es una solución *middleware open source* de la empresa Continuent capaz de manejar *clusters*, además de ofrecer mecanismos de balance de carga y de recuperación de fallos. Entre sus principales ventajas se encuentra que para su aplicación no es necesario hacer cambios sustanciales en las aplicaciones existentes, ya que únicamente basta con cambiar la URL de conexión hacia la base de datos. Su implementación ha sido realizada 100% en lenguaje Java, lo cual garantiza su portabilidad a cualquier sistema que posea un JRE superior al 1.4 así como la interoperabilidad con cualquier base de datos que provea un controlador

La Figura 1 muestra la arquitectura general de un sistema con bases de datos replicadas mediante SEQUOIA. El *core* del sistema está compuesto por controladores que implementan la tecnología RAIDb, los cuales se encuentran replicados para de esta forma garantizar una alta disponibilidad y una mejor escalabilidad. Estos mantienen la sincronización del *cluster* haciendo uso de primitivas de *group communication*, a través de HEDERA, un *wrapper* que permite ser configurado para trabajar con diferentes implementaciones de *group communication* tales como JGroups, Appia y Spread.

En los controladores se ubica una base de datos virtual (VDB - *Virtual Database*), a la cual los clientes se conectan, a su vez esta base de datos virtual ejecuta todas las transacciones recibidas por parte de los clientes en los servidores de bases de datos llamados *backends.* Esta conexión se realiza a través del controlador JDBC adecuado para cada tipo de motor. En la configuración de la VDB se especifican el tipo de replicación RAIDb a utilizar, así como el mecanismo de balance de carga y la implementación de *group comunication.*

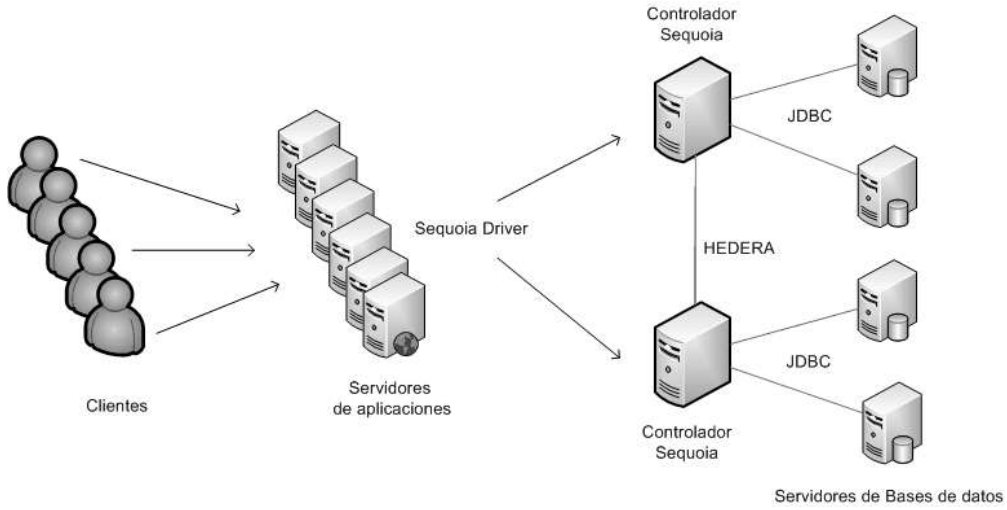

Figura 1. *Cluster* de bases de datos replicadas mediante SEQUOIA.

3. REQUERIMIENTOS

3.1 Hardware

- Un PC con buenas características de desempeño.

Como ejemplo a continuación se muestran las características del equipo utilizado para el desarrollo de las pruebas del proyecto.

- Procesador Intel Core 2 Duo 2.4 GHz
- 2 GB de memoria RAM
- Disco Duro de 160 GB
- Tarjeta de red Ethernet a 100 Mbps

3.2 Acceso de Red

Para un correcto funcionamiento, SEQUOIA debe contar con una red que soporte TCP/IP, esto para la comunicación entre el controlador y los diferentes *backends*, en entornos de producción se recomienda que esta conexión sea de tipo *full duplex* a 1Gbps, pero para propósitos académicos y de pruebas una interfaz a 100Mbps es suficiente.

