3.4: Arquitecturas Software de Varios Niveles

A. Goñi, J. Ibáñez, J. Iturrioz, J.A. Vadillo

informatika fakultatea
facultad de informática

OCW 2013

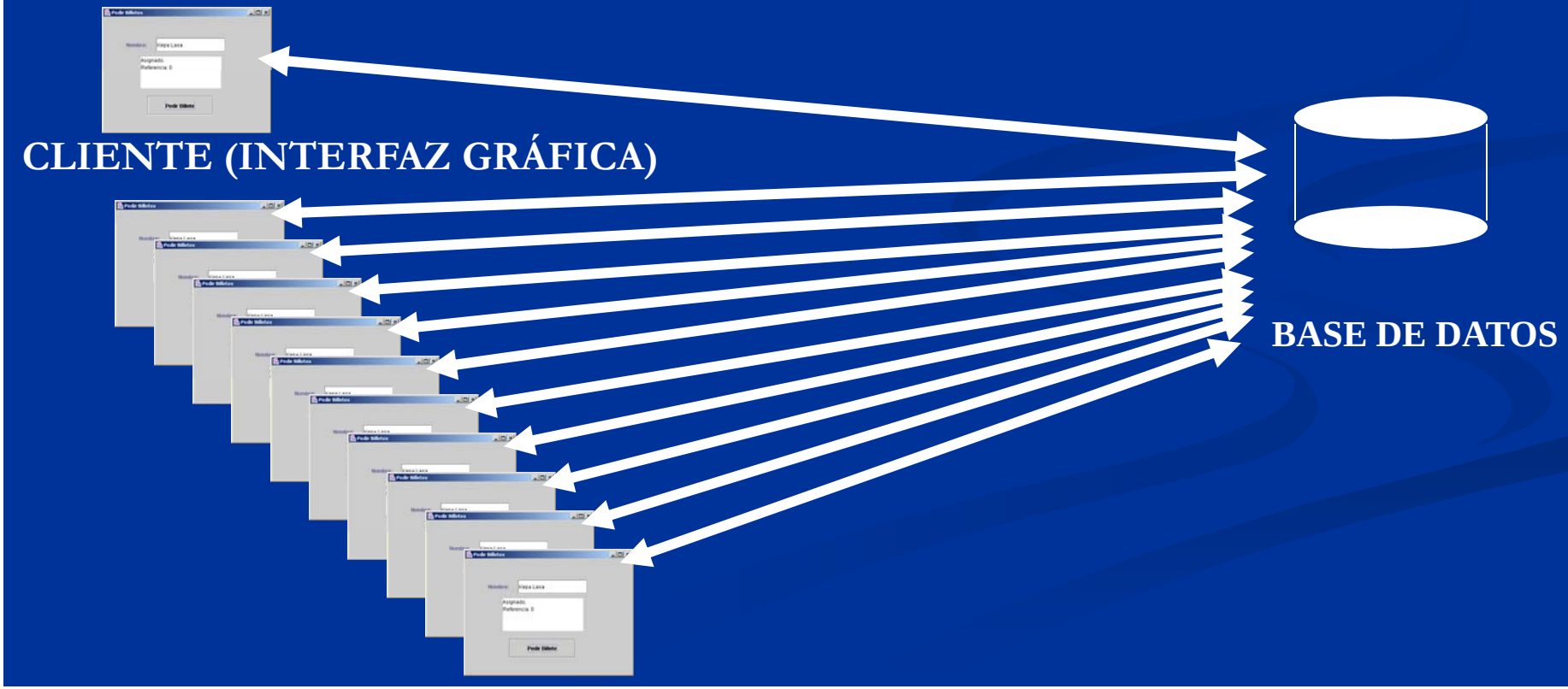
eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea

Índice

- Introducción
- Arquitectura lógica del software en capas: presentación, lógica del negocio y datos.
- Arquitectura física en 2 niveles: cliente gordo/servidor flaco
- Arquitectura física en 2 niveles: cliente flaco/servidor gordo
- Arquitectura física en 3 (o más) niveles

Introducción

- Hay aplicaciones que deben ejecutar operaciones
  - de manera CONCURRENTE, SEGURA, FIABLE y EFICIENTE



Introducción

Ejemplos:

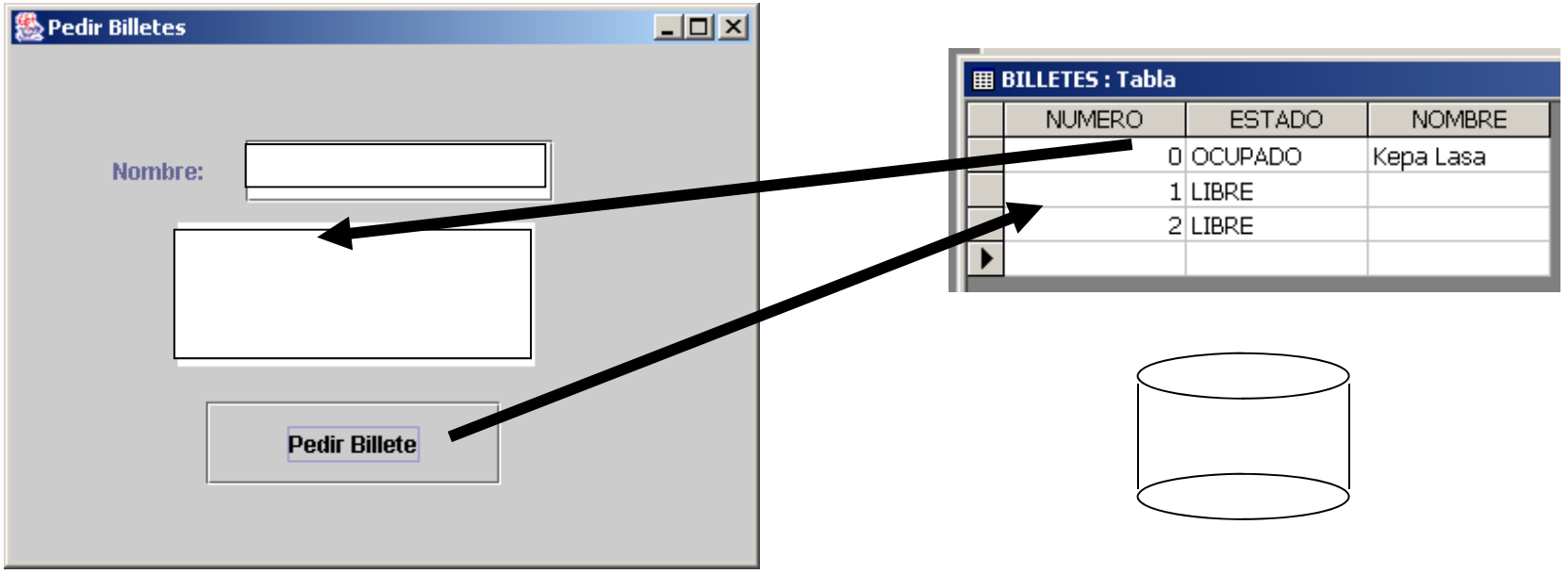
Terminales donde se pueden comprar entradas para espectáculos

Cajeros automáticos conectados a una central

Oficinas de reservas y compra de vuelos y viajes.

- SOLUCIÓN: usar una
  - **Arquitectura con despliegue de componentes software en el lado del servidor**
  - Componente: código que implementa un conjunto conocido de interfaces

