Introducción

En este laboratorio mostraremos la implementación existente de un proyecto de apuestas, que servirá de base para el desarrollo de un proyecto más completo en el dominio de las apuestas, que es lo que os ha pedido la empresa SINKING SOFT.

Objetivos

Los objetivos del laboratorio son los siguientes:

- Conocer cómo funciona la implementación actual: qué casos de uso se han implementado hasta el momento
- Entender la arquitectura con la que se ha construido la aplicación: las capas lógicas que forman y cómo se pueden desplegar en diferentes niveles o nodos físicos.

Y de paso: comprender que estamos desarrollando un **proyecto** definido según el **Proceso** Unificado de Desarrollo de Software (PUDS), y que por lo tanto tiene estas características: 1) está guiado por **casos de uso**, centrado en la **arquitectura**, y sigue un ciclo de vida **iterativo e incremental**.

Pasos a seguir

1. Descargar y probar el ejecutable resultado de la primera iteración

Descargar el fichero Bets-despliegue-1N.zip disponible en eGela y descomprimirlo en un directorio Bets-despliegue-1N.

Bets-despliegue-1N contiene el fichero Bets.jar con todas las clases que implementan el proyecto y los ficheros de internacionalización (.properties) y otro fichero de configuración (config.xml).

Ejecutar la aplicación haciendo doble click en Bets.jar

Podréis comprobar que ofrece 2 funcionalidades diferentes (**casos de uso** usando nuestra terminología). Intentad probarlo y ver qué se puede hacer y qué no se puede hacer. También podréis comprobar que se ha creado un fichero llamado bets.temp donde se almacena la base de datos objectDB, y que además se puede cambiar el idioma utilizado en la interfaz de usuario.

Pregunta: ¿Qué sucede si cerráis la aplicación y volvéis a lanzar la aplicación?

- ⇨ Que se han perdido los datos anteriores (las preguntas que hayáis podido crear) porque trabaja con una nueva BD inicializada con ciertos valores
- ⇨ Cambiad la configuración del fichero config.xml (abriéndolo con un editor de texto, y teniendo cuidado para grabarlo con el mismo nombre: config.xml)

En vez de <dataBaseOpenMode>**initialize**</dataBaseOpenMode>
poned <dataBaseOpenMode>**open**</dataBaseOpenMode>

Pregunta: ¿Las preguntas, son visibles para el resto de compañeros?

⇨ No, porque cada ordenador está trabajando de manera local con su propia base de datos, la cual no está compartida por el resto

Idea importante:

Siempre que queramos que desde distintas aplicaciones, ordenadores, etc. se acceda a algún RECURSO COMPARTIDO (bien sean datos, o funcionalidades, aplicaciones, etc.) será necesario definir un SERVIDOR donde se aloje y administre dicho recurso.

Conceptos a recordar sobre el PUDS:

Este proyecto está guiado por CASOS DE USO, ya que hay 2 para los cuales se han capturado y analizado los requisitos, diseñado, implementado y probado.

Este proyecto sigue un ciclo de vida ITERATIVO e INCREMENTAL, ya que se ha realizado una iteración para la cual ya tenemos un producto (software y documentación asociada). Habrá más iteraciones (3 más que deberéis realizar) las cuales producirán un incremento en el producto.

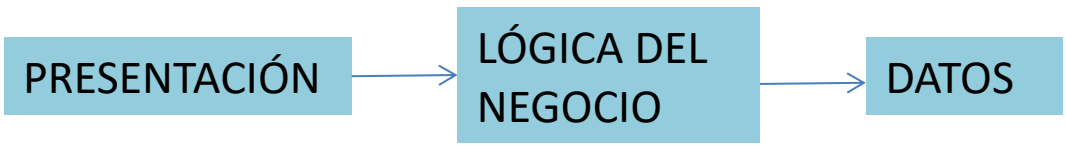
SI HAS LLEGADO A ESTE PUNTO, ESPERA POR FAVOR…

2. Ejecutar la aplicación en una arquitectura en 2 niveles (presentación y LN+BD)

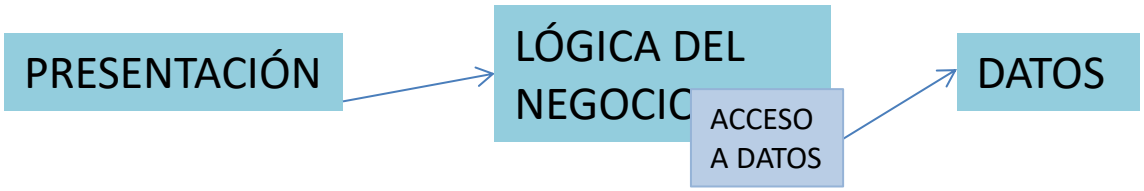
Afortunadamente, aunque tenemos pocos casos de uso todavía, hemos sido cuidadosos con el diseño de la arquitectura, porque si no hubiera sido así habría que rehacer muchas cosas cuando nos diéramos cuenta de que necesitábamos al menos un SERVIDOR DE DATOS.