Además de lo anterior se recomienda hacer uso direcciones IP estáticas evitando el uso de DHCP, esto debido a los problemas que se pueden generar para trabajar con ciertas características del controlador como lo es JGroups.

3.3 Software

- Cualquier sistema operativo que tenga soporte para Java.
- Controladores JDBC para cada tipo de base de datos que se vaya a utilizar como *backend*.
- Java Development Kit, JDK 1.4.2 o superior.

- Software de sincronización para el reloj del sistema, por ejemplo NTP (*Network Time Protocol*) en sistemas basados en Unix.

Adicional a lo presentado anteriormente cabe resaltar que el equipo donde va a instalar el controlador sequoia debe tener definida la variable de entorno JAVA_HOME apuntando al directorio de instalación de Java.

4. DESCARGA E INSTALACIÓN

Inicialmente se debe realizar la descarga del instalador de SEQUOIA desde su web oficial ingresando al siguiente enlace: https://forge.continuent.org/frs/?group_id=6. Aquí se pueden encontrar 3 versiones de este que son:

sequoia-2.10.9-bin-installer.jar – Asistente de instalación en modo gráfico
sequoia-2.10.9-bin.tar.gz – instalador en modo texto para plataformas Unix
sequoia-2.10.9-bin.zip - instalador en modo texto para plataformas Windows

A continuación se presentan los pasos para realizar la instalación haciendo uso del asistente en modo grafico que es la forma más recomendable de hacerlo, ya que este automáticamente creará la variable de entorno SEQUOIA_HOME de acuerdo a la configuración del sistema.

Inicialmente se debe lanzar el instalador, para ello desde una consola de línea de comandos se ejecuta la siguiente línea:

Java –jar sequoia-2.10.9-bin-installer.jar

Con esto se abrirá una pantalla como la presentada en la Figura 2, donde se muestra información sobre el fabricante, en esta pantalla se debe hacer clic en el botón *next* para proceder con la instalación.

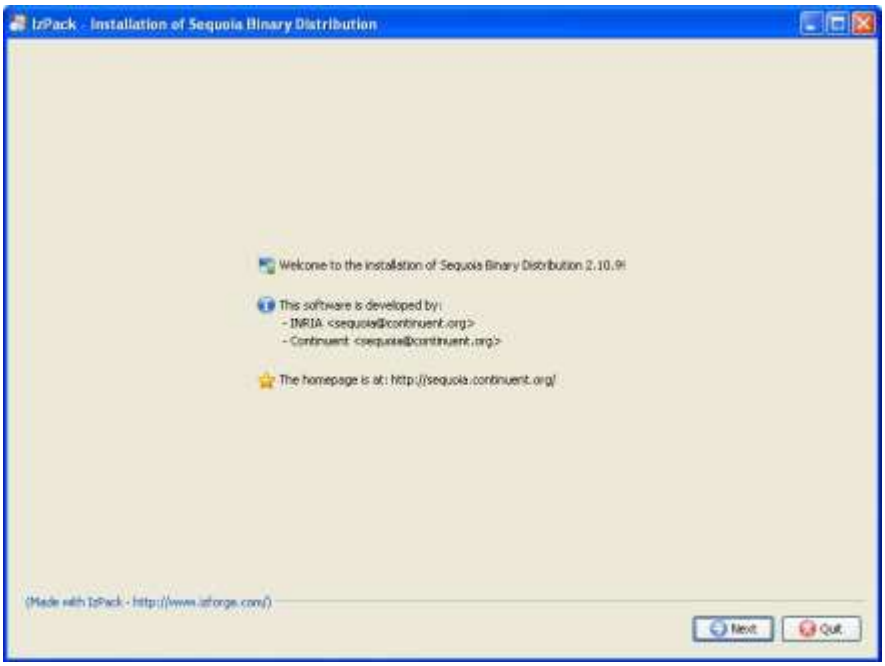

Figura 2. Ventana inicial del instalador de SEQUOIA.

