Acceso a Datos – Tarea 06

DESARROLLO DE APLICACIONES MULTIPLATAFORMA

FELIPE RODRIGUEZ

Contenido

Nota del Alumno sobre la tarea:

Se han realizado diversos cambios y mejoras en la estructura del ejercicio de ejemplo del temario del que se ha partido como base para la tarea, siendo principalmente los que continúan, que serán mostrados en detalle más adelante:

- *Se han realizado diversas mejoras a la base de datos del ejemplo, que se adjunta a la presente tarea, aunque el ejercicio es completamente funcional y compatible con la tabla de calificaciones normal.*
- *No es necesario copiar el .jar, dado que en el apartado 2 donde hacemos uso del componente, hace llamada a la ruta relativa a su ubicación, por lo que sólo tendremos que hacer clean and build cada vez que hagamos un cambio y encontrarse los dos proyectos en la misma carpeta.*
- *Este ejercicio ha sido realizado con una base de datos ubicada en red local, por lo que se ha modificado el ComponenteBean con una serie de atributos finales que pueden ser modificados con la configuración del servidor local/red así como su usuario y pass correspondiente, no teniendo que ser modificado posteriormente en el resto de métodos.*
- *Se han realizado algunos cambios en el funcionamiento de ciertos métodos, dado que no funcionaban correctamente. En el método seleccionaDNI(), al introducir un DNI, entraba en bucle infinito y no terminaba la ejecución. En el método recargarFilas(), se ha implementado limpiar el vector al recargar, dado que al no ser limpiado el vector, se añadían a los existentes la nueva lista de filas cada vez que se procedia a la recarga.*
- *Se ha añadido el método getTamanio(), que devuelve el tamaño en ese instante del vector de calificaciones, dado que en la estructura del ejemplo, solo permite listar si conocemos el número de filas que posee el vector. De esta manera, siempre se listarán todas sin tener que conocer el número de filas.*

Ejercicio 1

Añadir tabla de calificaciones

Se crea la el código SQL de la base de datos, ampliándose la existente del ejemplo:

alumnos.sql

```sql
--
-- Base de datos: `alumnos`
--
-- --------------------------------------------------------
--
-- Eliminamos base de datos existentes previas
--
DROP DATABASE IF EXISTS `alumnos`;
--
-- Creamos la base de datos
--
CREATE DATABASE IF NOT EXISTS `alumnos`;
--
-- Usamos la base de datos alumnos
--
USE `alumnos`;
--
-- Estructura de tabla para la tabla `alumnos`
--
CREATE TABLE IF NOT EXISTS `alumnos` (
  `DNI` varchar(9) NOT NULL,
  `Nombre` varchar(50) NOT NULL,
  `Apellidos` varchar(70) NOT NULL,
  `Direccion` varchar(100) NOT NULL,
  `FechaNac` date NOT NULL,
  PRIMARY KEY (`DNI`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
--
-- Volcar la base de datos para la tabla `alumnos`
--
INSERT INTO `alumnos` (`DNI`, `Nombre`, `Apellidos`, `Direccion`, `FechaNac`) VALUES
('12345678A', 'José Alberto', 'González Pérez', 'C/Albahaca, nº14, 1ºD', '1986-07-15'),
('23456789B', 'Almudena', 'Cantero Verdemar', 'Avd/ Profesor Alvarado, n27, 8ºA', '1988-11-04'),
('14785236D', 'Martín', 'Díaz Jiménez', 'C/Luis de Gongora, nº2.', '1987-03-09'),
('96385274F', 'Lucas', 'Buendia Portes', 'C/Pintor Sorolla, nº 16, 4ºB', '1988-07-10');
--
-- Estuctura de tabla para la tabla `calificaciones`
--
CREATE TABLE IF NOT EXISTS `calificaciones` (
  `id_curso`int NOT NULL AUTO_INCREMENT,
  `DNI` varchar(9) NOT NULL,
  `NombreCurso` varchar(60) NOT NULL,
  `Curso` varchar(5) NOT NULL,
  `Nota` double NOT NULL,
  PRIMARY KEY (`id_curso`),
  FOREIGN KEY (`DNI`) REFERENCES `alumnos`(`DNI`) ON DELETE CASCADE
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
--
-- Volcar la base de datos para la tabla `calificaciones`
--
INSERT INTO `calificaciones` (`DNI`, `NombreCurso`, `Curso`, `Nota`) VALUES
('12345678A', 'Acceso a datos', '17-18', 7),
('23456789B', 'Desarrollo Web Entorno Servidor', '16-17', 9.2),
('14785236D', 'Bases de Datos', '15-16', 8.5),
('96385274F', 'Desarrollo Web Entorno Cliente', '17-18', 5.3);
```