Ejemplo: comprar billetes para espectáculos

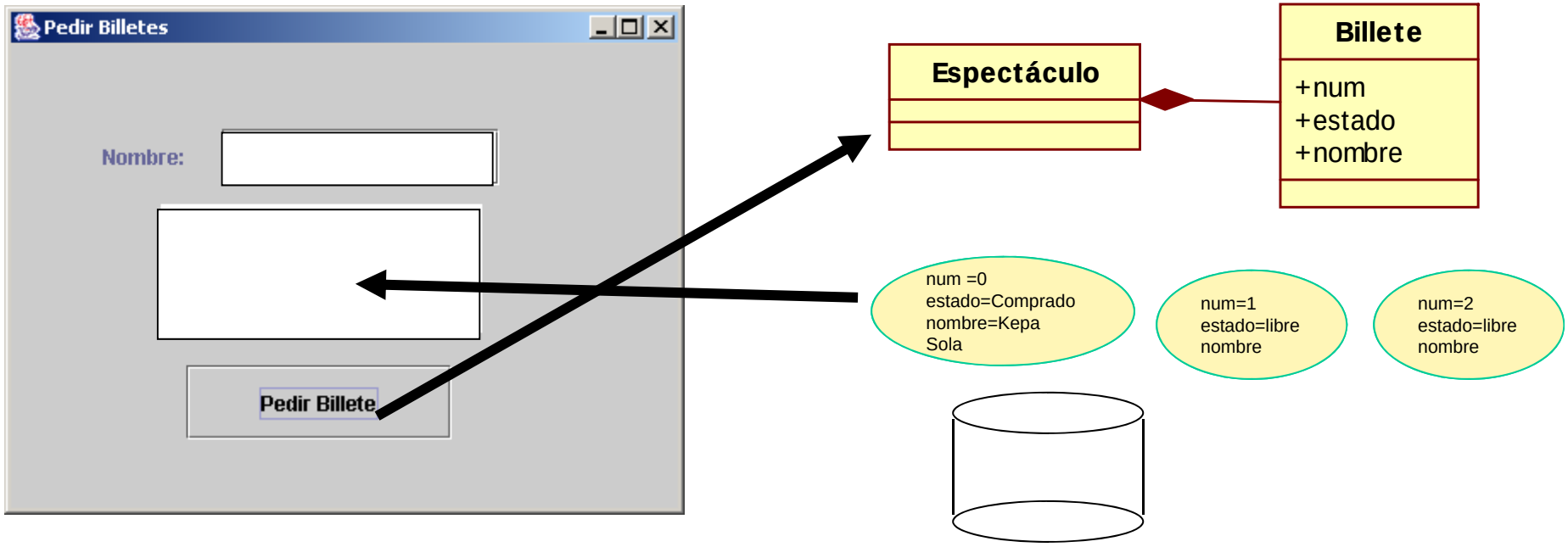


BASE DE DATOS RELACIONAL

```
public class PedirBilleteNO3NIVELES extends JFrame {
// Nota: NO ESTÁ COMPLETA !!
   JLabel jLabel1 = new JLabel(“Nombre:”);
   JButton jButton1 = new JButton(“Pedir Billete”);


public PedirBilleteNO3NIVELES() {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    conexion=DriverManager.getConnection("jdbc:odbc:Billetes"); }
void jButton1_actionPerformed(ActionEvent e) {
ResultSet rs = sentencia.executeQuery("SELECT NUMERO FROM"+
                    " BILLETES WHERE ESTADO=\'LIBRE\'");
     if (rs.next()) {
     int act = sentencia.executeUpdate("UPDATE BILLETES"+
      " SET ESTADO='OCUPADO', NOMBRE = "+jTextField1.getText()+
      " WHERE NUMERO="+rs.getString("NUMERO")+
      " AND ESTADO='LIBRE'");
     if (act>0) jTextArea1.append("Asignado. \nReferencia: "+n+"\n");
     else jTextArea1.append("Error al asignar billete"); }}
public static void main (String []arg) {
    PedirBilleteNO3NIVELES b = new PedirBilleteNO3NIVELES();
    b.setVisible(true);}}
```

Ejemplo: comprar billetes para espectáculos



```
public class PedirBilleteNO3NIVELES extends JFrame {
// Nota: NO ESTÁ COMPLETA !!
   JLabel jLabel1 = new JLabel(“Nombre:”);
   JButton jButton1 = new JButton(“Pedir Billete”);
   public static String ESPECTACULO = “Oscar 2011”;
   ObjectContainer db;

public PedirBilleteNO3NIVELES() {
    db=DB4oManager.getContainer();
}
void jButton1_actionPerformed(ActionEvent e) {
Espectaculo esp= Espectaculo.getEspectaculo(ESPECTACULO);
 Billete b= esp.getBilleteLibre();
 int num=b.comprar(jTextField1.getText());
 jTextArea1.append("Asignado. \nReferencia: "+n+"\n");
 db.store(b);
}

public static void main (String []arg) {
    PedirBilleteNO3NIVELES b = new PedirBilleteNO3NIVELES();
    b.setVisible(true);}}
```

Problema…

- En las implementaciones anteriores NO SE SEPARA el código correspondiente a los siguientes aspectos:
  - Presentación
  - Lógica del negocio
  - Datos
- O lo que es lo mismo:
  - NO SE APLICA UNA ARQUITECTURA SOFTWARE DE VARIOS NIVELES

```
public class PedirBilleteNO3NIVELES extends JFrame {
// Nota: NO ESTÁ COMPLETA !!
  JLabel jLabel1 = new JLabel("Nombre:");
  JButton jButton1 = new JButton("Pedir Billete");
   public static String ESPECTACULO = "Oscar 2011";
   ObjectContainer db;

public PedirBilleteNO3NIVELES() {
    db=DB4oManager.getContainer();
}
void jButton1_actionPerformed(ActionEvent e) {
Espectaculo esp= db.getEspectaculo(ESPECTACULO);
 Billete b= esp.getBilleteLibre();
 int num=b.comprar(jTextField1.getText());
 jTextArea1.append("Asignado. \nReferencia: "+n+"\n");
 db.store(b);
}

public static void main (String []arg) {
    PedirBilleteNO3NIVELES b = new PedirBilleteNO3NIVELES();
    b.setVisible(true);}}
```

PRESENTACIÓN

```
public class PedirBilleteNO3NIVELES extends JFrame {
// Nota: NO ESTÁ COMPLETA !!
   JLabel jLabel1 = new JLabel("Nombre:");
   JButton jButton1 = new JButton("Pedir Billete");
   public static String ESPECTACULO = "Oscar 2011";
   ObjectContainer db;

public PedirBilleteNO3NIVELES() {
    db=DB4oManager.getContainer();
}
void jButton1_actionPerformed(ActionEvent e) {
Espectaculo esp= db.getEspectaculo(ESPECTACULO);
 Billete b= esp.getBilleteLibre();
 int num=b.comprar(jTextField1.getText());
 jTextArea1.append("Asignado. \nReferencia: "+n+"\n");
 db.store(b);
}

public static void main (String []arg) {
   PedirBilleteNO3NIVELES b = new PedirBilleteNO3NIVELES();
   b.setVisible(true);}}
```

ACCESO A DATOS

```
public class PedirBilleteNO3NIVELES extends JFrame {
// Nota: NO ESTÁ COMPLETA !!
   JLabel jLabel1 = new JLabel("Nombre:");
   JButton jButton1 = new JButton("Pedir Billete");
   public static String ESPECTACULO = "Oscar 2011";
   ObjectContainer db;
public PedirBilleteNO3NIVELES() {
    db=DB4oManager.getContainer();
}
void jButton1_actionPerformed(ActionEvent e) {
Espectaculo esp= db.getEspectaculo(ESPECTACULO);
 Billete b= esp.getBilleteLibre();
 int num=b.comprar(jTextField1.getText());
 jTextArea1.append("Asignado. \nReferencia: "+n+"\n");
 db.store(b);
}

public static void main (String []arg) {
    PedirBilleteNO3NIVELES b = new PedirBilleteNO3NIVELES();
    b.setVisible(true);}}
```

LÓGICA DEL NEGOCIO

```
public class PedirBilleteNO3NIVELES extends JFrame {
// Nota: NO ESTÁ COMPLETA !!
   JLabel jLabel1 = new JLabel("Nombre:");
   JButton jButton1 = new JButton("Pedir Billete");
```

PRESENTACIÓN

```
public PedirBilleteNO3NIVELES() {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    conexion=DriverManager.getConnection("jdbc:odbc:Billetes"); }
void jButton1_actionPerformed(ActionEvent e) {
ResultSet rs = sentencia.executeQuery("SELECT NUMERO FROM"+
                    " BILLETES WHERE ESTADO=\'LIBRE\'");
     if (rs.next()) {
     int act = sentencia.executeUpdate("UPDATE BILLETES"+
      " SET ESTADO='OCUPADO', NOMBRE = "+jTextField1.getText()+
      " WHERE NUMERO="+rs.getString("NUMERO")+
      " AND ESTADO='LIBRE'");
     if (act>0) jTextArea1.append("Asignado. \nReferencia: "+n+"\n");
     else jTextArea1.append("Error al asignar billete"); }}
public static void main (String []arg) {
    PedirBilleteNO3NIVELES b = new PedirBilleteNO3NIVELES();
    b.setVisible(true);}}
```