La siguiente pantalla muestra los términos de la licencia de uso de la herramienta, aquí se debe seleccionar la opción: *I Accept the terms of this license agreement*, y posteriormente hacer clic en el botón *next*. Ver Figura 3.

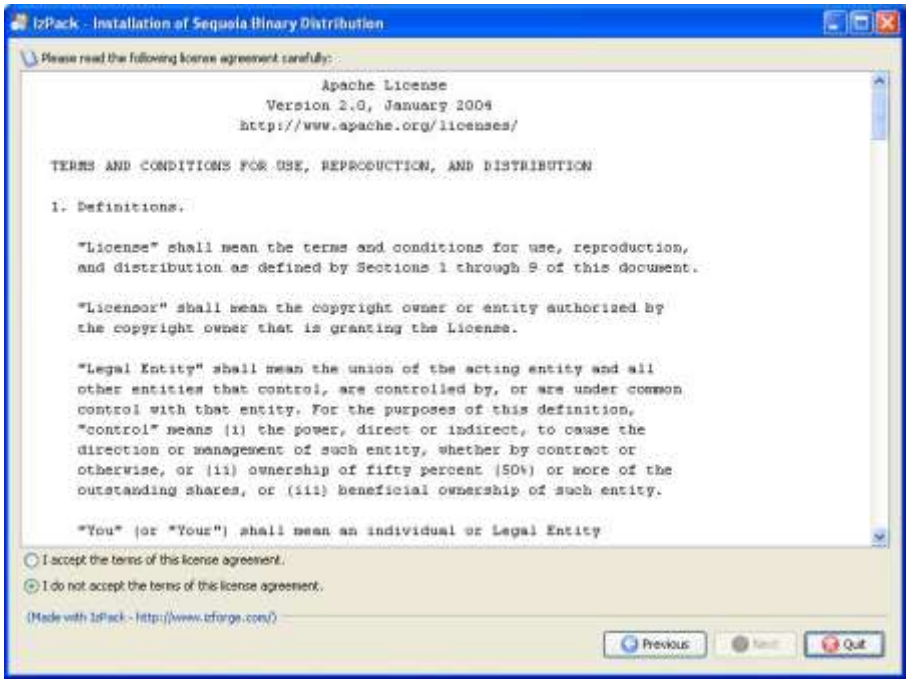

Figura 3. Licencia de uso de sequoia.

A continuación se despliega una nueva pantalla donde se debe introducir la ruta donde se quiere instalar el controlador SEQUOIA, para ello se debe hacer clic en el botón *browse* y seleccionar el directorio donde se desea instalar, tal como se muestra en la Figura 4. Una vez realizada la elección se debe continuar haciendo clic en el botón *next*.

Seguido esto, aparece una pantalla donde se pregunta qué componentes se desean instalar. Para una instalación estándar es recomendable seleccionar todas las opciones, seguido esto se hace clic en el botón *next* para iniciar la instalación como se ve en la Figura 5.

Con lo anterior aparecerá una pantalla con 2 barras de progreso que indican gráficamente el estado de la instalación, ver Figura 6.

Una vez que los indicadores muestren que el proceso de instalación ha terminado se debe hacer *click* en el botón *next* con lo que se presentará una pantalla como la mostrada en la Figura 7, que indica que la instalación se realizó satisfactoriamente, así pues únicamente basta con hacer clic en el botón *Done* y con esto se tendrá completamente instalado el controlador SEQUIOA.