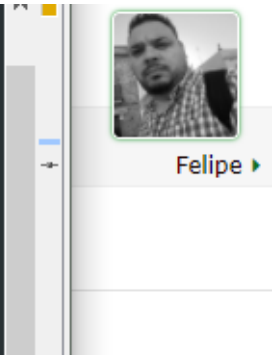
Observaciones:

- Se ha incluido que al importar el script SQL, elimine previamente alguna versión existente de la base de datos para que sea sustituida por esta misma. No es necesario, dado que en principio la tarea esta realizada con la tabla normal y corriente.
- Hay que modificar en el componente la url, user y Password de conexión a la base de datos previamente si queremos que funcione.
- El campo DNI de *calificaciones* será clave foránea del campo DNI de la tabla *alumnos* añadiendo integridad referencial.
- Para la consecución de lo anterior, se ha creado el campo id_curso como clave principal para poder usar DNI como clave foránea.

Creando el componente

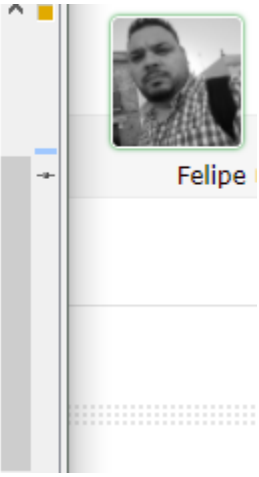
Atributos

```
/**
 * Clase CalificacionesBean
 *
 * @author Felipon
 */
public class CalificacionesBean implements Serializable {
    //ATRIBUTOS
    protected String DNI;
    protected String NombreCurso;
    protected String Curso;
    protected Double Nota;
```



Constructor

```
//CONSTRUCTOR DE LA CLASE
public CalificacionesBean() {
    propertySupport = new PropertyChangeSupport(this);

    try {
        recargarFilas();
    } catch (ClassNotFoundException ex) {
        this.DNI = "";
        this.NombreCurso = "";
        this.Curso = "";
        this.Nota = 0.0;
        Logger.getLogger(CalificacionesBean.class.getName()).log(Level.SEV
    }
}
```



Al ser instanciado, llamará al método recargarFilas(); que volcará en un vector los valores de la tabla.

Métodos getter y setter

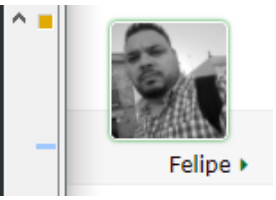
```java
/* GETTERS Y SETTERS */
/** Getter que devuelve DNI ...5 lines */
public String getDNI() {
    return DNI;
}
/** Setter que establece DNI ...5 lines */
public void setDNI(String DNI) {
    this.DNI = DNI;
}
/** Getter que devuelve NombreCurso ...5 lines */
public String getNombreCurso() {
    return NombreCurso;
}
/** Setter que establece NombreCurso ...5 lines */
public void setNombreCurso(String NombreCurso) {
    this.NombreCurso = NombreCurso;
}
/** Getter que devuelve Curso ...5 lines */
public String getCurso() {
    return Curso;
}
/** Setter que establece Curso ...5 lines */
public void setCurso(String Curso) {
    this.Curso = Curso;
}
/** Getter que devuelve Nota ...5 lines */
public Double getNota() {
    return Nota;
}
/** Setter que establece Nota ...5 lines */
public void setNota(Double Nota) {
    this.Nota = Nota;
}
```

