UNIVERSISTAT OBERTA DE CATALUNYA

ESTUDIOS DE INGENIERIA INFORMÁTICA

PROYECTO FINAL DE CARRERA

APLICACIONES WEB PARA TRABAJO COLABORATIVO

‘DISEÑO Y PROTOTIPO DE UN SOFTWARE PARA LA REPLICACIÓN DE FICHEROS EN SISTEMAS DE TRABAJO EN GRUPO P2P’

Manual de Administración

Alumno: **Ángel Navarro Estepa**

Consultor: **Fatos Xhafa**

ÍNDICE

1 INTRODUCCIÓN

El siguiente documento detalla la forma de poder administrar la aplicación que permitirá compartir ficheros dentro de un entorno colaborativo.

La administración de la misma tendrá dos partes:

- Administración del SuperPeer
- Administración del Peer (Nodo)

Previo a todo esto, se hará una breve explicación del porque del uno sin el otro.

1.1 SISTEMA: SUPERPEER / PEER

El sistema está formado por dos elementos que el uno sin el otro se podría decir que no tienen sentido.

El SuperPeer es el subsistema encargado de orquestar todo el volumen de información y asegurar la coherencia y existencia de la documentación.

Los Nodos (peer) son los encargados de mostrar una interficie gráfica mediante la cual el usuario podrá gestionar toda la documentación xml que se utilizará en el marco colaborativo creado.

Aunque como se ha podido ver la *misión* de los dos subsistemas es diferente, comparten arquitectura, por lo que se definirá qué requisitos son necesarios para poder gestionar cada uno de los elementos.

A parte, será necesario conocer cómo realizar la administración de un subsistema y del otro.

2 ELEMENTOS COMUNES

2.1 REQUISITOS DEL SISTEMA

La arquitectura utilizar en cualquier de los dos tipos de nodo (peer / superpeer) es la misma, por lo que será necesario disponer de la siguiente arquitectura para poder trabajar con la aplicación.

Sistema operativo

Indiferente; aunque la aplicación ha sido validada en Windows 7.

Java

Toda la aplicación está desarrollada con el lenguaje de programación *java*. Por ello es necesario disponer de una máquina virtual java (o superior):

Java versión "1.6.0_03"

Java SE Runtime Environment (build 1.6.0_03-b05)

Para el correcto funcionamiento de la solución, se tiene que dar de alta java en el path del sistema.

Librerías Gráficas

Con anterioridad se ha indicado que el sistema operativo es indiferente, aunque es imprescindible que es sistema operativo utilizado para trabajar con los nodos tipo *peer* tenga cargadas las librerías gráficas AWT.

Al tratarse de una aplicación con una interficie de usuario basada en las librerías Swing de Java es un requisito para poder trabajar con ventanas.

No pasa lo mismo con el sistema de la máquina donde corra el nodo *superpeer*. Este tipo de nodo no tiene interficie gráfica por lo que no utilizará ventanas.

Conectividad con el exterior

Salvo que todo nuestro entorno colaborativo lo creemos en una red interna, será necesario que todas las máquinas donde se instalen los nodos (del tipo que sea) tengan conectividad a internet.

La base de esta aplicación es el compartir documentos xml desde nodos que pueden estar en cualquier parte del mundo.

Tener en cuenta que si utilizamos un sistema con cortafuegos o antivirus que se encargan de controlar y gestionar los puertos de la máquina en la que se esté trabajando, esto pueda impedir la comunicación con el nodo *superpeer* e incluso con el resto de nodos del sistema.

Para ello es aconsejable abrir los puertos (en caso de ser necesario) utilizados en la aplicación (consultar apartados *SuperPeer y Peer* para tener más información de cómo cambiar estos parámetros).

Espacio en disco

Se trata de una aplicación que gestiona documentos xml. El número de documentos no está limitado, por lo que será necesario disponer de espacio en disco.

Por otro lado, la aplicación genera una serie de ficheros de trazas donde poder consultar en todo momento que está sucediendo en la misma. Estos ficheros requieren de espacio en disco.