ACCESO A DATOS

```
public class PedirBilleteNO3NIVELES extends JFrame {
// Nota: NO ESTÁ COMPLETA !!
   JLabel jLabel1 = new JLabel("Nombre:");
   JButton jButton1 = new JButton("Pedir Billete");

public PedirBilleteNO3NIVELES() {
   Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   conexion=DriverManager.getConnection("jdbc:odbc:Billetes"); }
void jButton1_actionPerformed(ActionEvent e) {
ResultSet rs = sentencia.executeQuery("SELECT NUMERO FROM"+
    " BILLETES WHERE ESTADO=\'LIBRE\'");
    if (rs.next()) {
    int act = sentencia.executeUpdate("UPDATE BILLETES"+
     " SET ESTADO='OCUPADO', NOMBRE = "+jTextField1.getText()+
     " WHERE NUMERO="+rs.getString("NUMERO")+
     " AND ESTADO='LIBRE'");
    if (act>0) jTextArea1.append("Asignado. \nReferencia: "+n+"\n");
    else jTextArea1.append("Error al asignar billete"); }}
public static void main (String []arg) {
   PedirBilleteNO3NIVELES b = new PedirBilleteNO3NIVELES();
   b.setVisible(true);}}
```

```
public class PedirBilleteNO3NIVELES extends JFrame {
// Nota: NO ESTÁ COMPLETA !!
   JLabel jLabel1 = new JLabel(“Nombre:”);
   JButton jButton1 = new JButton(“Pedir Billete”);
```

LÓGICA DEL NEGOCIO

```
public PedirBilleteNO3NIVELES() {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    conexion=DriverManager.getConnection("jdbc:odbc:Billetes"); }
void jButton1_actionPerformed(ActionEvent e) {
ResultSet rs = sentencia.executeQuery("SELECT NUMERO FROM"+
                    " BILLETES WHERE ESTADO=\'LIBRE\'");
     if (rs.next()) {
     int act = sentencia.executeUpdate("UPDATE BILLETES"+
      " SET ESTADO='OCUPADO', NOMBRE = "+jTextField1.getText()+
      " WHERE NUMERO="+rs.getString("NUMERO")+
      " AND ESTADO='LIBRE'");
     if (act>0) jTextArea1.append("Asignado. \nReferencia: "+n+"\n");
     else jTextArea1.append("Error al asignar billete"); }}
public static void main (String []arg) {
    PedirBilleteNO3NIVELES b = new PedirBilleteNO3NIVELES();
    b.setVisible(true);}}
```

Arquitectura lógica del software en niveles (capas)

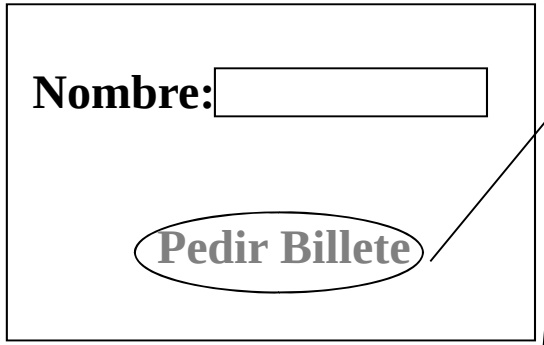
- NIVEL (CAPA) DE PRESENTACIÓN
  - interfaces de usuario y la interacción con el mismo
- NIVEL (CAPA) DE LÓGICA DEL NEGOCIO
  - resolver los problemas del negocio
  - implementar las reglas propias del negocio.
- NIVEL (CAPA) DE DATOS
  - BD donde se proporciona la persistencia

SE APLICA UNA ARQUITECTURA LÓGICA EN VARIAS CAPAS SI ESTAS ESTÁN SEPARADAS EN CLASES O COMPONENTES DISTINTOS

Arquitectura lógica del software en niveles

MÁS EXTENSIBILIDAD

1.- NIVEL DE PRESENTACIÓN



INTERFAZ GRÁFICO DE USUARIO

2.- NIVEL DE LÓGICA DEL NEGOCIO

```
public class GB
  implements GestorBilletes {
...
public int getBillete
  (String nom) {...}
```

CLASES CON OPERACIONES PROPIAS DEL NEGOCIO

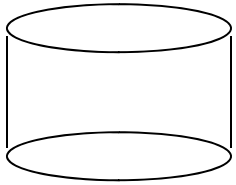
- **inicializarSala**
- **getBillete,...**

Aquí se pueden aplicar reglas del negocio: por cada 10 billetes comprados se regala uno, etc...

3.- NIVEL DE DATOS

```
SELECT ...
INSERT ...
```

ObjectContainer db
db.store()



BASE DE DATOS

Arquitectura lógica del software en niveles

- Ventaja: Se puede cambiar cada uno de los niveles minimizando los cambios en los otros niveles

UNA ARQUITECTURA LÓGICA DEL SOFTWARE EN VARIOS NIVELES FAVORECE LA EXTENSIBILIDAD Y REUTILIZACIÓN DEL SOFTWARE

El diseño de los casos de uso que hemos realizado es compatible con una arquitectura software en varios niveles

Flujo de Eventos: PEDIR BILLETE
****

- El usuario proporciona su nombre.
- El sistema busca si hay billetes libres.
- Si encuentra alguno entonces lo asigna, guarda el nombre y devuelve el número de billete

Diagrama de secuencia (BD OO)