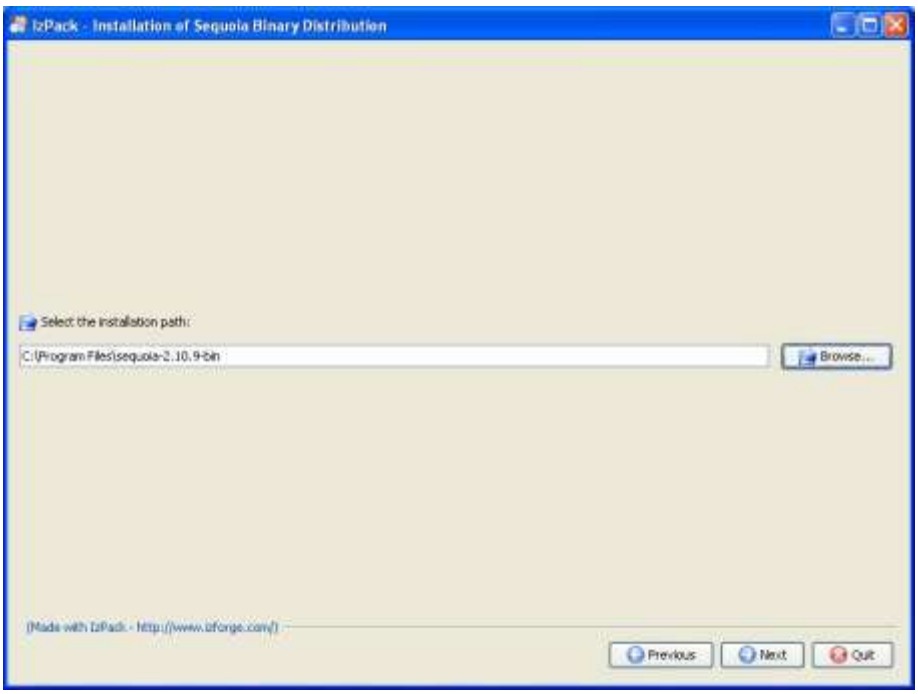

Figura 4. Selección del directorio destino de la instalación.

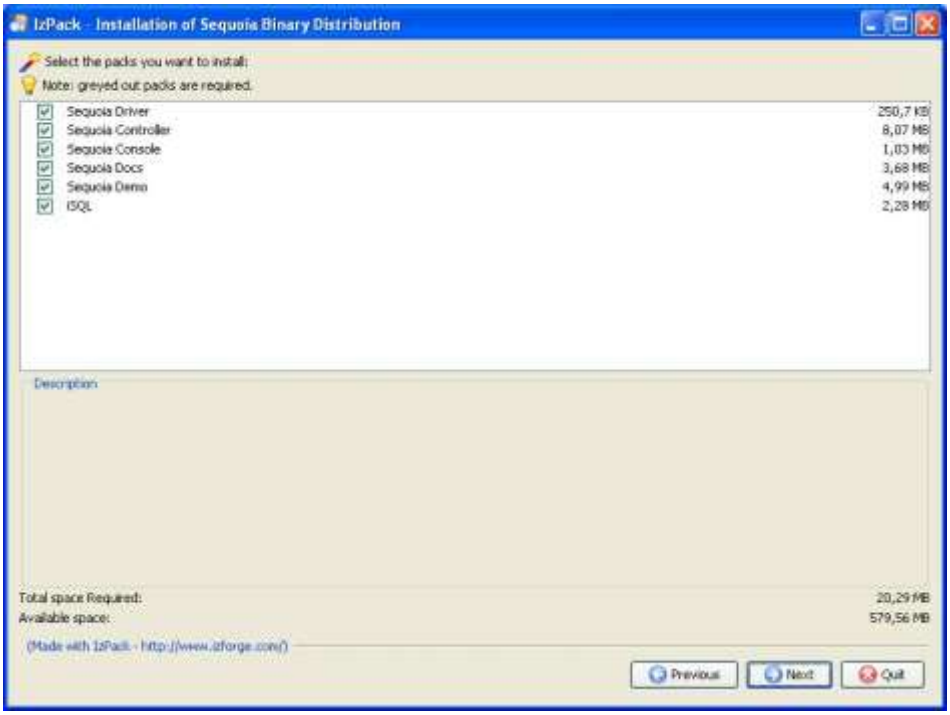

Figura 5. Selección de componentes a instalar.