Aunque indicar un espacio específico es difícil, se establece que el espacio mínimo requerido es de:

30Mb

Disponibles en la unidad de disco donde se instale la aplicación.

En caso de cambiar el directorio de trabajo, disponer de una capacidad mínima de **15Mb**.

Resaltar que conforme vayan aumentando el volumen de documentos a gestionar este requisito mínimo de espacio se quedará corto.

2.2 SISTEMA DE TRAZAS

Cada uno de los dos subsistemas utiliza una gestión de trazas para poder hacer un seguimiento de la operativa de la aplicación.

El sistema de trazas utilizado trabaja con tres ficheros:

- derby.log: Indica la operativa que se realiza internamente en la base de datos
- pfc_prototipo_bbdd_trazas.log: Indica la operativa que realiza la aplicación con la base de datos.

- pfc_prototipo_trazas.log: Indica la operativa global de la aplicación.
- Pfc_prototipo_rendimiento_trazas.log: Indica logs de tiempo para poder saber aspectos de rendimiento de la aplicación

El primer fichero lo gestiona la propia base de datos.

El segundo, tercer y cuarto fichero se configuran mediante el fichero:

log4j.properties

En este fichero está definido el nivel de trazas y los ficheros destino.

2.2.1 CAMBIAR NIVEL DE LOG

Por defecto el nivel de log es el más bajo (el que más trazas deja): DEBUG.

Una vez la aplicación lleva varios días funcionando sin anomalías se puede subir el nivel de log a ERROR. De esta forma únicamente saldrán las trazas de error que pueda tener la aplicación y de esta forma se consigue disminuir el espacio en disco utilizado.

Para ello, cambiar del fichero de configuración los puntos resaltados en rojo:

```
#logger para BBDD

log4j.appender.BBDD=org.apache.log4j.FileAppender

log4j.appender.BBDD.Threshold=DEBUG

log4j.appender.BBDD.ImmediateFlush=true

log4j.appender.BBDD.file=pfc_prototipo_bbdd_trazas.log

log4j.appender.BBDD.layout=org.apache.log4j.PatternLayout

log4j.appender.BBDD.layout.ConversionPattern=%d %-5p %C.%M(%L) --> %m%n

log4j.appender.Default.append=false


#logger para DEFECTO

log4j.appender.DEFECTO=org.apache.log4j.FileAppender

log4j.appender.DEFECTO.Threshold=DEBUG

log4j.appender.DEFECTO.ImmediateFlush=true

log4j.appender.DEFECTO.file=pfc_prototipo_trazas.log

log4j.appender.DEFECTO.layout=org.apache.log4j.PatternLayout
```

log4j.appender.DEFECTO.layout.ConversionPattern=%d %-5p %C.%M(%L) --> %m%n

log4j.appender.Default.append=false

Poniendo ERROR.

En caso de querer poner un nivel de trazas intermedio, hacer lo mismo pero indicando INFO.

2.2.2 CAMBIAR RUTA FICHEROS DE TRAZAS

Ya se ha comentado que el sistema de trazas de la propia aplicación genera tres ficheros:

pfc_prototipo_bbdd_trazas.log

pfc_prototipo_trazas.log

pfc_prototipo_rendimiento_trazas.log

En el fichero de configuración se define el path de estos ficheros. Por defecto está indicado que lo deje en el mismo directorio de instalación. Si se quiere cambiar el path, indicar delante del nombre del fichero el path deseado (se resalta en rojo el sitio donde se indica path completo con nombre de los ficheros):