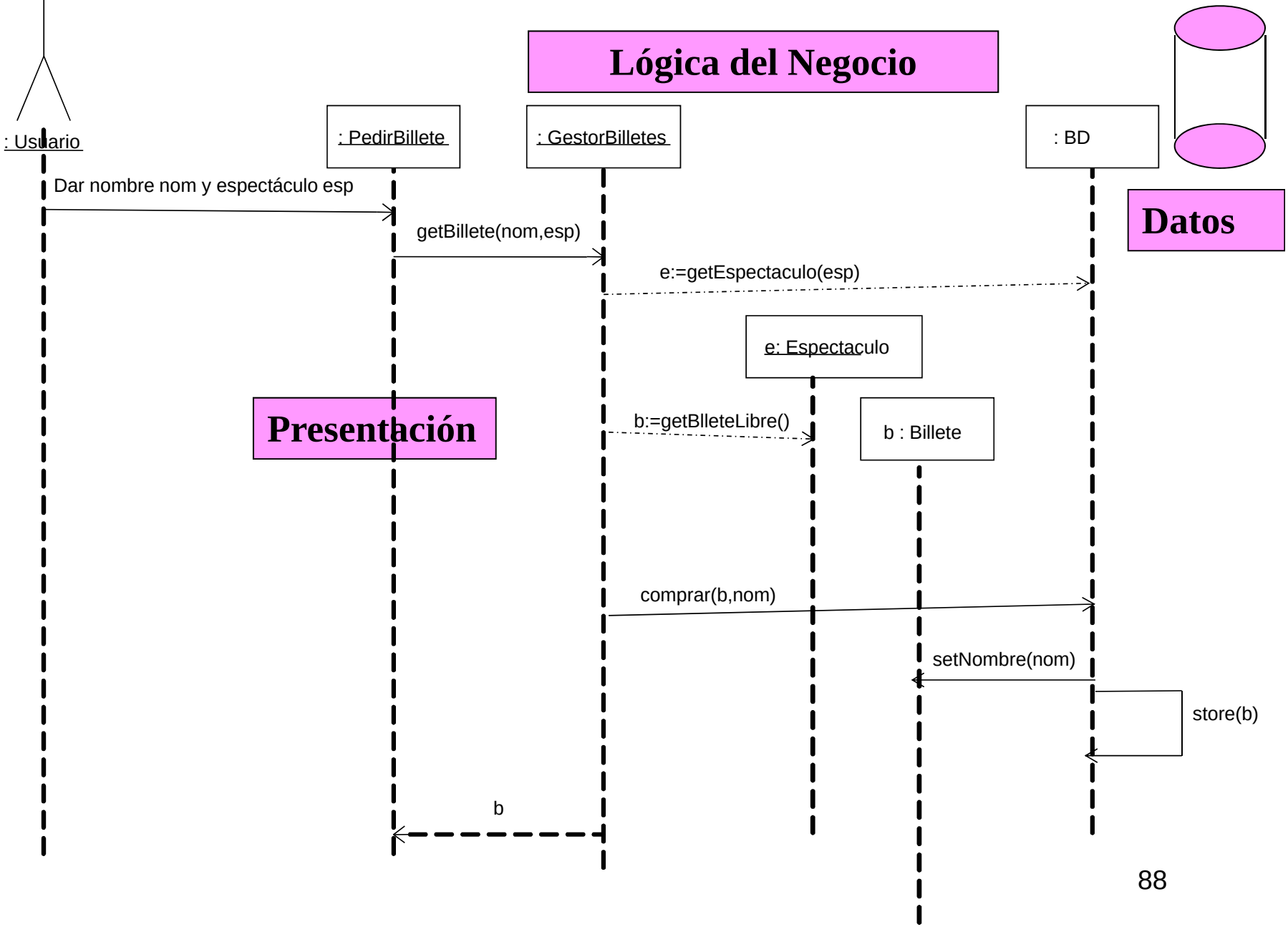
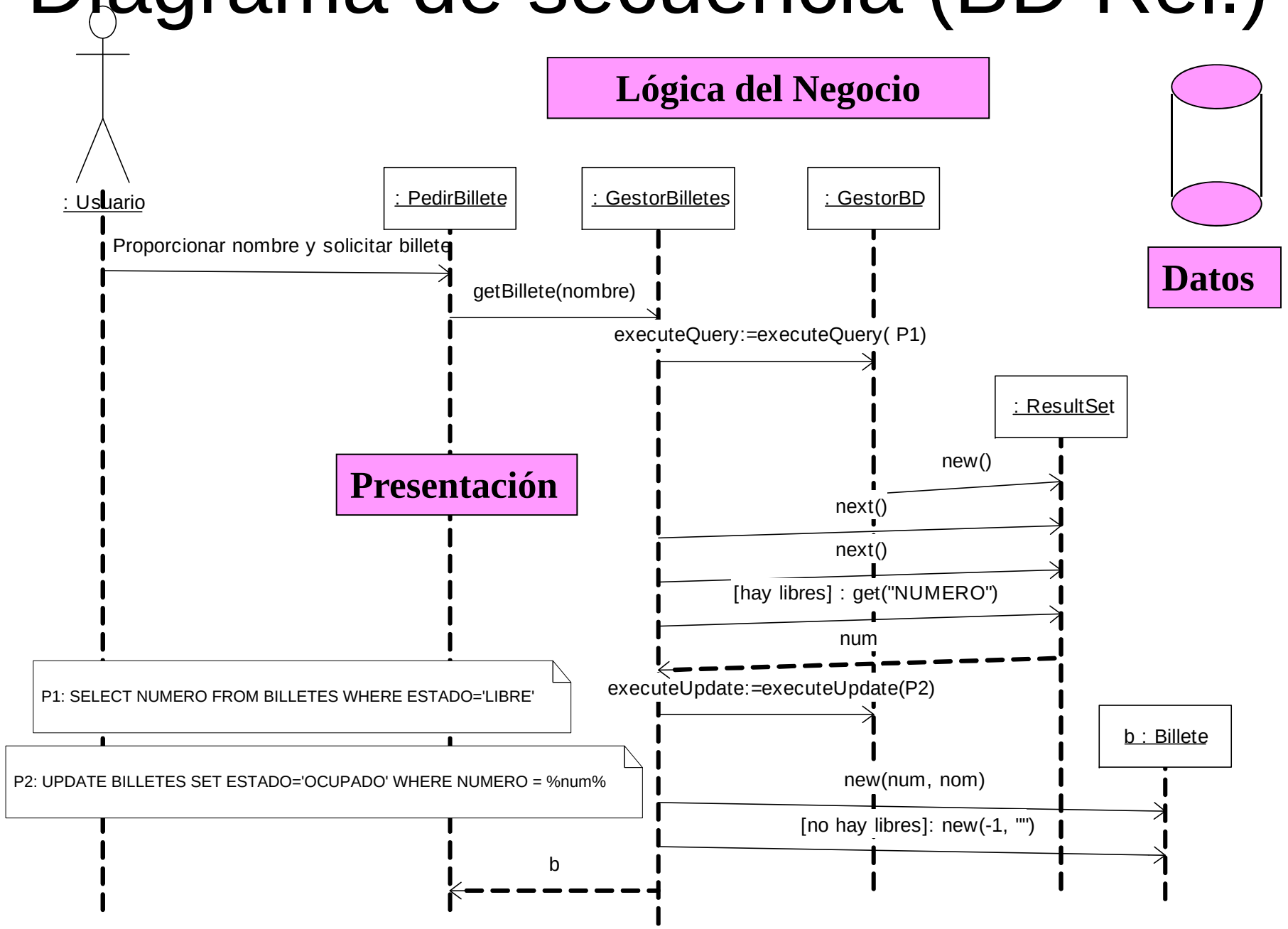


Diagrama de secuencia (BD Rel.)



Arquitectura física del software en niveles

- **Las CAPAS LÓGICAS se pueden desplegar en distintos NODOS/NIVELES FÍSICOS**
- Arquitectura en 2 niveles
  - Capa de presentación en NODO CLIENTE
  - Capa de datos en NODO SERVIDOR (de BD)
  - ¿Y la capa de lógica del negocio?
    - En el CLIENTE: junto con el nivel de presentación
    - PARTE podría juntarse con el nivel de datos
- Arquitectura en 3 niveles (o más)
  - Cada nivel, al menos, en un nodo distinto

Arquitectura física en 2 niveles: cliente gordo/servidor flaco

- El nivel de presentación y el de la lógica del negocio se unen en un nodo. En el otro queda el nivel de datos.

- **Comunicación entre Cliente y Servidor en SQL / db4o**
- **Se necesitan APIs como por ejemplo JDBC y/o ODBC, db4o**
- **Deben instalarse DRIVERS de la BD en todos los clientes**

CLIENTE

Presentación

```
public class PedirBillete2NivCliGordo extends JFrame {
GestorBilletes2NivCliGordo gestorBilletes;
void jButton1_actionPerformed(ActionEvent e) {
   int res = gestorBilletes.getBillete(jTextField1.getText()).getNum();
    if (res<0) jTextArea1.append("Error al asignar billete");
    else jTextArea1.append("Asignado. \nReferencia: "+res+"\n");} }
```

Lógica del Negocio

```
public class GestorBilletesBD
      implements GestorBilletes2NivCliGordo
{ public GestorBilletesBD() {
   Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   conexion=DriverManager.getConnection("jdbc:odbc:Billetes");}
   public Billete getBillete(String nom)
    {ResultSet rs = sent.executeQuery("SELECT NUMERO…";);
      int act = sent.executeUpdate("UPDATE BILLETES …“);
      if (act>0) return new Billete(n,nom); // Núm. billete asignado
     else return new Billete(-1,"");; } // No había ninguno libre}}}
```

DEFINIR FUENTE DATOS ODBC “Billetes”

E INSTALAR LA CLASE sun.jdbc.odbc.JdbcOdbcDriver

BD

Datos

BILLETES : Tabla

	NUMERO	ESTADO	NOMBRE
	0	OCUPADO	Kepa Lasa
	1	LIBRE	
	2	LIBRE	

SERVIDOR

CLIENTE

Presentación

```
public class PedirBillete2NivCliGordo extends JFrame {
GestorBilletes2NivCliGordo gestorBilletes;
void jButton1_actionPerformed(ActionEvent e) {
   int res = gestorBilletes.getBillete(jTextField1.getText()).getNum();
    if  (res<0) jTextArea1.append("Error al asignar billete");
    else jTextArea1.append("Asignado. \nReferencia: "+res+"\n");} }
```

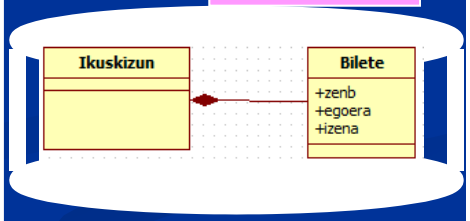
Lógica del Negocio

```
public class GestorBilletesBD
      implements GestorBilletes2NivCliGordo
{ public GestorBilletesBD () {
   db=DB4oManager.getContainer();
}
   public Billete getBillete(String nom)
    {Espectaculo esp= db.get Espectaculo (nom);
     Billete b=esp.getBilleteLibre();
     int num=b.comprar(nom);
     db.store(b);
 return b;}
```