Clase auxiliar Calificación

```java
/**
 * Clase auxiliar que usaremos para crear un vector privado de calificaciones
 */
private class Calificacion {
    //Atributos
    public String DNI;
    public String NombreCurso;
    public String Curso;
    public Double Nota;

    //Constructor vacio
    public Calificacion() {  };

    //Constructor con parametros
    public Calificacion(String DNI, String NombreCurso, String Curso, Double Nota) {
        this.DNI = DNI;
        this.NombreCurso = NombreCurso;
        this.Curso = Curso;
        this.Nota = Nota;
    }
}
```

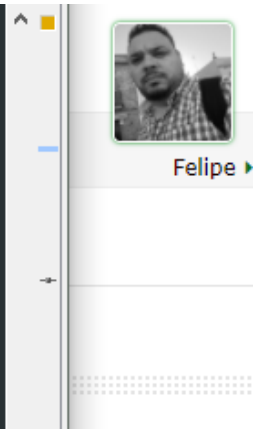
Atributo Calificaciones

```
/**
 * Usaremos un atributo con un vector auxiliar para cargar la información
 * de la tabla de forma que tengamos acceso a los datos sin necesidad de
 * estar conectados constantemente
 */
private Vector Calificaciones = new Vector();
```



Atributo Tamanio y método getTamanio

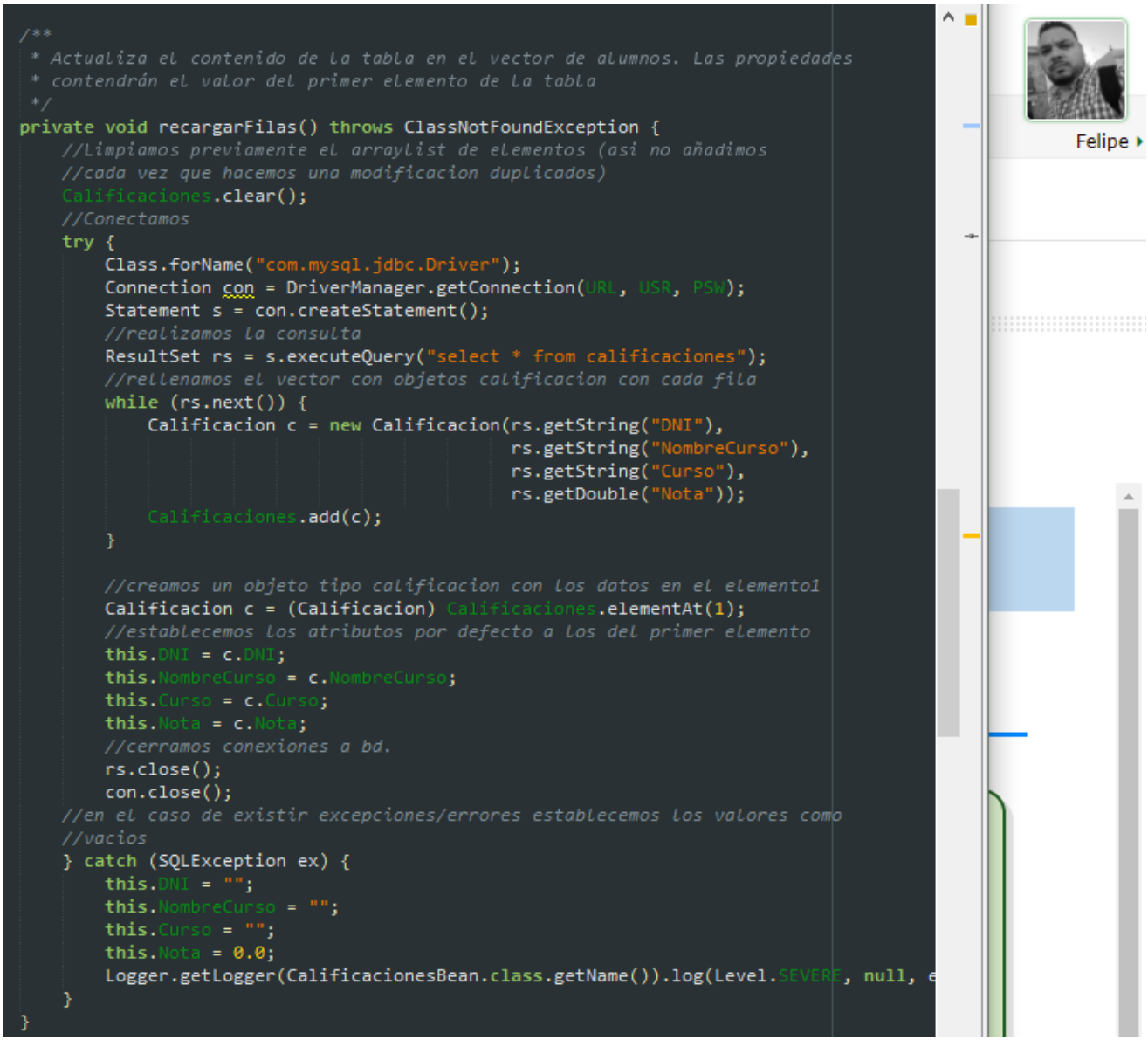
```
//Almacenamos el tamaño del vector
private int tamanio;

/**
 * Método getTamanio(), que devuelve un entero con el tamaño en dicho momento
 * del vector Calificaciones. Útil para los bucles y listar datos.
 *
 * @return tamanio int
 */
public int getTamanio() {
    //actualiza el tamaño actual del vector
    this.tamanio = Calificaciones.size();
    return this.tamanio;
}
```