#logger para BBDD

log4j.appender.BBDD=org.apache.log4j.FileAppender

log4j.appender.BBDD.Threshold=DEBUG

log4j.appender.BBDD.ImmediateFlush=true

log4j.appender.BBDD.file=pfc_prototipo_bbdd_trazas.log

log4j.appender.BBDD.layout=org.apache.log4j.PatternLayout

log4j.appender.BBDD.layout.ConversionPattern=%d %-5p %C.%M(%L) --> %m%n

log4j.appender.Default.append=false

#logger para RENDIMIENTO

log4j.appender.RENDIMIENTO=org.apache.log4j.FileAppender

log4j.appender.RENDIMIENTO.Threshold=DEBUG

log4j.appender.RENDIMIENTO.ImmediateFlush=true

log4j.appender.RENDIMIENTO.file=pfc_prototipo_rendimiento_trazas.log

log4j.appender.RENDIMIENTO.layout=org.apache.log4j.PatternLayout

log4j.appender.RENDIMIENTO.layout.ConversionPattern=%d --> %m%n

```
log4j.appender.Default.append=false

#logger para DEFECTO

log4j.appender.DEFECTO=org.apache.log4j.FileAppender

log4j.appender.DEFECTO.Threshold=DEBUG

log4j.appender.DEFECTO.ImmediateFlush=true

log4j.appender.DEFECTO.file=pfc_prototipo_trazas.log

log4j.appender.DEFECTO.layout=org.apache.log4j.PatternLayout

log4j.appender.DEFECTO.layout.ConversionPattern=%d %-5p %C.%M(%L) --> %m%n

log4j.appender.Default.append=false
```

2.3 ESTRUCTURA DE DIRECTORIOS

La estructura que se crea para poder ejecutar la aplicación es la siguiente (con independencia del tipo de nodo):

- .cache: Subdirectorio creado por la plataforma jxta donde almacena toda la configuración necesaria para crear un entorno colaborativo entre nodos y *superpeer*. Al instalar las aplicaciones este directorio no existe. Lo crea el sistema la primera vez que se conecta a la red.
- documentos: path inicial donde se almacenarán todos los documentos. Path por defecto y final del *superpeer* en sus funciones de *data center*.
- pfc: Estructura de carpetas y ficheros con los binarios de la aplicación
- PFCGrupoBD: Estructura de carpetas y ficheros generados y que forman la base de datos de la aplicación. Al instalar las aplicaciones este directorio no existe. Lo crea el sistema la primera vez que se ejecuta la solución.

A parte, en el directorio raíz de la aplicación residirán los siguientes ficheros:

- derby.log: Trazas propias de la base de datos
- pfc_prototipo_bbdd_trazas.log: Trazas de la aplicación donde se refleja la intercomunicación con la base de datos.
- Pfc_prototipo_trazas.log: Trazas de la propia aplicación
- peer.bat / superpeer.bat: Fichero de ejecución de la aplicación. El primero (peer.bat) estará en las instalaciones de los nodos

tipo *peer*. El segundo (superpeer.bat) estará en la instalación del nodo tipo *superpeer*.

- log4j.properties: Fichero configuración del propio sistema de trazas de la aplicación.
- jaxen.jar: Librería *jaxen* utilizada por la aplicación
- jdom-1.1.2.jar: Librería *jdom* utilizada por la aplicación
- derby.jar: Librería *derby* (base de datos) utilizada por la aplicación
- log4j-1.2.16.jar: Librería *log4j* utilizada por la aplicación
- jxta.jar: Librería *jxta* utilizada por la aplicación

3 NODO SUPERPEER

3.1 INTRODUCCIÓN

Aunque en un sistema p2p se puede disponer de 'n' nodos tipo *superpeer*, en nuestro despliegue nos basamos en disponer únicamente de un nodo de este tipo.

Como punto de partida indicar que para que el sistema pueda funcionar de forma óptima el nodo *superpeer* siempre tiene que estar levantado y tiene que ser el primer elemento a arrancar en la puesta en marcha del entorno colaborativo a crear con el arranque de los nodos tipo *peer*.

También tener en cuenta que en este desarrollo, el nodo *superpeer* hace las funciones de *Data Center*.

3.2 INSTALAR SUPERPEER

Siempre y cuando el sistema donde se va a poner en marcha el nodo *superpeer* disponga de los requisitos definidos en el apartado *2.1 Requisitos del sistema*, no se precisa de realizar instalación previa de cualquier otro elemento.

Por ello, la instalación del nodo *superpeer* será bastante sencilla y únicamente se tendrá que descomprimir el fichero *superpeer.zip*.