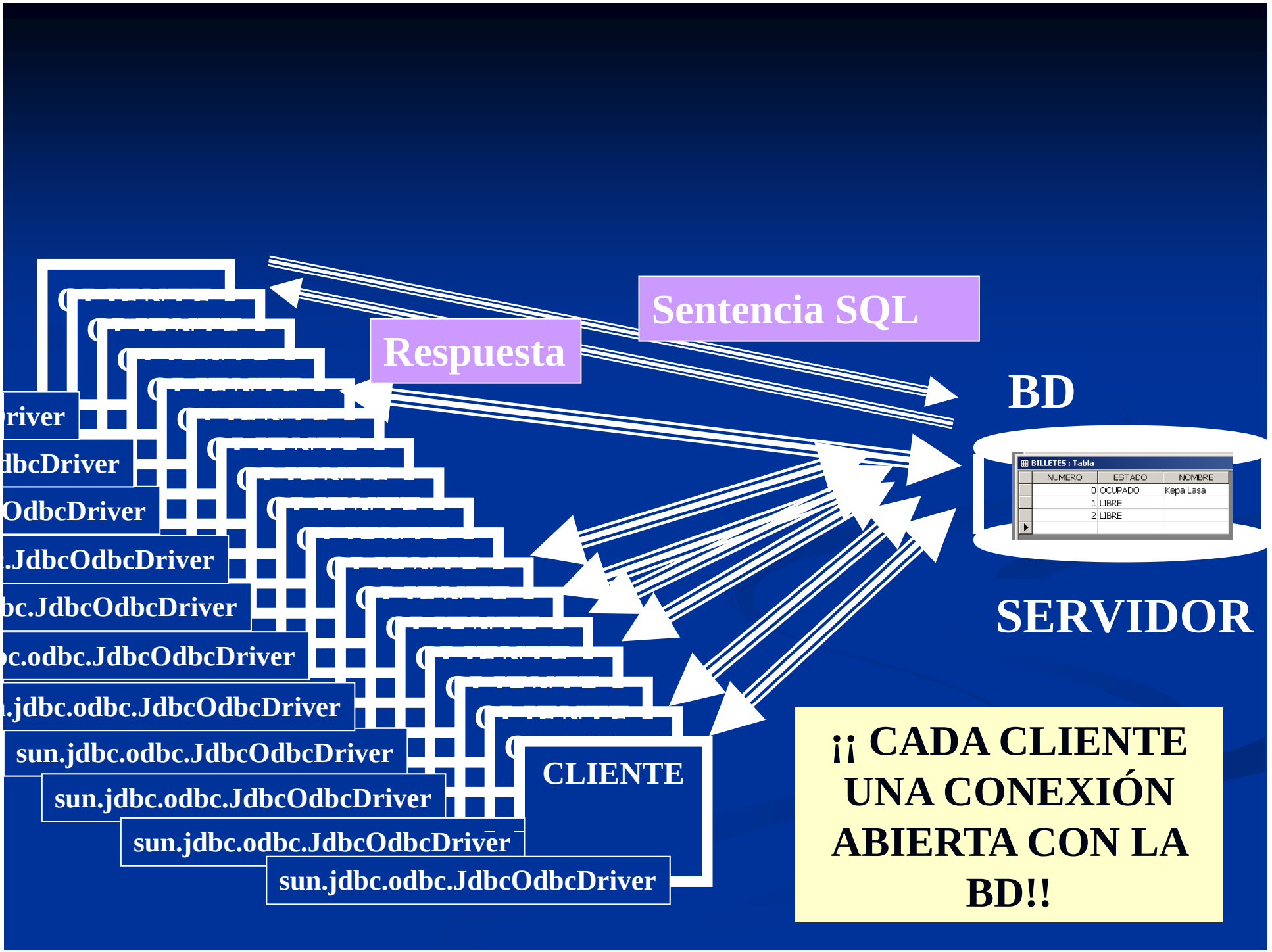
INSTALAR LA CLASE com.db4o.DB4oManager

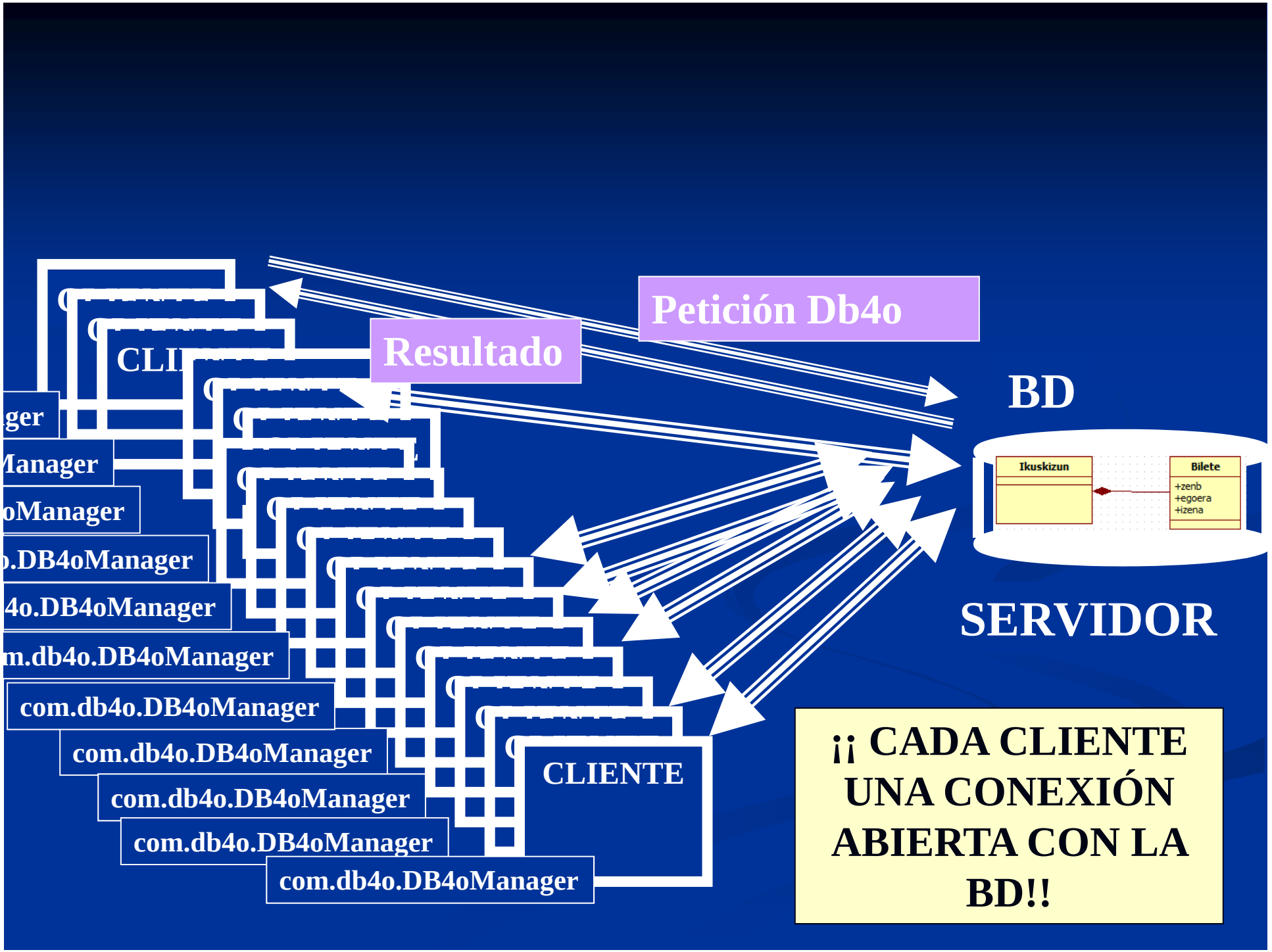
BD

Datos



SERVIDOR

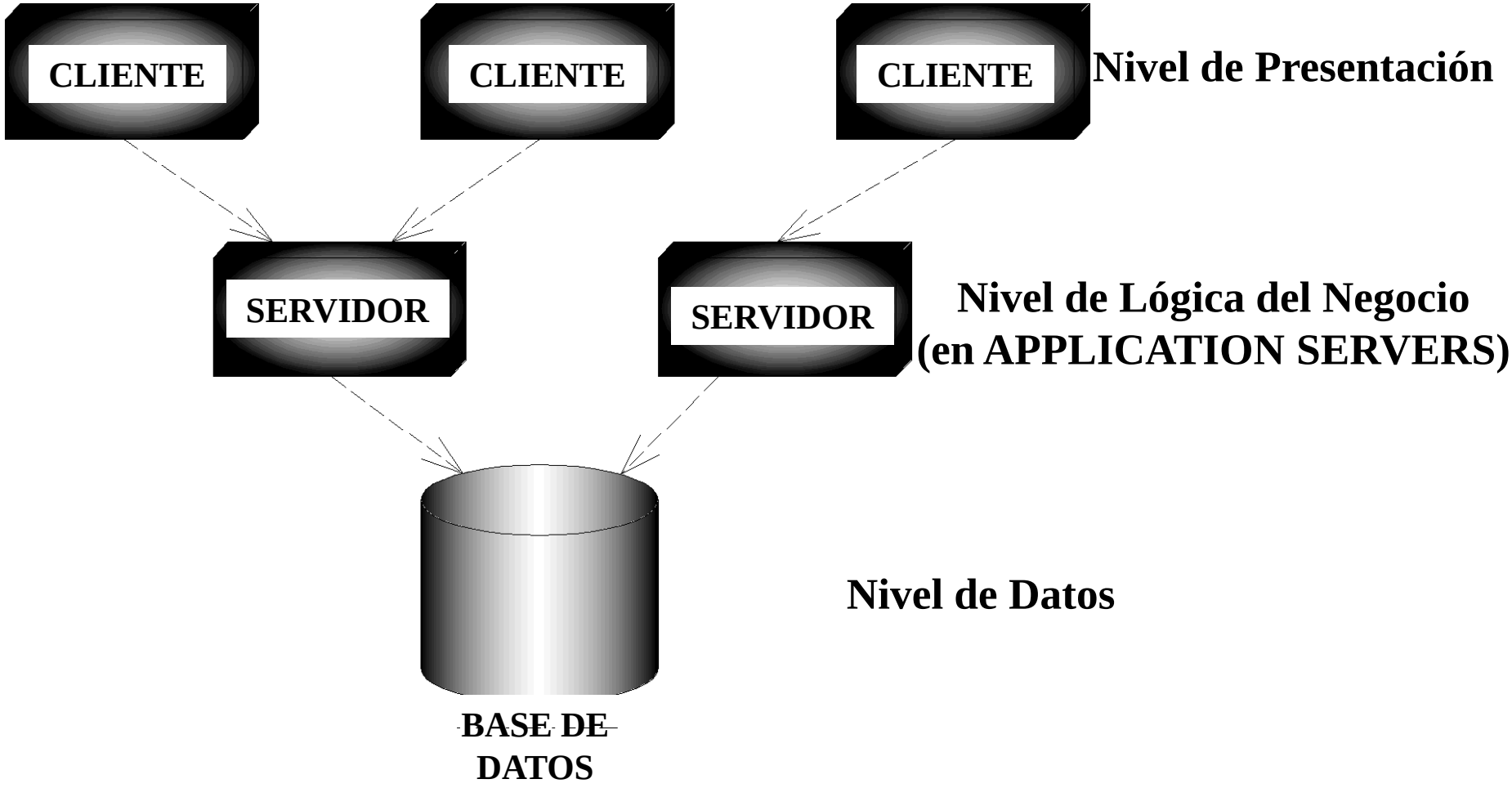

Sentencia SQL
Respuesta
BD
BILLETES : Tabla
NUMERO
ESTADO
NOMBRE
0 OCUPADO Kepa Lasa
1 LIBRE
2 LIBRE
SERVIDOR
river
dbcDriver
OdbcDriver
.JdbcOdbcDriver
bc.JdbcOdbcDriver
c.odbc.JdbcOdbcDriver
.jdbc.odbc.JdbcOdbcDriver
sun.jdbc.odbc.JdbcOdbcDriver
sun.jdbc.odbc.JdbcOdbcDriver
sun.jdbc.odbc.JdbcOdbcDriver
sun.jdbc.odbc.JdbcOdbcDriver
CLIENTE
¡¡ CADA CLIENTE UNA CONEXIÓN ABIERTA CON LA BD!!


Petición Db4o
Resultado
BD
Ikuskizun
Bilete
+zenb
+egoera
+izena
SERVIDOR
CLIENTE
com.db4o.DB4oManager
com.db4o.DB4oManager
com.db4o.DB4oManager
com.db4o.DB4oManager
com.db4o.DB4oManager
¡¡ CADA CLIENTE UNA CONEXIÓN ABIERTA CON LA BD!!

Arquitectura física en 2 niveles: cliente gordo/servidor flaco

- El despliegue de la aplicación es alto: instalar drivers y configurar todos los clientes
- Cambiar de SGBD requiere reinstalar todos los clientes
- Cambiar el esquema de la BD puede afectar a los clientes
- Cambiar la lógica del negocio implica recompilar y desplegar en todos los clientes
- Costos de conexión con la BD son altos. Cada cliente una conexión.
- La red se puede sobrecargar. Cada sentencia SQL usa la red.

Arquitectura física en 3 niveles



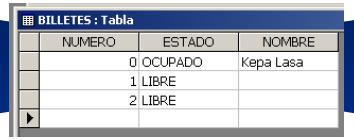
CLIENTE

```
public class PedirBillete extends JFrame {
GestorBilletes gestorBilletes;
void jButton1_actionPerformed(ActionEvent e) {
   int res = gestorBilletes.getBillete(jTextField1.getText()).getNum();
    if (res<0) jTextArea1.append("Error al asignar billete");
    else jTextArea1.append("Asignado. \nReferencia: "+res+"\n");} }
```