Este método se ha implementado para recorrer el vector Calificaciones y poder mostrar correctamente todas las calificaciones al proceder a su listado. En el ejemplo de la unidad debía conocerse el número finito de filas existentes del vector.

Ahora es posible listar los cursos sin necesidad de conocer el tamaño del vector interno.

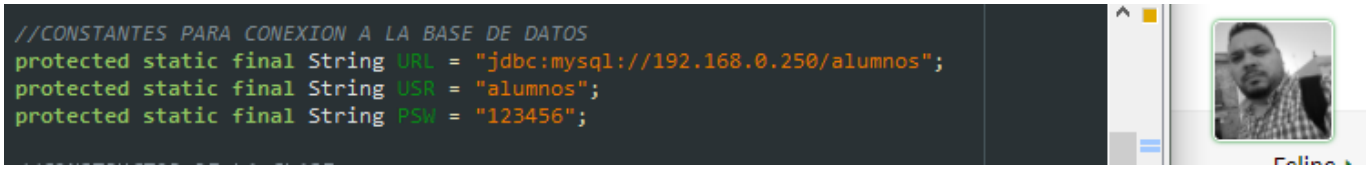
Método recargarFilas



```
/**
 * Actualiza el contenido de la tabla en el vector de alumnos. Las propiedades
 * contendrán el valor del primer elemento de la tabla
 */
private void recargarFilas() throws ClassNotFoundException {
    //Limpiamos previamente el arraylist de elementos (asi no añadimos
    //cada vez que hacemos una modificacion duplicados)
    Calificaciones.clear();
    //Conectamos
    try {
        Class.forName("com.mysql.jdbc.Driver");
        Connection con = DriverManager.getConnection(URL, USR, PSW);
        Statement s = con.createStatement();
        //realizamos la consulta
        ResultSet rs = s.executeQuery("select * from calificaciones");
        //rellenamos el vector con objetos calificacion con cada fila
        while (rs.next()) {
            Calificacion c = new Calificacion(rs.getString("DNI"),
                                              rs.getString("NombreCurso"),
                                              rs.getString("Curso"),
                                              rs.getDouble("Nota"));
            Calificaciones.add(c);
        }

        //creamos un objeto tipo calificacion con los datos en el elemento1
        Calificacion c = (Calificacion) Calificaciones.elementAt(1);
        //establecemos los atributos por defecto a los del primer elemento
        this.DNI = c.DNI;
        this.NombreCurso = c.NombreCurso;
        this.Curso = c.Curso;
        this.Nota = c.Nota;
        //cerramos conexiones a bd.
        rs.close();
        con.close();
    //en el caso de existir excepciones/errores establecemos los valores como
    //vacios
    } catch (SQLException ex) {
        this.DNI = "";
        this.NombreCurso = "";
        this.Curso = "";
        this.Nota = 0.0;
        Logger.getLogger(CalificacionesBean.class.getName()).log(Level.SEVERE, null, e
    }
}
```