Una vez realizada esto, se creará la siguiente estructura de directorios (para más información sobre el significado de cada directorio consultar el apartado *2.3 Estructura de directorios*):

- pfc

y en el directorio raíz existirán los siguientes ficheros:

- superpeer.bat
- log4j.properties
- jaxen.jar
- derby.jar
- log4j-1.2.16.jar
- jxta.jar

Una vez hecho y comprobado todo esto, el nodo *superpeer* estará listo para poder ponerse en marcha.

3.3 ARRANCAR SUPERPEER

Ejecutar el siguiente fichero:

Superpeer.bat

De esta forma se iniciar el nodo *superpeer* con sus valores por defecto.

Hay que tener en cuenta que al iniciar el nodo *superpeer* por primera vez se crea la base de datos, se crea configuración del sistema y se crea configuración del entorno colaborativo.

Una vez arrancado, si se observa la estructura de directorios se puede observar que se han creado tres nuevos directorios:

- .cache
- documentos
- PFCGrupoBD

A parte, se han generado los siguientes ficheros:

- derby.log
- pfc_prototipo_trazas.log
- pfc_prototipo_bbdd_trazas.log
- pfc_prototipo_rendimiento_trazas.log

Para parar el nodo *superpeer* basta con hacer un Control+C del proceso lanzado.

3.4 CONFIGURACIÓN

Cuando se arranca el nodo *superpeer* se crea la configuración por defecto. Esta configuración contempla los siguientes puntos:

Parámetro	Valor
CONF_NombrePeer	PFC_SuperPeer
CONF_IdPeer	Crear identificador único en base al nombre del peer y al nombre del grupo
CONF_NombreGrupo	PFC_GrupoTrabajo
CONF_IdPipeMulticast	Crear identificador único en base al nombre del PipeMulticast(ComunicacionGrupo) y al nombre del grupo

CONF_IdPipeUnicast	Crear identificador único en base al nombre del PipeUnicast(ComunicacionPeer) y al nombre del grupo
CONF_SuperPeerPuerto	9717
CONF_Directorio	Será el path de ejecución de la aplicación +"\documentos\"

El único parámetro que se puede modificar es el puerto.

3.4.1 CAMBIO DEL PUERTO

Por defecto el nodo *superpeer* trabaja con el puerto 9717; pero se puede hacer que el puerto utilizado sea otro. Para ello, modificar el fichero de arranque del nodo para añadir (lo indicado en rojo):

```
java -cp jxta.jar;derby.jar;log4j-1.2.16.jar;jdom-1.1.2.jar;jaxen.jar;.
pfc.prototipo.jxta.superpeer.PFC_SuperPeer 9767
```

3.4.2 DIRECTORIO DE TRABAJO DOCUMENTOS

El nodo *superpeer* también hace las funciones de *data center*; esto implica que tiene que almacenarse físicamente todos los documentos que se van generando en el entorno colaborativo.

Los documentos xml se guardan en el directorio indicado en el parámetro de configuración *CONF_Directorio*.

Para el nodo *superpeer* este parámetro es fijo y tal como se ha explicado, se crea en la raíz donde está instalado el nodo el directorio *documentos*.

4 NODO PEER

4.1 INTRODUCCIÓN

Los nodos tipo *peer* son nodos que ofrecen una interficie gráfica al usuario. Aunque para restringir el ámbito del proyecto se ha definido la utilización de un total de 4 nodos del tipo *peer*, se podrían tener tantos nodos activos como se quisieran. El único requisito imprescindible es que el nombre de cada uno de ellos sea diferente. La forma de identificar a los nodos lo veremos en la parte de arranca del nodo.

4.2 INSTALAR NODO PEER

Siempre y cuando el sistema donde se va a poner en marcha el nodo disponga de los requisitos definidos en el apartado *2.1 Requisitos del sistema*, no se precisa de realizar instalación previa de cualquier otro elemento.

Por ello, la instalación del *peer* será bastante sencilla y únicamente se tendrá que descomprimir el fichero *peer.zip*.