En realidad, la arquitectura que hemos definido, además de poder trabajar con un servidor de datos, permite trabajar también con un servidor de lógica de negocio. La capa de datos permite compartir datos entre los distintos clientes (presentaciones) y la capa de lógica del negocio permite compartir funciones entre los distintos clientes (presentaciones)

Esta es la arquitectura de la aplicación:

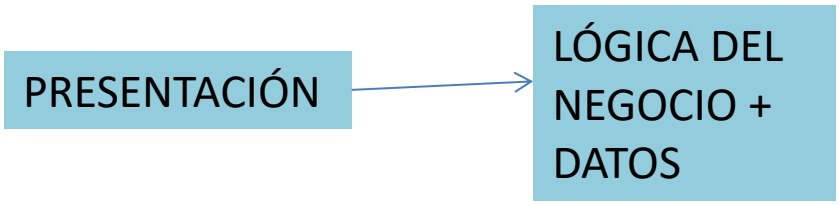


Donde en la lógica del negocio se diferencia la parte encargada del ACCESO A LOS DATOS de la que no accede a los datos.



Pues bien, la aplicación proporcionada contiene todos esos módulos los cuales se pueden configurar o desplegar de diferentes maneras.

En concreto vamos a definir una arquitectura con 2 despliegues diferentes en dos ordenadores; en uno el cliente con la presentación que invoca a la lógica del negocio remota (no local) y en el otro el servidor con la lógica del negocio invocada por el cliente. En dicho servidor se invoca a la capa de datos de manera local. Esta es una arquitectura en 2 niveles con cliente flaco y servidor gordo.



Para ello copiad el directorio Bets-despliegue-1N en 2 directorios llamados por ejemplo Bets-despliegue-P-2N y Bets -despliegue-LN+BD-2N

En ambos directorios modificad el fichero config.xml de esta manera: <businessLogic local="**false**"> y <database local="**true**">, para indicar que en esa configuración la lógica del negocio no se va a ejecutar de manera local sino remota.

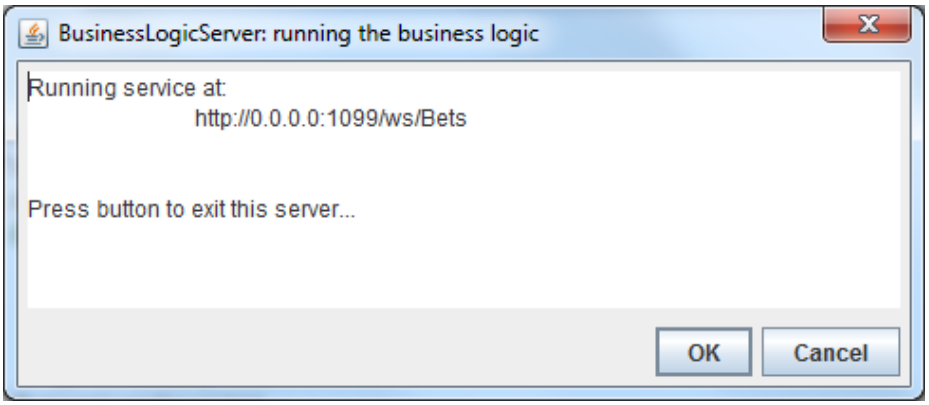
En el fichero MANIFEST.MF que se encuentra en el Bets.jar de Bets-despliegue-LN+BD-2N hay que modificar la clase que se ejecuta al hacer doble click en Bets.jar, para que en vez de que sea gui.ApplicationLauncher sea businessLogic.BusinessLogicServer. No hagáis cortar y pegar del contenido porque dará problemas con los espacios y con los caracteres de fin de línea; sustituid simplemente “gui.ApplicationLauncher” por “businessLogic.BusinessLogicServer”

```
Manifest-Version: 1.0
Class-Path: .
Main-Class: gui.ApplicationLauncher
```

Nota: un fichero .jar se puede abrir con un programa de descomprimir como .7z, winzip, winrar, etc. Extraed el fichero MANIFEST.MF que se encuentra en el directorio META-INF. Modificadlo con un editor de textos teniendo cuidado de que el nombre quede como MANIFEST.MF, y lo volvéis a meter en el fichero Bets.jar

Podéis ejecutar los dos DESPLIEGUES, esto es, el Bets.jar de Bets-despliegue-LN+BD-2N primero y el Bets.jar de Bets-despliegue-P-2N después.

Nota: sabemos que el servidor de lógica del negocio ha sido correctamente lanzado si al final nos muestra el mensaje “Press button to exit this server”.