Presentación

BD

Datos



SERVIDOR DATOS

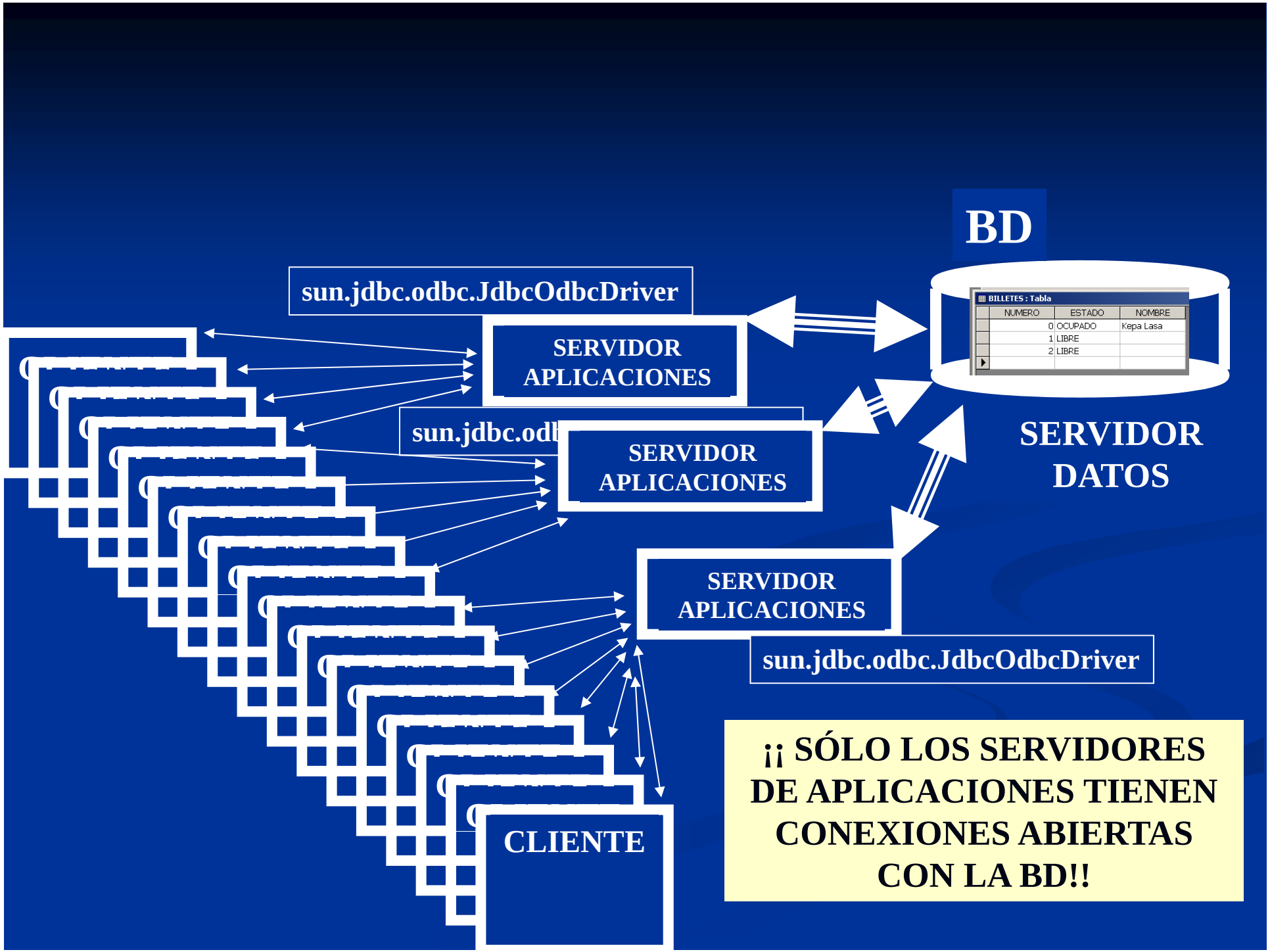
SERVIDOR APLICACIONES

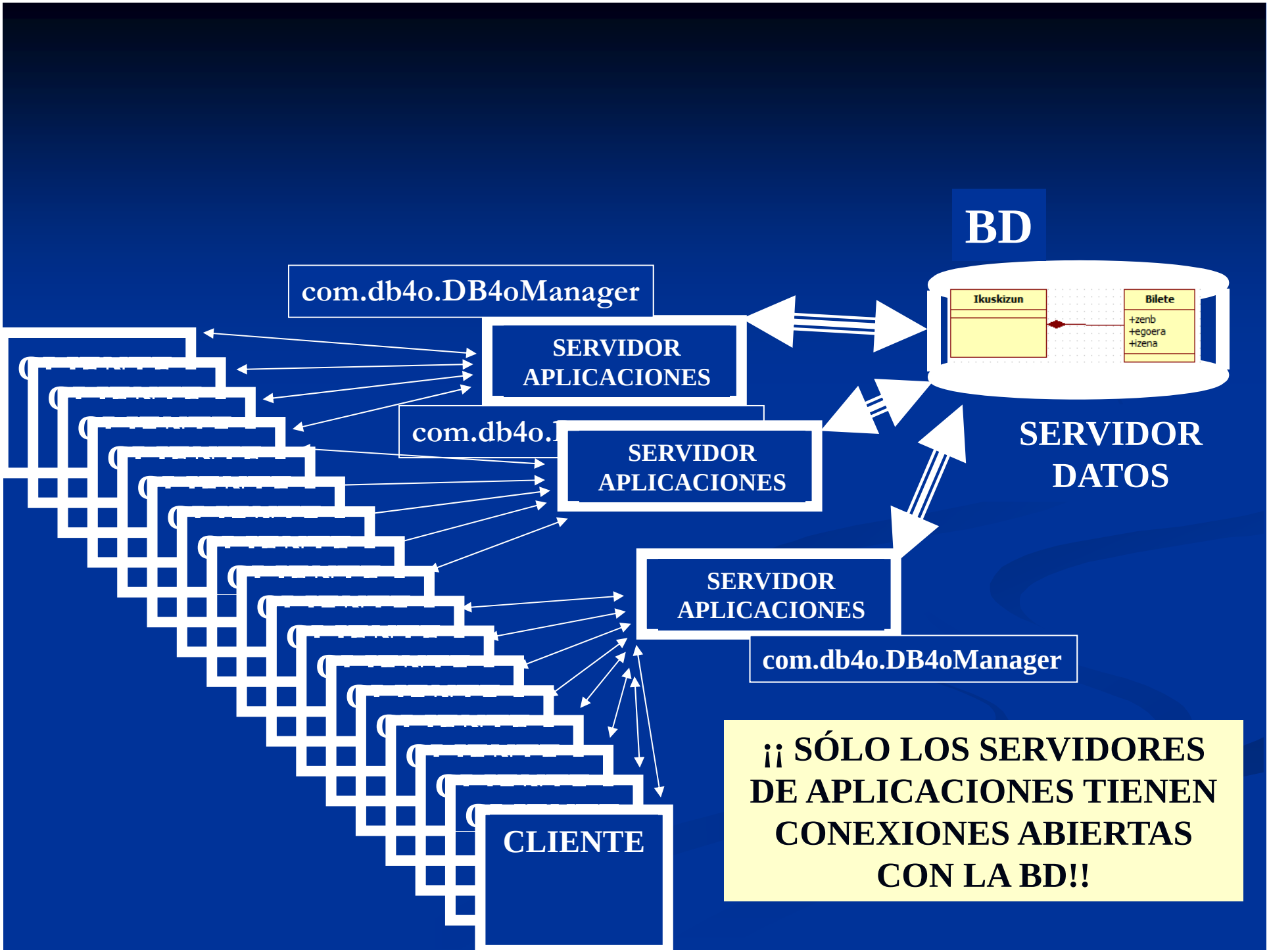
INSTALAR LA CLASE sun.jdbc.odbc.JdbcOdbcDriver

Y DEFINIR FUENTE DATOS ODBC "Billetes"

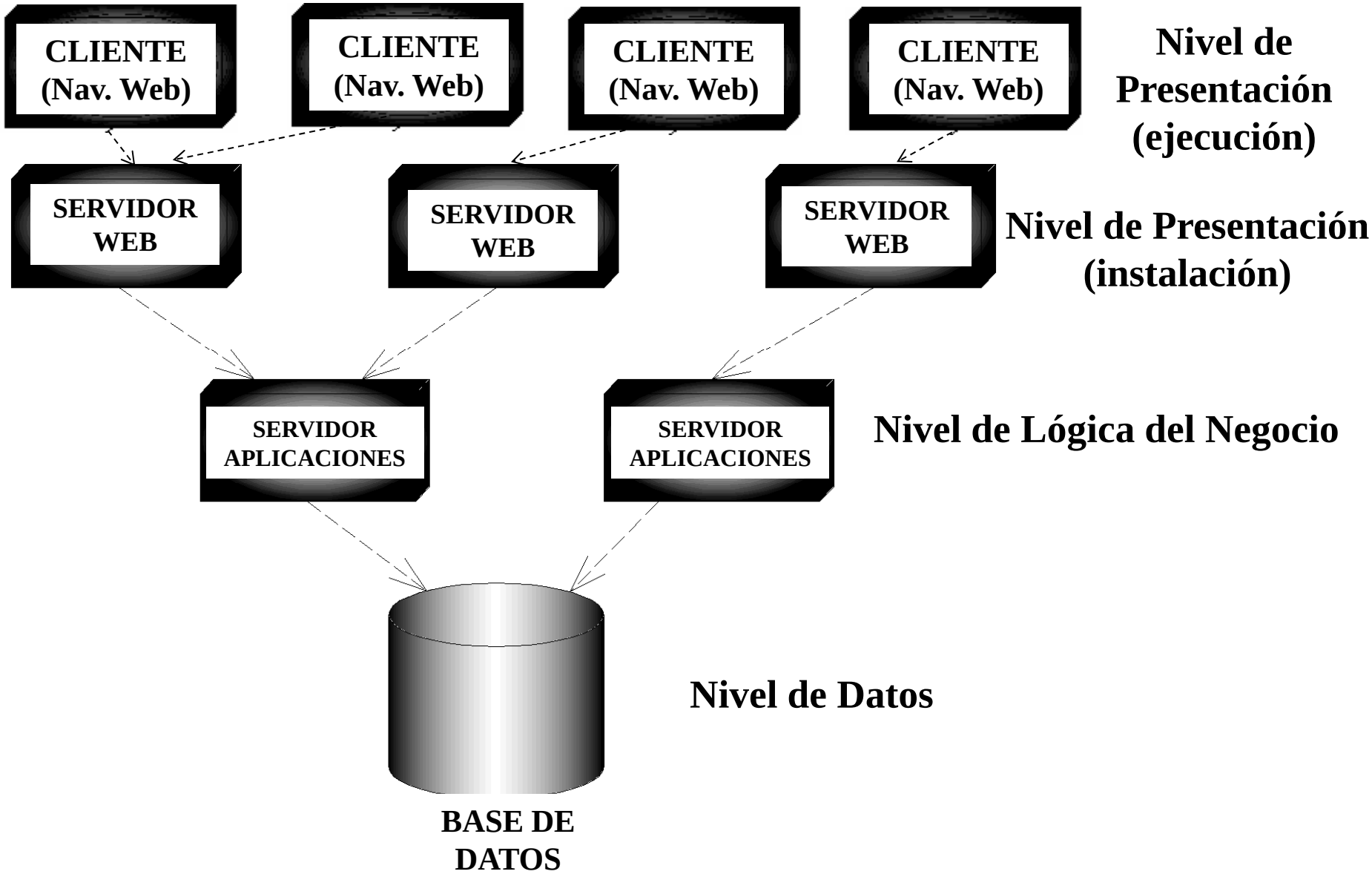
Lógica del Negocio

```
public class ServidorGestorBilletesBD
      implements GestorBilletes
{ public ServidorGestorBilletesBD() {
   Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   conexion=DriverManager.getConnection("jdbc:odbc:Billetes");}
   public Billete getBillete(String nom)
    {ResultSet rs = sent.executeQuery("SELECT NUMERO…";);
      int act = sent.executeUpdate("UPDATE BILLETES …“);
      if (act>0) return new Billete(n,nom); // Núm. billete asignado
     else return new Billete(-1,"");; } // No había ninguno libre}}}
```


BD
BILLETES : Tabla
NUMERO
ESTADO
NOMBRE
0
OCUPADO
Kepa Lasa
1
LIBRE
2
LIBRE
SERVIDOR DATOS
sun.jdbc.odbc.JdbcOdbcDriver
SERVIDOR APLICACIONES
sun.jdbc.odb
SERVIDOR APLICACIONES
SERVIDOR APLICACIONES
sun.jdbc.odbc.JdbcOdbcDriver
CLIENTE
¡¡ SÓLO LOS SERVIDORES DE APLICACIONES TIENEN CONEXIONES ABIERTAS CON LA BD!!