Una vez realizada esto, se creará la siguiente estructura de directorios (para más información sobre el significado de cada directorio consultar el apartado *2.3 Estructura de directorios*):

- pfc

y en el directorio raíz existirán los siguientes ficheros:

- peer.bat
- log4j.properties
- jaxen.jar
- derby.jar
- log4j-1.2.16.jar
- jxta.jar

Ahora se configurará el número de nodo que se arrancará. Para ello, habrá que editar el fichero ***peer.bat*** y poner el número indicado (resaltado en rojo):

```
java -cp jxta.jar;derby.jar;log4j-1.2.16.jar;jdom-1.1.2.jar;jaxen.jar;. pfc.prototipo.PFC_PanelPrincipal 1
```

Como se puede observar, en este caso se arranca el nodo número 1.

Es muy importante que cada nodo tenga un número diferente ya que si no habría conflictos de nombres en la red colaborativa.

Una vez hecho y comprobado todo esto, el nodo *peer* estará listo para poder ponerse en marcha. Por ello, se lanzará el .bat y si todo ha funcionado correctamente se mostrará la interficie gráfica del nodo indicado.

Para arrancar el nodo 2, 3 y 4 se seguirán los mismos pasos indicando en el fichero ***peer.bat*** el número de nodo indicado.

Si se quiere ejecutar más de un nodo en la misma máquina tener en cuenta que se tiene que realizar en distintos directorios.

4.3 ARRANCAR NODO PEER

Ejecutar el siguiente fichero:

peer.bat

De esta forma se iniciar un nodo tipo *peer* con sus valores por defecto.

Hay que tener en cuenta que al iniciar el nodo *peer* por primera vez se crea la base de datos, se crea configuración del sistema y se crea configuración del entorno colaborativo.

Una vez arrancado, si se observa la estructura de directorios se puede observar que se han creado tres nuevos directorios:

- .cache
- documentos
- PFCGrupoBD

A parte, se han generado los siguientes ficheros:

- derby.log
- pfc_prototipo_trazas.log
- pfc_prototipo_bbdd_trazas.log
- pfc_prototipo_rendimiento_trazas.log

4.4 CONFIGURACIÓN

Cuando se arranca un nodo tipo *peer* se crea la configuración por defecto. Esta configuración contempla los siguientes puntos:

Parámetro	Valor
CONF_NombrePeer	PFC_Nodox (donde x es el parámetro recibido en main)
CONF_IdPeer	Crear identificador único en base al nombre del peer y al nombre del grupo
CONF_NombreGrupo	PFC_GrupoTrabajo
CONF_IdPipeMulticast	Crear identificador único en base al nombre del PipeMulticast(ComunicacionGrupo) y al nombre del grupo
CONF_IdPipeUnicast	Crear identificador único en base al nombre del PipeUnicast(ComunicacionPeer) y al nombre del grupo
CONF_SuperPeerIP	127.0.0.1
CONF_SuperPeerPuerto	9717
CONF_Directorio	Será el path de ejecución de la aplicación +"\documentos\"

En los nodos tipo *peer* se puede cambiar la IP/Puerto del *Superpeer* y del directorio donde residirán los documentos. Al tratarse una aplicación con interficie de usuario, consultar Manual de Administración para realizar estos cambios.

5 ANEXOS

5.1 IJ COMO CONSOLA DE BASE DE DATOS

Aunque se trata una herramienta para la consulta de base de datos directa (sin pasar por la propia aplicación p2p) y no hace falta una vez la solución ya la está utilizando el usuario final; puede ser de mucha utilidad poder atacar y extraer información directamente de la base de datos.

Para ello, *Derby* te ofrece una consola donde mediante una interfaz de texto, poder conectar y ejecutar cualquier sentencia de base de datos.

5.1.1 INSTALAR IJ

Para poder utilizar IJ hay que descargarse de la web de *DERBY*

http://db.apache.org/derby/

La release de los binarios que se distribuye (en nuestro caso la 10.8.1.2).