Pregunta: ¿Qué sucede si ejecutáis primero el Bets.jar de Bets-despliegue-P-2N antes que el Bets.jar de Bets-despliegue-LN+BD-2N?

Pregunta: ¿Hemos solucionado el problema de compartición de los datos anterior?

- ⇨ No, lo que hay que hacer es que en un ordenador se ejecute el Bets.jar de Bets -despliegue-LN+BD-2N y en todos los demás ejecutemos solamente el Bets.jar de Bets-despliegue-P-2N pero configurando correctamente la lógica del negocio

  Hay que cambiar el <businessLogicNode>**0.0.0.0**</businessLogicNode> de todos los clientes y poner, en vez de **0.0.0.0**, la dirección IP del ordenador con el despliegue Bets-despliegue-LN+BD-2N, esto es, el que está ejecutando el Bets.jar de Bets-despliegue-LN+BD-2N.

Nota: Para saber la dirección IP de un equipo: Inicio => Ejecutar => cmd => IPCONFIG

Concepto a recordar sobre el PUDS:

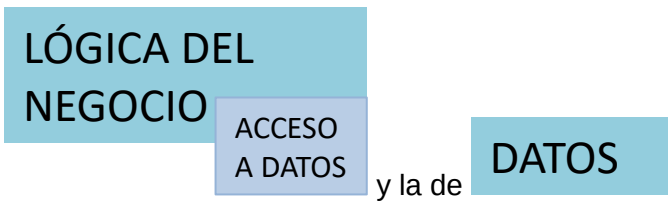
Este proyecto se ha desarrollado CENTRADO EN LA ARQUITECTURA, ya que permite realizar diferentes despliegues del mismo; permite definir diferentes tipos de arquitecturas Cliente/Servidor.

SI HAS LLEGADO A ESTE PUNTO, ESPERA POR FAVOR…

3. Ejecutar la aplicación en una arquitectura en 3 niveles (presentación, LN y BD)

En este punto vamos a definir una arquitectura con los 3 niveles separados así:

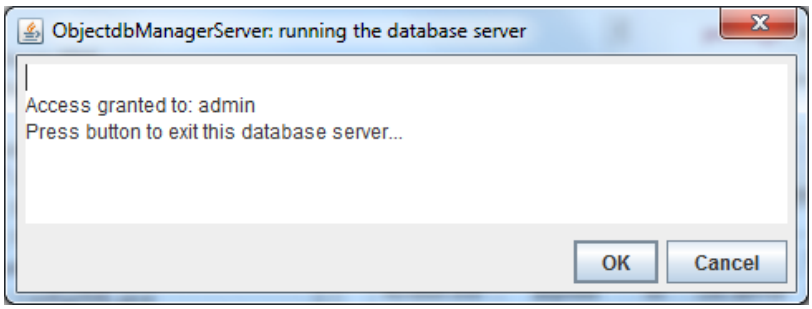
Para ello usaremos el despliegue de presentación anterior y los dos siguientes:



Para crearlos hay que copiar el directorio Bets-despliegue-1N en 3 directorios llamados por ejemplo Bets-despliegue-P-3N, Bets-despliegue-LN-3Ny Bets-despliegue-BD-3N

En el fichero config.xml de todos ellos poned lo siguiente: <businessLogic local="**false**"> y <database local="**false**">, para indicar que en esa configuración ni la lógica del negocio ni los datos son locales, sino remotos.
En el MANIFEST.MF del Bets.jar del despliegue Bets-despliegue-LN-3N hay que poner:
Main-Class: businessLogic.BusinessLogicServer
En el MANIFEST.MF del Bets.jar del despliegue Bets-despliegue-BD-3N hay que poner:
Main-Class: dataAccess.ObjectdbManagerServer

Ejecutar los tres DESPLIEGUES en este orden. Primero el correspondiente al servidor de la base de datos (Bets-despliegue-BD-3N), que mostrará lo siguiente:



Después el servidor de la lógica del negocio. Para que se cree correctamente la BD en el servidor de datos habrá que lanzarlo dos veces: la primera con la opción <dataBaseOpenMode>**initialize**</dataBaseOpenMode> en el fichero config.xml, y la segunda con la opción <dataBaseOpenMode>**open**</dataBaseOpenMode>. Y, por último se lanzará la presentación: el Bets.jar de Bets-despliegue-P-3N.

Pregunta: ¿Qué sucede si se ejecutan los despliegues en otro orden?