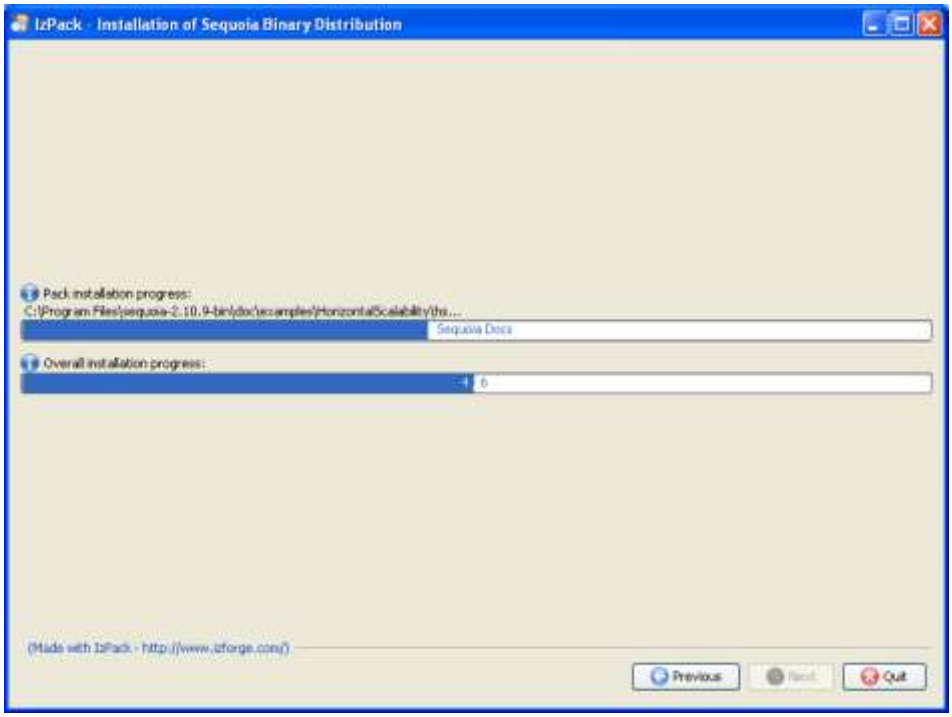

Figura 6. Progreso de la instalación.

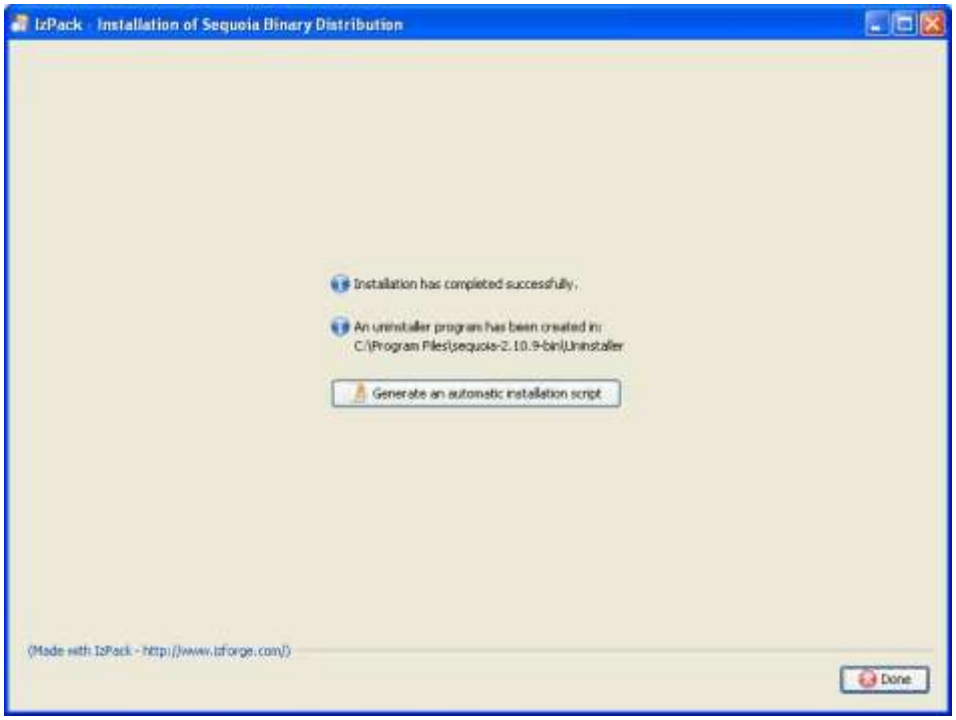

Figura 7. Confirmación de la instalación satisfactoria del controlador.