Esta release está compuesta por la siguiente estructura de directorios:

- db-derby-10.8.1.2-bin
  - bin
  - demo
  - docs
  - javadoc
  - lib
  - test

Con esto, se crea una variable de sistema con el nombre y contenido:

DERBY_HOME = ../ db-derby-10.8.1.2-bin/

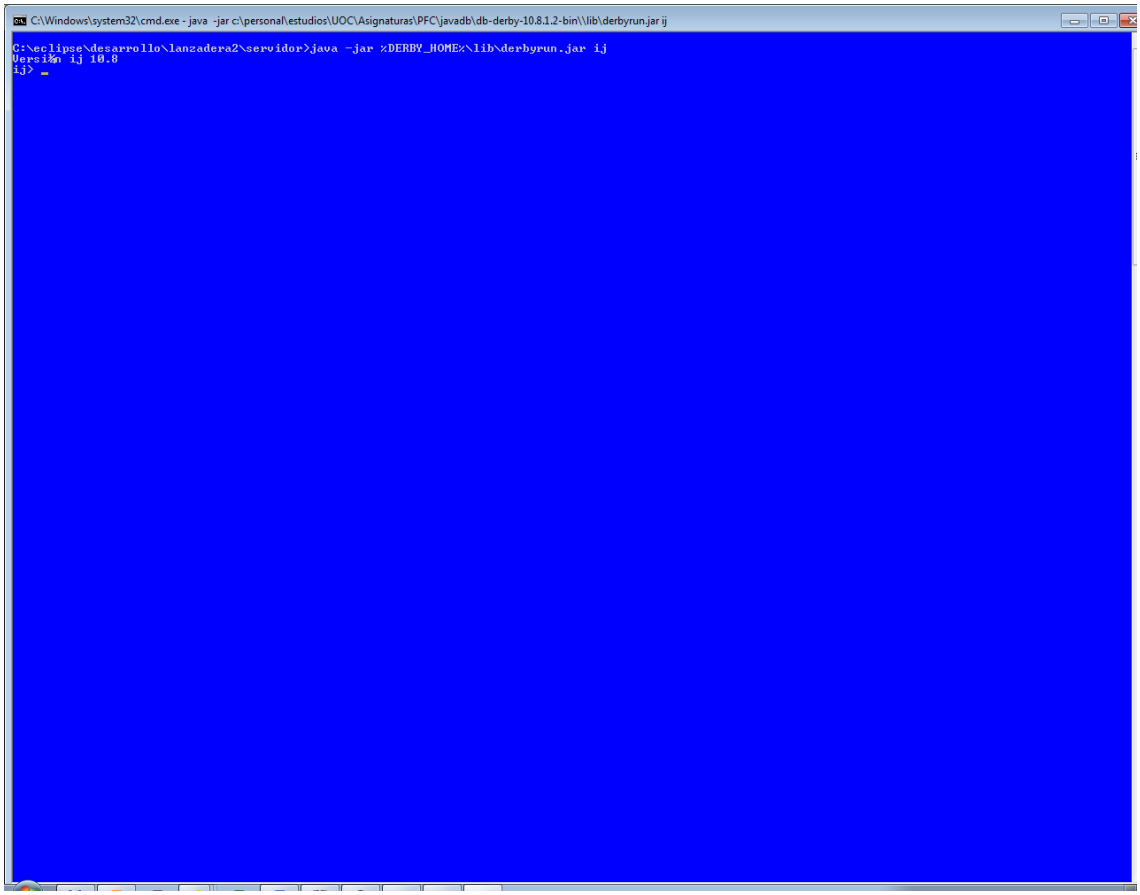
Y a su vez, se incluye esta variable en la el path del sistema.