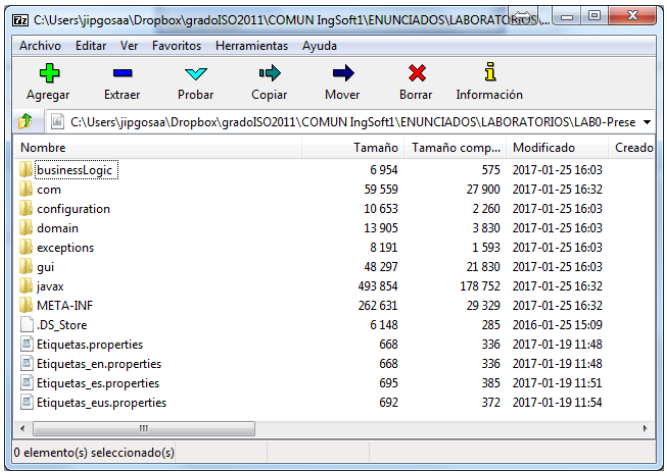
Por último, no olvidar que el despliegue físico en tres tipos de ordenadores se consigue configurando correctamente los IP de los ordenadores que tengan la lógica del negocio lanzada y el servidor de la base de datos.
Para ello:

1) Hay que cambiar el <businessLogicNode>**0.0.0.0**</businessLogicNode> de todos los clientes y poner la dirección IP del ordenador que está ejecutando el Bets.jar de Bets-despliegue-LN-3N en vez de **0.0.0.0**.
2) Hay que cambiar el <databaseNode>**0.0.0.0**</databaseNode> de todos los ordenadores que tengan ejecutando la lógica del negocio y poner la dirección IP del ordenador que está ejecutando el Bets.jar de Bets-despliegue-BD-3N en vez de **0.0.0.0**.

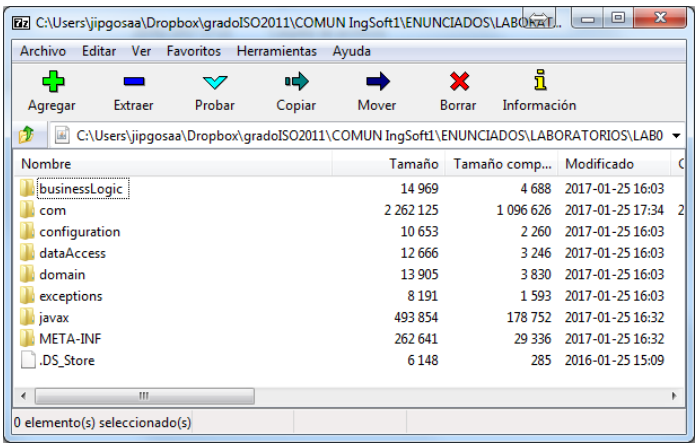
En este punto se puede intentar hacer entre todos lo siguiente: un ordenador que ejecute el servidor de datos, dos ordenadores que ejecuten la lógica del negocio conectada al servidor de datos anterior, la mitad de resto de ordenadores que ejecuten la presentación con una de las lógicas del negocio, y la otra mitad que ejecuten la presentación conectada al otro servidor de lógica del negocio.

Por último, hay que indicar que aunque el fichero Bets.jar incluido en cada uno de los despliegues ha sido el mismo, lo cierto es que no tienen por qué estar todas las clases en el cliente y en el servidor. Se podrían eliminar ciertas partes de cada Bets.jar. En concreto las siguientes:

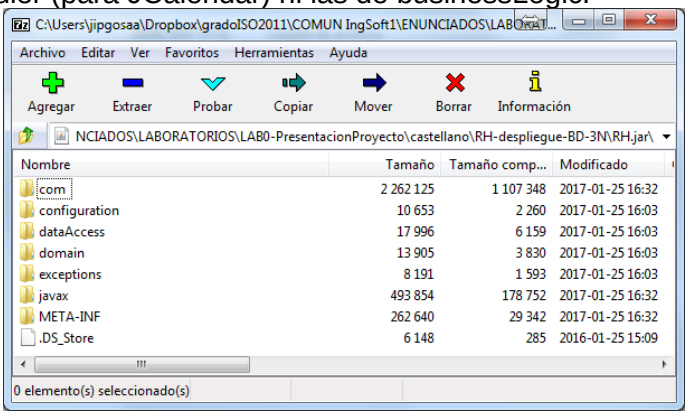
En el Bets.jar de Bets-despliegue-P-3N no se necesita ni las clases de dataAccess, ni la clases BusinessLogicServer y BLFacadeImplementation de businessLogic, ni el paquete com.objectdb:



En el Bets.jar de Bets-despliegue-LN-3N no se necesitan las clases de gui, ni el paquete com.toedler (para JCalendar), ni la clase ObjectdbManagerServer



En el Bets.jar de Bets-despliegue-BD-3N no se necesitan ni las clases de gui, ni el paquete com.toedler (para JCalendar) ni las de businessLogic.



TAREA A REALIZAR: Trabajar en la CAPTURA DE REQUISITOS, definir nuevos CASOS de USO y/o modificar los existentes.