BD
com.db4o.DB4oManager
SERVIDOR APLICACIONES
com.db4o.
SERVIDOR APLICACIONES
SERVIDOR APLICACIONES
com.db4o.DB4oManager
CLIENTE
Ikuskizun
Bilete
+zenb
+egoera
+izena
SERVIDOR DATOS
¡¡ SÓLO LOS SERVIDORES DE APLICACIONES TIENEN CONEXIONES ABIERTAS CON LA BD!!

Las aplicaciones Web permiten más niveles



CLIENTE

Navegador Web

Ejecutar Presentación

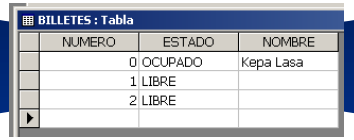
SERVIDOR WEB

HTML + JSP + Bean

Instalar Presentación

BD

Datos



SERVIDOR DATOS

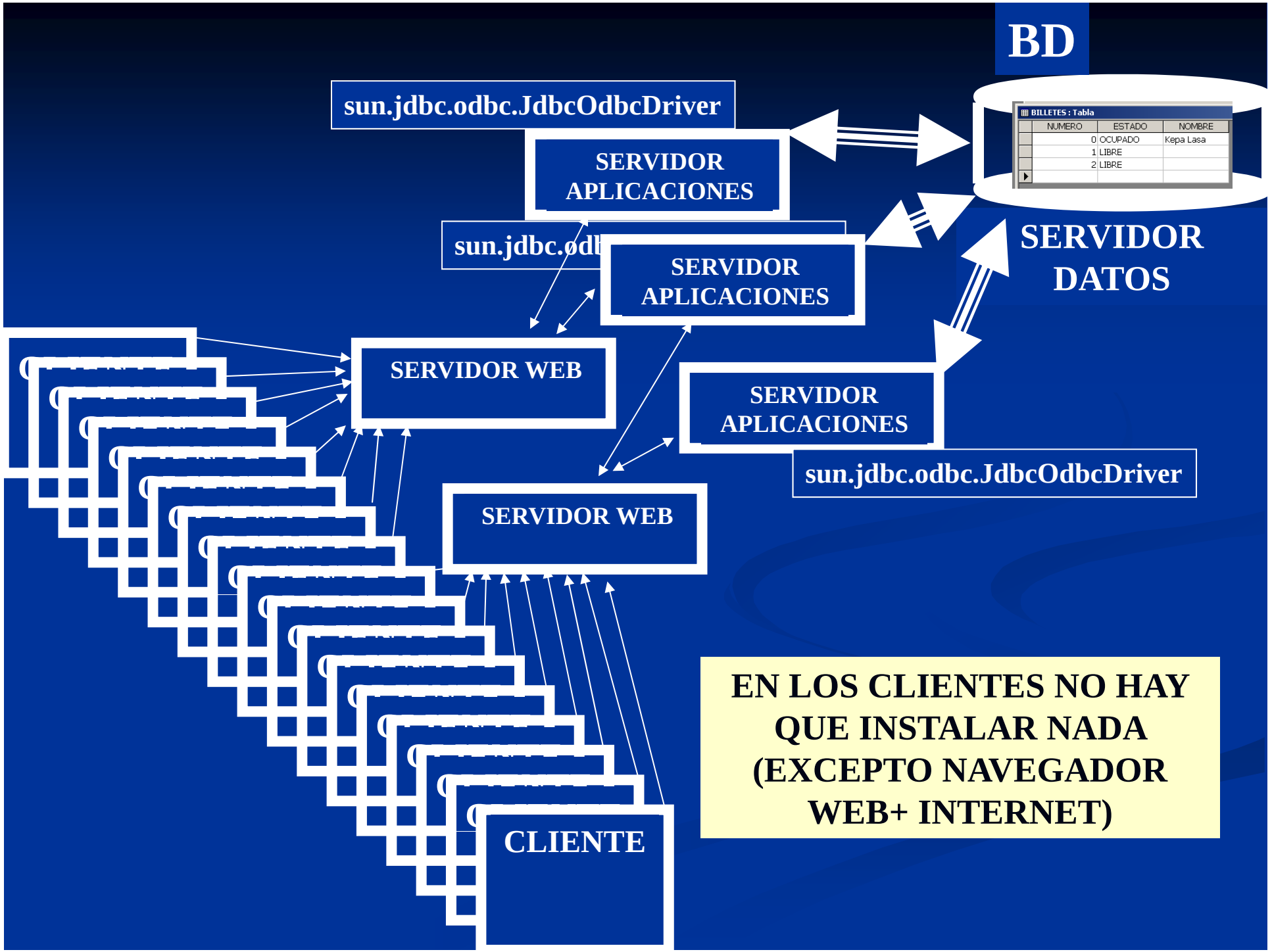
SERVIDOR APLICACIONES

INSTALAR LA CLASE sun.jdbc.odbc.JdbcOdbcDriver

Y DEFINIR FUENTE DATOS ODBC "Billetes"

Lógica del Negocio

```
public class ServidorGestorBilletesBD
      implements GestorBilletes
{ public ServidorGestorBilletesBD() {
   Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   conexion=DriverManager.getConnection("jdbc:odbc:Billetes");}
   public Billete getBillete(String nom)
    {ResultSet rs = sent.executeQuery("SELECT NUMERO…";);
      int act = sent.executeUpdate("UPDATE BILLETES …“);
      if (act>0) return new Billete(n,nom); // Núm. billete asignado
     else return new Billete(-1,"");; } // No había ninguno libre}}}
```


BD
BILLETES : Tabla
NUMERO
ESTADO
NOMBRE
0
OCUPADO
Kepa Lasa
1
LIBRE
2
LIBRE
sun.jdbc.odbc.JdbcOdbcDriver
SERVIDOR APLICACIONES
sun.jdbc.odb
SERVIDOR APLICACIONES
SERVIDOR DATOS
SERVIDOR WEB
SERVIDOR APLICACIONES
sun.jdbc.odbc.JdbcOdbcDriver
SERVIDOR WEB
CLIENTE
EN LOS CLIENTES NO HAY QUE INSTALAR NADA (EXCEPTO NAVEGADOR WEB+ INTERNET)

Arquitectura física en 3 niveles

- Sólo hay que instalar los drivers de la BD en los nodos donde se encuentre la lógica del negocio (nodos servidores)
- Cambiar de SGBD/esquema de la BD NO requiere reinstalar todos los clientes. Sólo los de la lógica del negocio.
- Cambiar la lógica del negocio NO implica recompilar y desplegar en todos los clientes.
- Costos de conexión con la BD NO son tan altos. Los clientes no realizan conexiones con la BD. Sólo los servidores con la lógica del negocio lo hacen.

En general, se MEJORA en EFICIENCIA, MANTENIMIENTO y EXTENSIBILIDAD

Arquitectura física en 3 niveles

- Existe tecnología que permite construir aplicaciones siguiendo esta filosofía de componentes y objetos distribuidos **(server-side components)**
  - Entreprise JavaBeans (EJBs) es la arquitectura de componentes para la plataforma Java 2 Enterprise Edition (J2EE). Definido por Sun Microsystems
    - El nivel de presentación se puede dividir más usando Java Applets, Servlets y/o JSPs
  - CORBA es una arquitectura para comunicación entre objetos distribuidos a través de ORBs (Object Request Brokers). Es un estándar definido por OMG.
  - DCOM/COM+ y la plataforma .NET es la tecnología equivalente desarrollada por Microsoft

Arquitectura física en 3 niveles

- Pero también se puede conseguir con tecnología "sencilla" de Java:
  - Construcción de interfaces gráficas en Java (AWT y SWING) para definir el NIVEL DE PRESENTACIÓN
  - Ejecución del NIVEL DE PRESENTACIÓN EN UN navegador WEB (Págs. con Applets o Págs. JSPs)
  - Computación con objetos distribuidos (RMI) para definir el NIVEL LÓGICA DEL NEGOCIO e invocarlo desde el NIVEL DE PRESENTACIÓN
  - Llamadas a JSPs y de ellos a JavaBeans
  - Conexión con BDs relacionales (JDBC) o con BDOOs (db4o) para conseguir la comunicación entre NIVEL LÓGICA DEL NEGOCIO y DATOS