5. CONFIGURACIÓN

Para trabajar correctamente el consolador SEQUIOA debe contar con los siguientes archivos de configuración

- Un archivo de configuración por cada base de datos virtual
- Un archivo de configuración del controlador

Todos estos son documentos XML que deben ubicarse en directorios específicos.

5.1 Archivo de configuración del controlador

Cada controlador que se desee usar debe tener su propio archivo de configuración llamado controller.xml, este es usado durante el arranque del controlador. Las propiedades definidas en este serán aplicadas a todas las bases de datos virtuales alojadas por el controlador.

El archivo controler.xml, debe estar ubicado en la sub carpeta /config/controller de la instalación de SEQUIOA. Normalmente en esta carpeta se encuentra un archivo de configuración por defecto que puede ser usado como plantilla para generar una configuración personalizada de acuerdo a las necesidades que se posea en cada caso.

Un archivo de configuración de un controlador SEQUIOA luce más o menos como el siguiente:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE SEQUOIA-CONTROLLER PUBLIC "-//Continuent//DTD SEQUOIACONTROLLER
@VERSION@//EN" "http://sequoia.continuent.org/dtds/sequoiacontroller-@
VERSION@.dtd">
<SEQUOIA-CONTROLLER>
        <Controller ipAddress="127.0.0.1" port="25322">
                <JmxSettings>
                        <RmiJmxAdaptor port="1090"/>
                </JmxSettings>
<VirtualDatabase configFile="hsqldb-raidb1-distribution-1.xml"
virtualDatabaseName="myDB" autoEnableBackends="true"/>
</Controller>
</SEQUOIA-CONTROLLER>
```

Donde cada elemento introducido aquí cumple una función específica como se verá a continuación.

En el elemento Controller, el atributo ipAddress hace referencia a la interfaz de red por donde el controlador escuchará las peticiones, en caso que este se omita la IP por defecto será 0.0.0.0, es decir todas las interfaces de red disponibles.

El atributo *port*, se refiere al puerto que los clientes utilizarán para conectarse al controlador a través del *driver* de SEQUOIA, si se elige el puerto 0, este será asignado automáticamente por el sistema.

El elemento JmxSettings que a su vez contiene a RmiJmxAdapter, permite que el controlador sea administrado de forma remota usando la consola de línea de comandos de SEQUOIA, la cual es un cliente JMX (*Java Management eXtensions*) basado en el estándar RMI (*Java Remote Method Invocation*). La dirección IP y el puerto por defecto para este elemento por defecto son 0.0.0.0:1090, esto dado a que no se definió ninguno de estos 2 atributos específicamente.

El elemento VirtualDatabase, es opcional y es usado para indicarle al controlador que debe cargar la base de datos virtual especificada por el archivo de configuración definido en el atributo configFile, durante su arranque. El nombre de la base de datos virtual está dado por el parámetro virtualDatabaseName y debe ser igual al utilizado en el archivo de configuración de la base de datos virtual, adicionalmente, el parámetro autoEnableBackends le dice al controlador que una vez cargada la base de datos virtual habilite automáticamente todos los *backends* que esta tiene asociados.

Con esto ya se tiene un controlador listo para ser usado, ahora hace falta definir el archivo de configuración de la o las bases de datos virtuales que alojará el controlador.

5.2 Archivo de configuración de la base de datos virtual

Al igual que el caso del controlador, la base de datos virtual necesita tener su archivo de configuración en una ruta específica, para este caso es en la sub carpeta /config/virtualdabase de la instalación de SEQUIOA.

La configuración de la base de datos virtual es más compleja que la realizada para el controlador, debido a la gran cantidad de parámetros involucrados, así pues la descripción de este archivo se realiza en pequeños bloques para facilitar su comprensión.

El primer y principal elemento que se debe definir es SEQUIOA y dentro de este el elemento VirtualDatabase y como su principal atributo name, el cual define el nombre de la base de datos virtual que estará alojada en el controlador quedando de esta forma:

```
<SEQUOIA>
        <VirtualDatabase name="myDB">
        <Distribution>
        </Distribution>
        </VirtualDatabase>
</SEQUOIA>
```

El elemento Distribution se define para que el controlador pueda compartir la base de datos virtual con otros controladores haciendo uso de *group communication.* Si el controlador alberga múltiples bases de datos virtuales y se desea utilizar diferentes configuraciones de *group communication* en cada una de ellas es necesario añadir el atributo hederaPropertiesFile, de esta forma el elemento Distribution se definiría así:

```
<Distribution hederaPropertiesFile=”vdb2.properties”/>
```

El elemento Backup es utilizado para definir cual implementación de backuper será usada para realizar las copias de seguridad del la base de datos virtual.

```
<Backup>
        <Backuper backuperName=”Octopus” className=
                ”org.continuent.sequoia.controller.backup.backupers.
        OctopusBackuper” options=”zip=true”/>
</Backup>
```

En el ejemplo se está usando el backuper llamado Octopus, el cual hace uso de la implementación de Octopus dada por el atributo className. Adicionalmente, se pueden agregar opciones extra como la de habilitar la compresión de los *backups*.

En el elemento authenticationManager, se definen los usuarios de la base de datos virtual. En el sub elemento Admin, se redefine el nombre de usuario y el *password* del administrador de la Base de datos. Con estos datos se puede realizar la administración completa a través de la consola SEQUIOA. Por su parte los virtualUsers son todos aquellos usuarios de van a hacer uso de la base de datos desde las aplicaciones cliente.

```
<AuthenticationManager>
        <Admin>
                <User username="admin" password=""/>
        </Admin>
        <VirtualUsers>
```

```
            <VirtualLogin vLogin="user" vPassword="tester"/>
        </VirtualUsers>
</AuthenticationManager>
```

En el elemento DatabaseBackend, se definen cuales son las bases de datos reales que hacen parte de la base de datos virtual, se debe declarar un elemento DatabaseBackend por cada base de datos que pertenezca al *cluster*.

Los atributos que se deben especificar son el nombre del *backend*, el controlador JDBC que se usa para conectarse a esa base de datos, la URL de conexión hacia esa base de datos y una sentencia SQL para verificar si la conexión a la base de datos aun se mantiene después de que se produzca un fallo.

```
<DatabaseBackend>
        name=”localhost1”
        driver=”org.hsqldb.jdbcDriver”
        url=”jdbc:hsqldb:hsql://localhost:9001”
        connectionTestStatement=”call now()”>
        <ConnectionManager vLogin=”user” rLogin=”TEST” rPassword=””>
        </ConnectionManager>
</DatabaseBackend>
```

Dicha sentencia varía de acuerdo al tipo de motor de bases de datos con el que se esté trabajando y puede tomar alguno de los siguientes valores:

- Para MySQL usar select 1
- Para PostgreSQL usar select now().
- Para Apache Derby usar values 1.
- Para HSQL usar call now().
- Para SAP DB (MySQL MaxDB) usar select count(*) from versions.
- Para Oracle usar select * from dual.
- Para Firebird usar select 1 from rdb$types.
- Para InstantDB usar set date format "yyyy/mm/dd".
- Para Interbase usar select * from rdb$types.
- Para Microsoft SQL server 2000 usar select 1.

Si el usuario y el *password* de la base de datos es diferente al definido en el elemento VirtualUsers del AthenticationManager, se debe hacer un mapeo entre los datos reales y los datos virtuales, para esto se usa el elemento ConnectionManager y se definen los atributos vUser, rLogin y rPassword para especificar el usuario virtual, el usuario real y el *password* real.