5.1.2 ARRANCAR IJ

Para poder utilizar IJ, a parte de seguir los pasos anteriores, necesitamos de una base de datos donde conectarnos. Para ello, abriremos cualquier línea de comandos y podemos colocarnos en el directorio raíz de cualquier de nuestros nodos, a parte, ejecutar:

java -jar %DERBY_HOME%\lib\derbyrun.jar ij

De esta forma, se iniciará la línea de comandos del propio IJ de Derby:

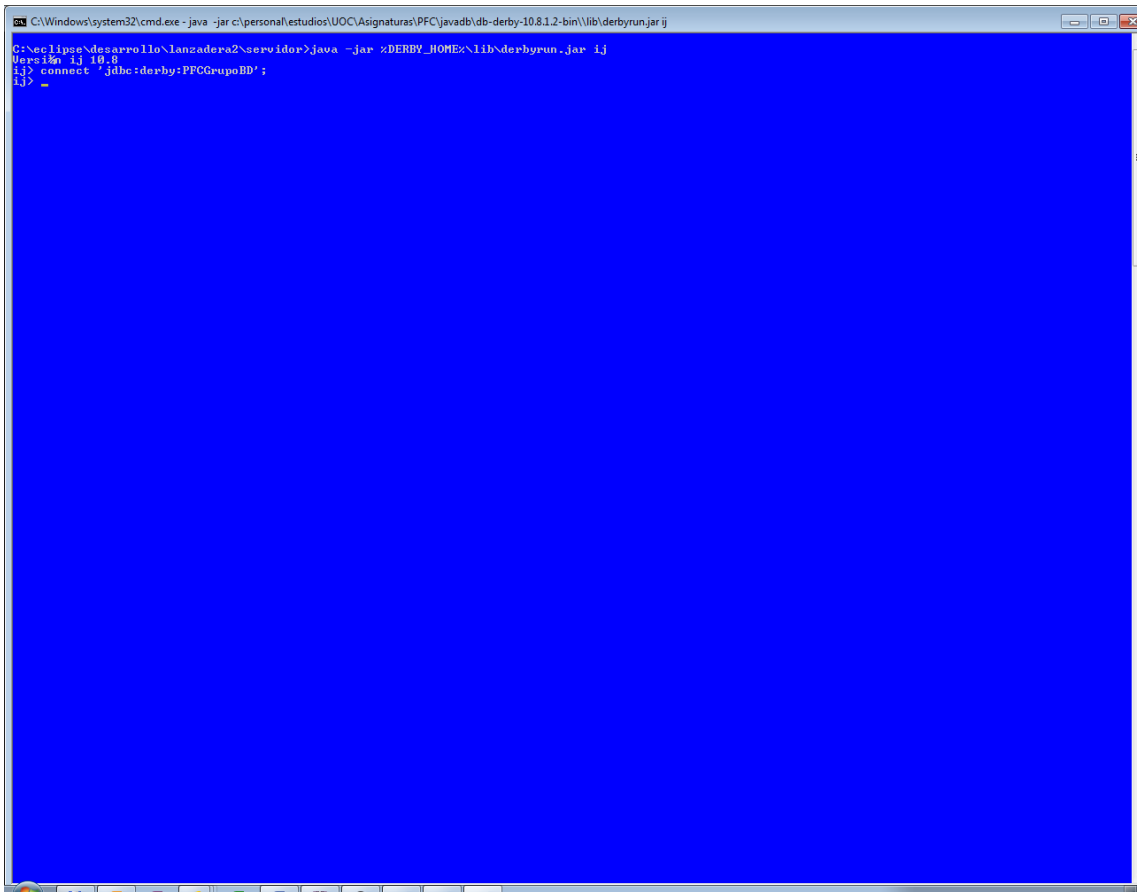


5.1.3 CONECTARSE A UNA BASE DE DATOS

Una vez iniciar IJ, lo primero que se tiene que hacer es conectarse a una base de datos. En nuestro caso nos conectaremos a la base de datos creada por el nodo *superpeer* (aunque cualquier nodo crea la misma base de datos):

connect 'jdbc:derby:PFCGrupoBD';

De esta forma se abre una conexión a una base de datos existente. La base de datos se llama *PFCGrupoBD* y tiene que existir en el directorio donde hemos ejecutado IJ.



A partir de este momento se puede lanzar cualquier sentencia SQL.