UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA

P3
Diseño, implementación y ejecución

Mª Pilar Moreno Duque
Julio Alberto López Gómez
Raúl García-Hidalgo Tajuelo
Raquel Lorente Almansa

Asignatura: Ingeniería del Software I

Grupo de Titulación (20/21): 20

Titulación: Grado en Ingeniería Informática

Fecha: 18/12/2012

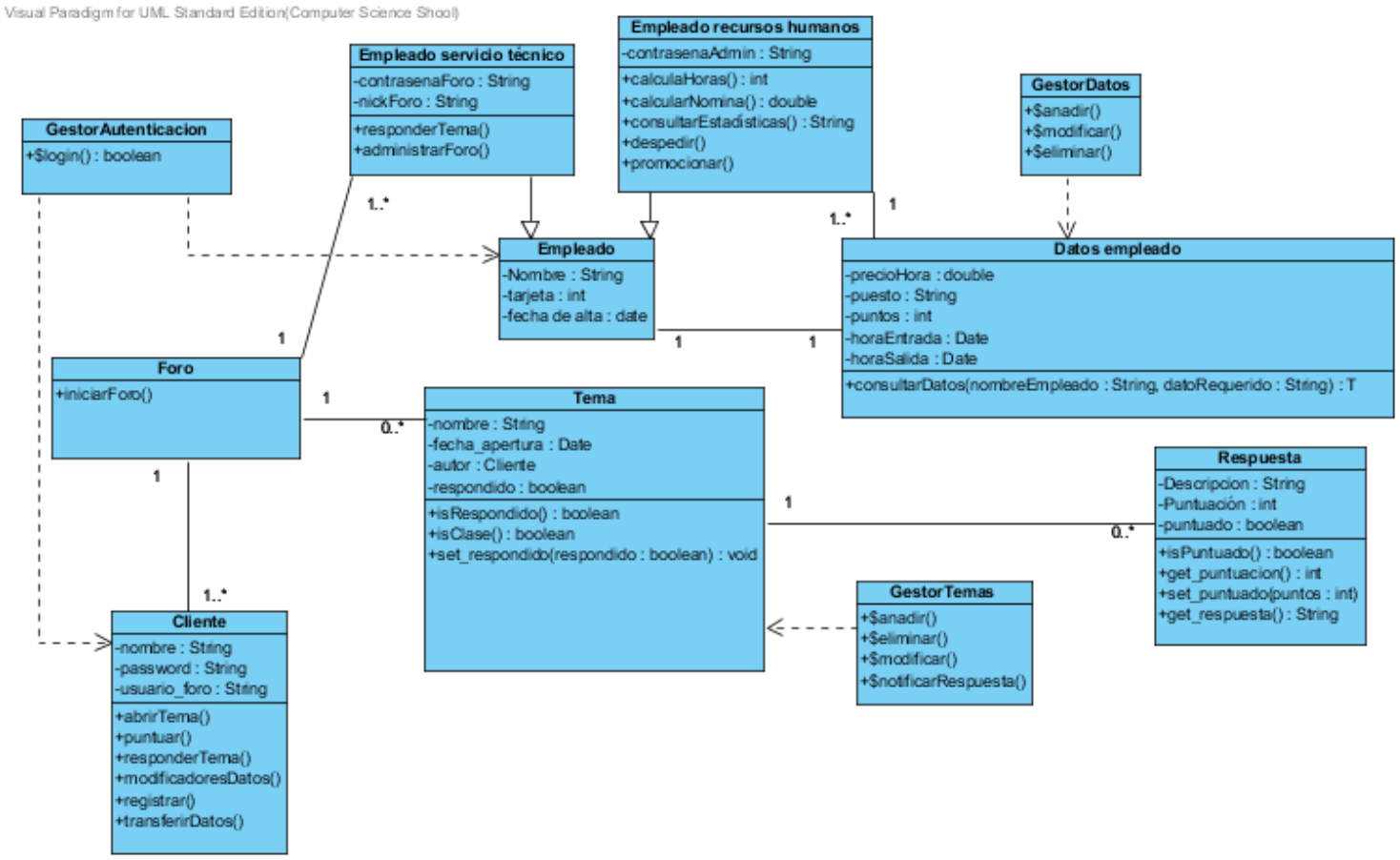
Ficha del Trabajo:

Código:		Fecha:	21/11/2012
Título:	Diseño, implementación y ejecución		

Equipo	**Nº:**	
Apellidos y Nombre	Firma	Puntos
Mª del Pilar Moreno Duque		25 %
Julio Alberto López Gómez		25%
Raúl García-Hidalgo Tajuelo		25%
Raquel Lorente Almansa		25%

Contenido

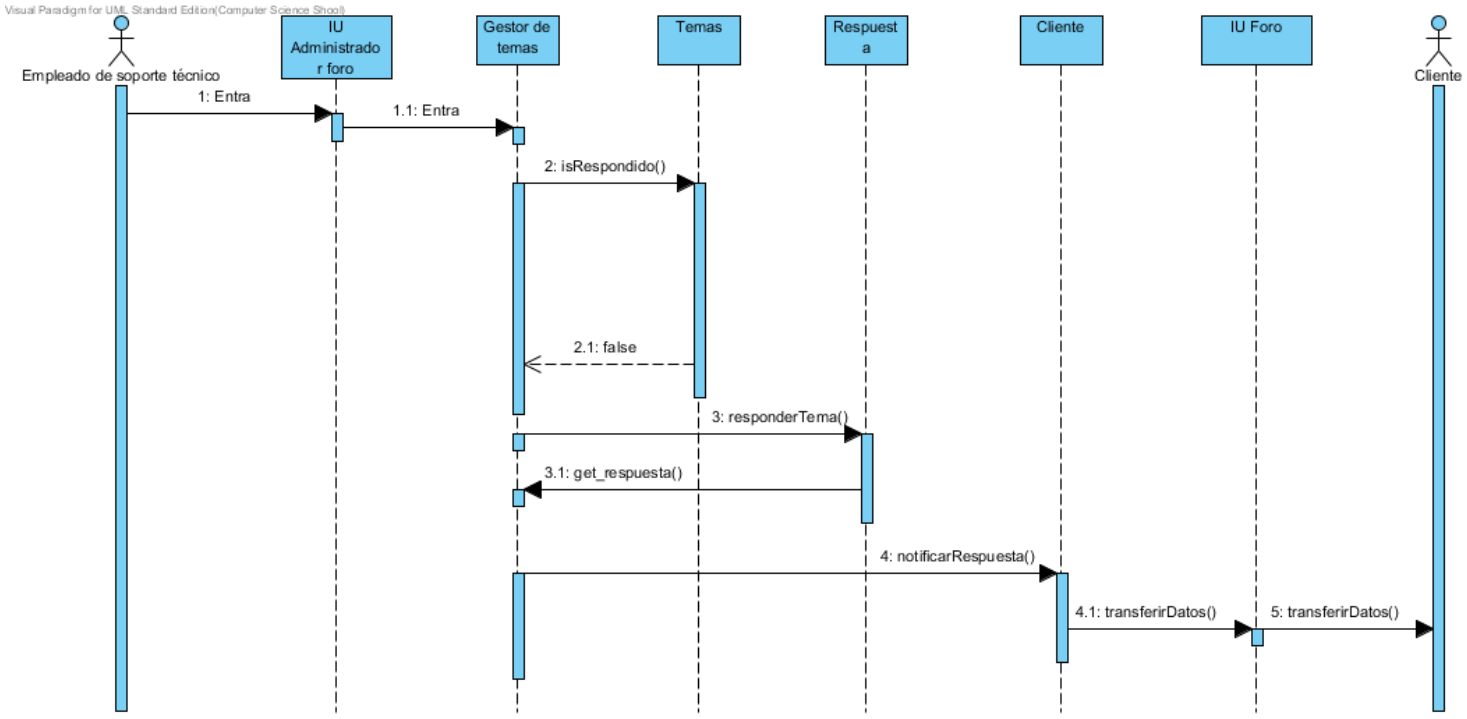
1. Diagramas de clases de diseño



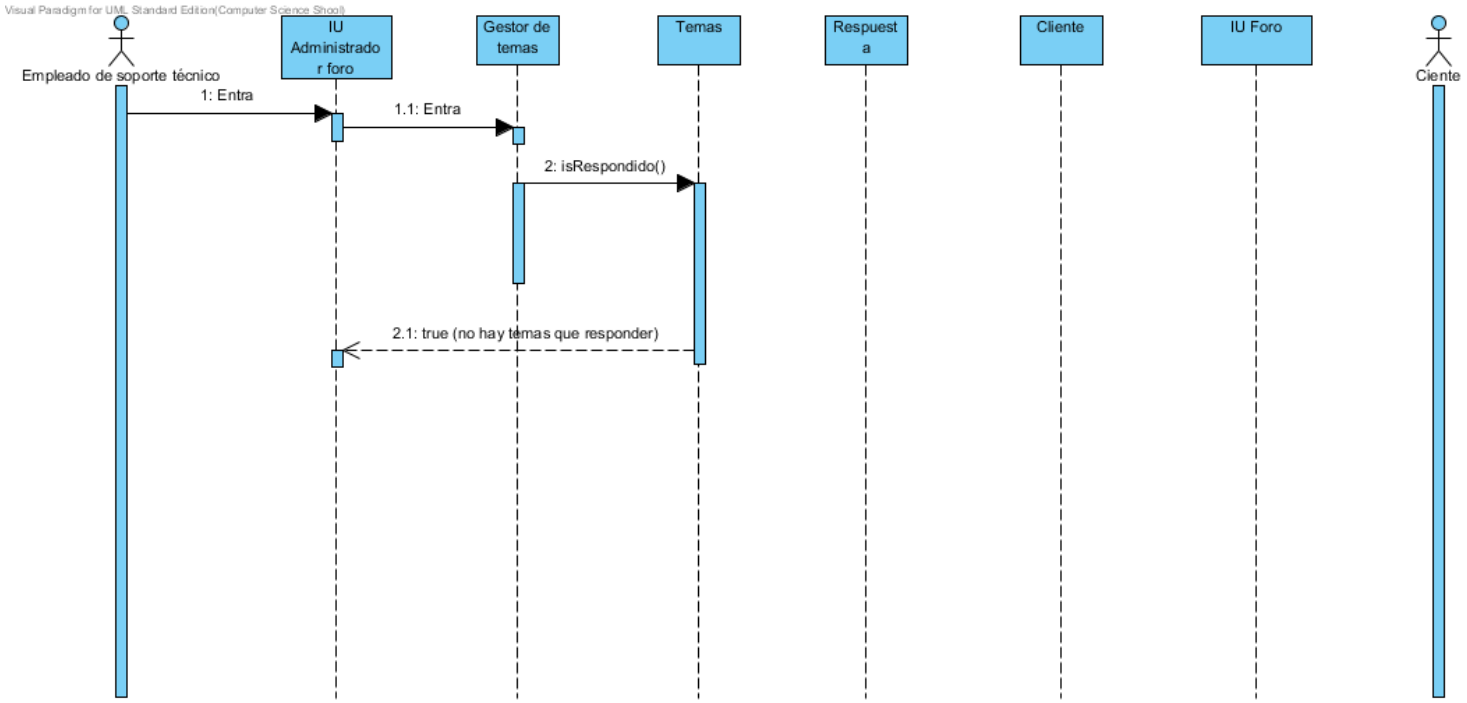
2. Diagramas de secuencia de diseño

a) Responder tema:

-Escenario normal:

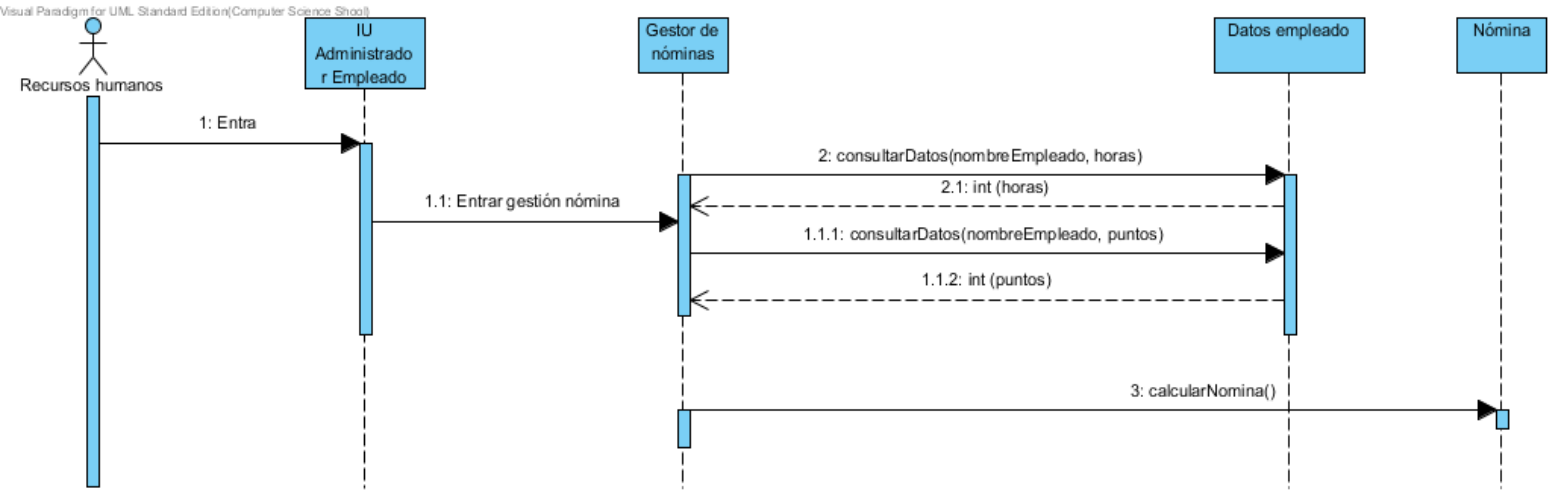


-Escenario alternativo:

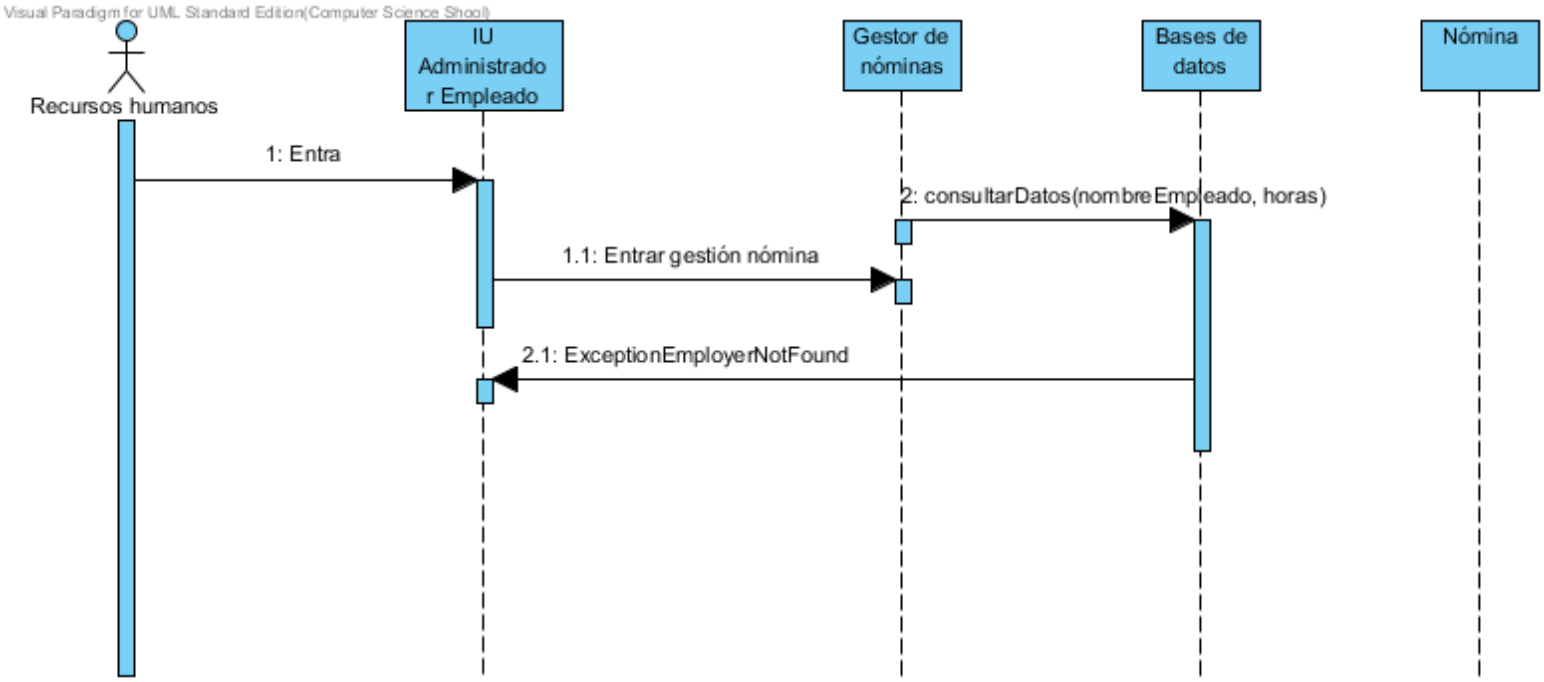


b) Calcular nómina:

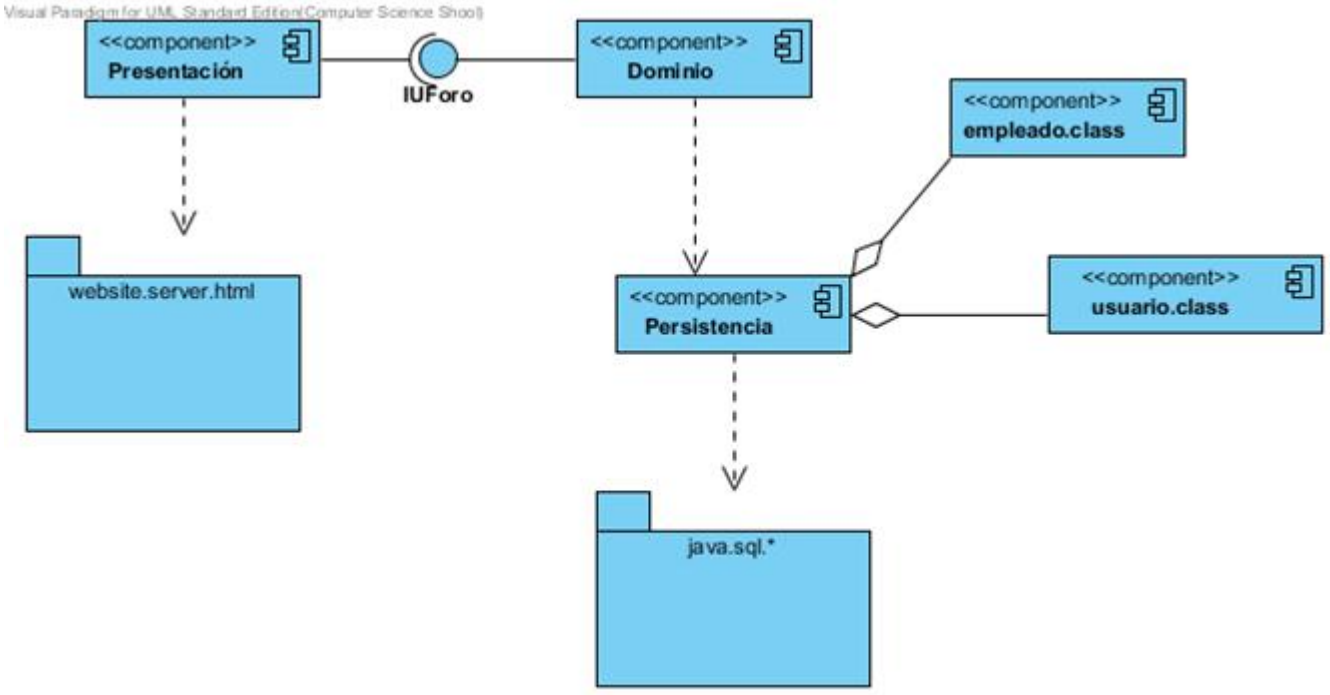
-Escenario normal:



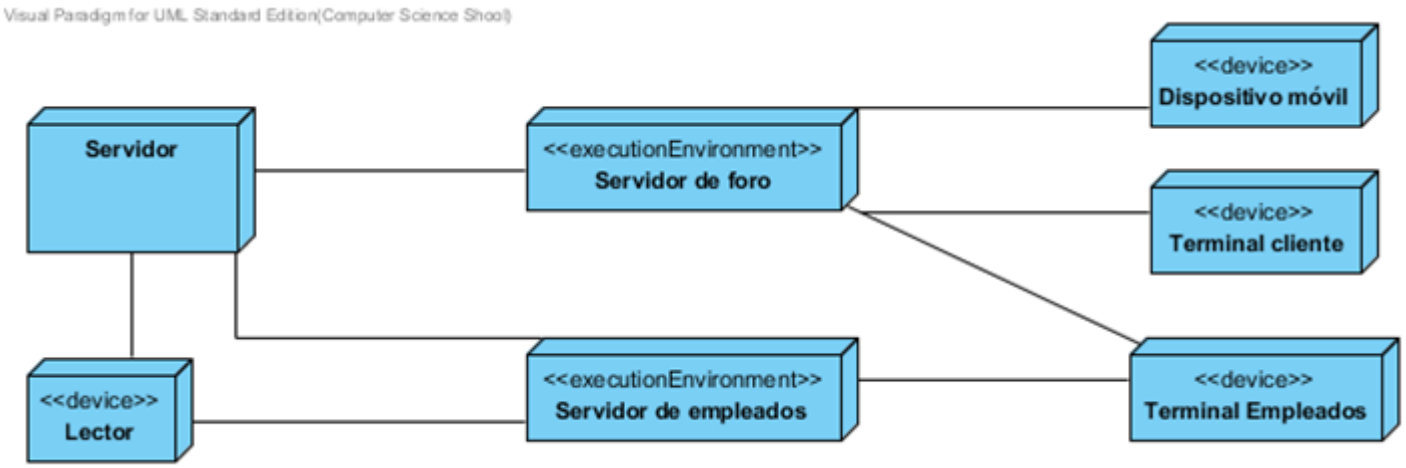
-Escenario alternativo:



3. Diagrama de componentes



4. Diagrama de despliegue



5. Implementación de un caso de uso

Hemos decidido implementar el caso de uso Autenticarse, que hemos considerado suficientemente representativo del sistema. Lo hemos elegido así, porque muchos de los casos de uso que hemos incluido en nuestro diagrama requieren primero la autenticación del usuario en el sistema. Para ello, hemos utilizado los contenidos vistos en clase sobre bases de datos y JDBC.