```
<RequestManager >
<RequestScheduler>
  <RAIDb-1Scheduler level="passThrough" />
</RequestScheduler>
  <LoadBalancer transactionIsolation="databaseDefault">
        <RAIDb-1>
```

```
        <WaitForCompletion policy="first" enforceTableLocking="false"
        deadlockTimeoutInMs="30000" />
        <RAIDb-1-LeastPendingRequestsFirst />
        </RAIDb-1>
  </LoadBalancer>
```

En el elemento RequestManager se definen los mecanismos de balance de carga y de replicación deseados, teniendo que existen cuatro formas de realizar esta replicación:

- Sin replicación: como su nombre lo indica no se realiza ninguna replicación y solo se tiene un controlador y un *backend*.
- RAIDb – 0: provee un particionamiento completo de las tablas, es decir que las diferentes tablas que pertenecen a una base de datos se encuentran distribuidas en diferentes *backends*
- RAIDb – 1: provee una replicación completa de toda la base de datos en todos los *backends*.
- RAIDb – 2: provee una replicación completa en todos los *backends*, pero únicamente a nivel de tabla.

Los elementos LoadBalancer y RequestScheduler dependen del tipo de replicación que se elija.

Dentro de RequestManager también se define la estructura de la base de datos del RecoveryLog que va a ser usado para pode restaurar los *backends* cuando ocurra un fallo.

```
<RecoveryLog driver="com.mysql.jdbc.Driver"
      url="jdbc:mysql://thor.unicauca.edu.co:3306/log" login="log" password="loger"
      requestTimeout="60" recoveryBatchSize="10" idleConnectionTimeout="0">
    <RecoveryLogTable tableName="RECOVERY" logIdColumnType="BIGINT NOT NULL"
          vloginColumnType="TEXT NOT NULL" sqlColumnName="sql2"
          sqlColumnType="TEXT NOT NULL" sqlParamColumnType="TEXT NOT NULL"
          transactionIdColumnType="BIGINT NOT NULL"
          extraStatementDefinition=",PRIMARY KEY (log_id)" createTable="CREATE
          TABLE" autoConnTranColumnType="CHAR(1) NOT NULL"
          requestIdColumnType="BIGINT" execTimeColumnType="BIGINT"
          updateCountColumnType="INT" />

    <CheckpointTable tableName="CHECKPOINT"
          checkpointNameColumnType="VARCHAR(200) NOT NULL"
          createTable="CREATE TABLE" logIdColumnType="BIGINT"
          extraStatementDefinition=",PRIMARY KEY (name)" />

    <BackendTable tableName="BACKEND" databaseNameColumnType="TEXT NOT
          NULL" backendNameColumnType="TEXT NOT NULL"
          checkpointNameColumnType="TEXT NOT NULL" createTable="CREATE
          TABLE" backendStateColumnType="INTEGER" extraStatementDefinition="" />

    <DumpTable tableName="DUMP" dumpNameColumnType="TEXT NOT NULL"
          dumpDateColumnType="TIMESTAMP" dumpPathColumnType="TEXT NOT
```

```
                NULL" dumpFormatColumnType="TEXT NOT NULL"
                checkpointNameColumnType="TEXT NOT NULL"
                backendNameColumnType="TEXT NOT NULL" tablesColumnType="TEXT NOT
                NULL" createTable="CREATE TABLE" tablesColumnName="tables"
                extraStatementDefinition="" />
  </RecoveryLog>
</RequestManager>
```

Al igual que se hace en un *backend*, en este punto se debe proporcionar el nombre de usuario, el *password* , el *driver* y la URL donde está ubicada la base de datos que va a funcionar como RecoveryLog. Y posteriormente se debe dar la estructura de cada una de las tablas que lo componen.

- Tabla Recovery: guarda la información que viaja en las peticiones de las aplicaciones cliente.
- Tabla Chaeckpoint: Guarda información de los puntos de control lógicos que genera el controlador.
- Tabla Backends: Guarda información referente a los *backends* asociados a la base de datos virtual
- Tabla Dump: almacena información referente a los archivos de respaldo generados con el *backuper*.