Se ha modificado el método para que limpie el vector previamente antes de recargar el contenido de filas en el vector. En el caso de no realizar este paso, se insertarán al final del vector existente, por lo que creará un número irreal con respecto a la tabla, mostrando registros duplicados.

Atributos de conexión a base de datos

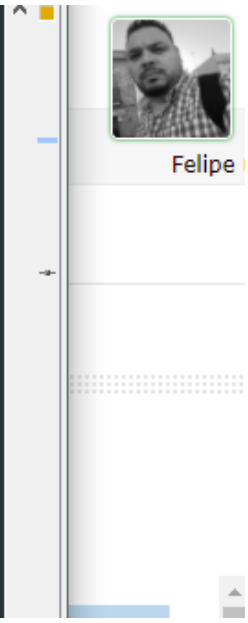


```
//CONSTANTES PARA CONEXION A LA BASE DE DATOS
protected static final String URL = "jdbc:mysql://192.168.0.250/alumnos";
protected static final String USR = "alumnos";
protected static final String PSW = "123456";
```

Se han creado estos atributos al inicio de la clase con el fin de facilitar la configuración de la conexión a la base de datos como atributos constantes (finales). De esta manera se pueden evitar posibles errores. Están modificados para conectar a un servidor de BD en red local con usuario y clave concreta, verificar estos datos previamente antes de ejecutar el proyecto.

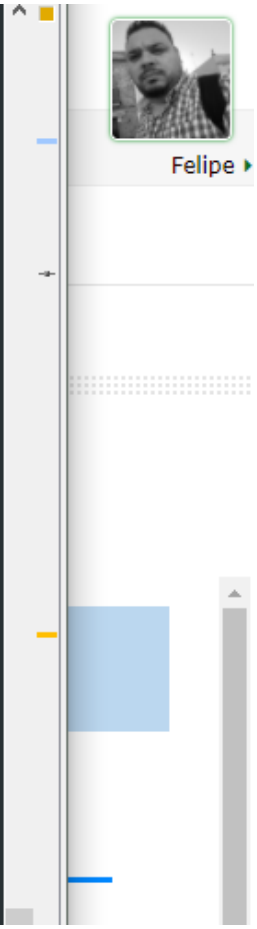
Método seleccionar fila

```java
/**
 * Metodo seleccionarFila al que se le pasa un valor de fila entero por
 * parametro y establece los atributos existentes. En caso de no existir
 * dichos datos o pasar un valor de fila incorrecto, se cargaran valores
 * por defecto.
 *
 * @param i numero de la fila a cargar en las propiedades del componente
 */
public void seleccionarFila(int i) {
    if (i <= Calificaciones.size()) {
        Calificacion c = (Calificacion) Calificaciones.elementAt(i);
        this.DNI = c.DNI;
        this.NombreCurso = c.NombreCurso;
        this.Curso = c.Curso;
        this.Nota = c.Nota;
    } else {
        this.DNI = "";
        this.NombreCurso = "";
        this.Curso = "";
        this.Nota = 0.0;
    }
}
```