Este es el patrón Agente:

```
import java.sql.*;
import java.util.*;
import java.util.Vector;
public class Agente {
    protected static Agente mInstancia=null;
    protected static Connection mBD;
    //Constructor
    private Agente()throws Exception {
        Agente.conectar();
    }

     public static Agente getAgente() throws Exception{
        if (mInstancia==null){
            mInstancia=new Agente();
        }
        return mInstancia;
     }
     public boolean login() {
         boolean in=false;
```

```
        String nombreUsuario, contrasena;
        Scanner sc = new Scanner(System.in);
        sc.useLocale(Locale.US);
        System.out.println("Introduzca el nombre de
usuario:");
        nombreUsuario = sc.next();
        System.out.println("Introduzca la contraseña");
        contrasena = sc.next();
        int numColumnas = 0;

        try{
                Agente.conectar();
                PreparedStatement sentence =
mBD.prepareStatement("SELECT login, passw FROM Cliente");
                ResultSet respuesta = sentence.executeQuery();



                while(respuesta.next()){
                        if (nombreUsuario.equals(
respuesta.getString("login"))){
                                if
(contrasena.equals(respuesta.getString("passw"))){
                                        in = true;
                                }
                        }
                }
                Agente.desconectar();
                sc.close();
                return in;
        }catch (Exception e){
                System.out.println("Fallo de autenticación");
        }
        throw new UnsupportedOperationException();
 }



    public static void conectar() throws Exception {
        String url="jdbc:odbc:nombreODBC";
        String driver="sun.jdbc.odbc.JdbcOdbcDriver";
        Class.forName(driver);
        mBD=DriverManager.getConnection(url, "login",
"password");
    }

    public static void desconectar() throws
Exception{mBD.close();}

    public int insert(String SQL) throws SQLException,
Exception{
       PreparedStatement sentencia =
mBD.prepareStatement(SQL);
```

```
        return sentencia.executeUpdate();
    }

    public Vector select(String SQL) throws
SQLException,Exception{
        int numColumnas = 0;
        Vector resultado = new Vector();
        Vector registro = new Vector();
        //Creamos la consulta y la lanzamos contra la BD
        PreparedStatement sentencia =
mBD.prepareStatement(SQL);
        ResultSet respuesta = sentencia.executeQuery();
        //Obtenemos metainformaci—n del resultado de la
consulta
        ResultSetMetaData rsmd = respuesta.getMetaData();
        numColumnas = rsmd.getColumnCount();
        //Al ejecutar "next()", el objeto ResultSet apunta al
siguiente
        //registro, y la primera vez, apunta al primer registro
de todos.
        while(respuesta.next()){
             //Por cada registro inicializamos el vector
registro
             registro = new Vector();
             //Dentro del While, vamos a procesar uno a uno los
registros
             //Muy importante, empezamos a contar desde la
posici—n 1, si buscamos
             //en la posici—n 0 de los registros del ResultSet
dar‡ un error
             for(int i=1; i<=numColumnas; i++){
                  //En este switch controlamos el tipo de cada
uno de las columnas del
                  //registro, y segœn el tipo, utilizamos un
mŽtodo y otro de la clase
                  //ResultSet para extraer el dato.
                  //Para simplificar, podemos poner s—lo los
tipos de dato que tenemos
                  //en la BD de ejemplo de la pr‡ctica
                  switch(rsmd.getColumnType(i)){
                           case Types.INTEGER:
                               registro.add(new
Integer(respuesta.getInt(i)));
                           break;
                           case Types.VARCHAR:

registro.add(respuesta.getString(i));
                           break;
                           case Types.DOUBLE:
                               registro.add(new
Double(respuesta.getDouble(i)));
                           break;
                  }
```

```
            }
            //Al terminar de procesar cada registro, lo
almacenamos en el vector resultado
            resultado.add(registro);
        }
        //Al terminar, ya hemos procesado todos los registros
del ResultSet, los tenemos
        //en el vector, y podemos cerrar todos los punteros que
hay hacia la BD y
        //desconectar de la BD
        respuesta.close();
        sentencia.close();
        desconectar();

        return resultado;
    }

    public int delete(String SQL) throws
SQLException,Exception{
        PreparedStatement sentencia =
mBD.prepareStatement(SQL);
        return sentencia.executeUpdate();
    }

    public int update(String SQL) throws
SQLException,Exception{
        PreparedStatement sentencia =
mBD.prepareStatement(SQL);
        return sentencia.executeUpdate();
    }
}
```

Y esta es la prueba Agente:

```
import java.sql.SQLException;
import java.util.Vector;


public class PruebaAgente {
    public static void main(String[] args){


        String SQL_insert_1 = "INSERT INTO Cliente VALUES
('jgh13','pass12');";
        String SQL_insert_2 = "INSERT INTO Cliente VALUES
('reshu23','contrasena');";
        String SQL_insert_3 = "INSERT INTO Cliente VALUES
('olakase','111111t');";

        try{
           Agente.getAgente().delete("DELETE * FROM Cliente");
            Agente.getAgente().insert(SQL_insert_1);
           Agente.getAgente().insert(SQL_insert_2);
```

```
            Agente.getAgente().insert(SQL_insert_3);
           //int intentos=0;
          // while(intentos<3){
                boolean log=Agente.getAgente().login();

              if(log){
                 System.out.println("Acceso concedido");
              }
              else{
                 System.out.println("Acceso denegado");

                 //intentos++;
                 //System.out.println("Quedan"+(3-
intentos)+"intentos");
               //}
           }

           Agente.desconectar();
         }
         catch(SQLException sqle){
             //Gesti—n de excepciones de la base de datos
             System.out.println("excepcion sqle");
             return;
         }
         catch(Exception e){
             //Gesti—n de excepciones
             System.out.println("No se pudo cargar el puente
JDBC-ODBC");
             return;
         }
     }
}
```