Método seleccionar DNI

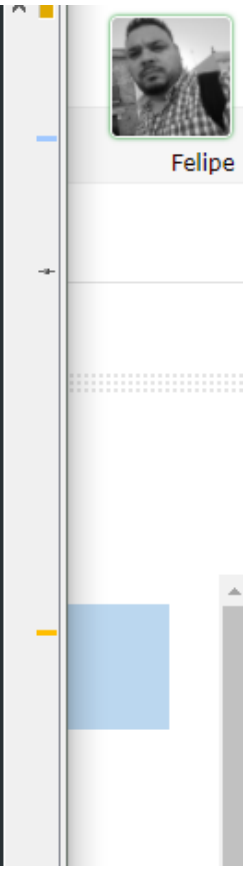
```java
/**
 * Método seleccionarDNI, que establece los atributos del componente por los
 * del objeto que coincida en DNI en el vector. En el caso de no existir, se
 * establecerán datos por defecto.
 *
 * Se ha corregido del ejemplo del temario al entrar en bucle constante usando
 * while. El funcionamiento es identico.
 *
 * Asimismo, no es un metodo completo dado que devolverá el único o el último
 * registro existente con dicho DNI.
 *
 * @param nDNI String con el DNI completo
 */
public void seleccionarDNI(String nDNI) {
    //se establecen datos por defecto
    this.DNI = "";
    this.NombreCurso = "";
    this.Curso = "";
    this.Nota = 0.0;
    //Se recorre el vector en búsqueda de la ultima coincidencia
    for (int i=0; i < Calificaciones.size(); i++) {
        //Se obtiene elemento a elemento del vector
        Calificacion c = (Calificacion) Calificaciones.elementAt(i);
        //Se compara con el DNI pasado por parametro, si es válido
        if (c.DNI.equals(nDNI)) {
            //Establecemos sus atributos como los del componente
            this.DNI = c.DNI;
            this.NombreCurso = c.NombreCurso;
            this.Curso = c.Curso;
            this.Nota = c.Nota;
        }
    }
}
```



Se ha modificado el while por un ciclo for (probado, no terminaba de funcionar, quedando en bucle continuo).

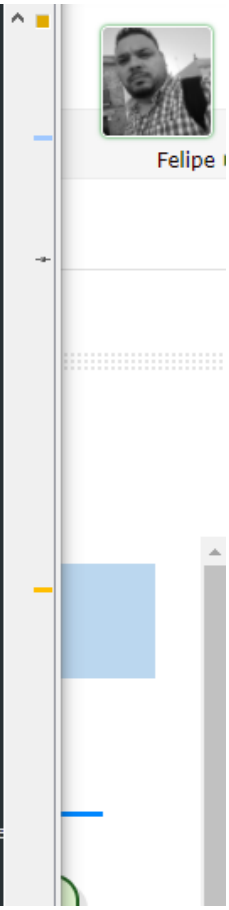
Añadiendo eventos

```java
/**
 * Código para añadir un nuevo alumno a la base de datos. cada vez que se
 * modifca el estado de la BD se genera un evento para que se recargue el
 * componente.
 */
private BDModificadaListener receptor;

//Clase que implementa los eventos
public class BDModificadaEvent extends java.util.EventObject {
    // constructor
    public BDModificadaEvent(Object source) {
        super(source);
    }
}

//Definiendo la interfaz de los eventos
public interface BDModificadaListener extends EventListener {
    //metodo para modificaciones generales de la BD
    public void capturarBDModificada(BDModificadaEvent ev);
    //metodo para modificaciones de la nota
    public void capturarNotaBDmodificada(BDModificadaEvent ev);
}

//añadir oyente
public void addBDModificadaListener(BDModificadaListener receptor) {
    this.receptor = receptor;
}
//eliminar oyente
public void removeBDModificadaListener(BDModificadaListener receptor) {
    this.receptor = null;
}
```



Método agregarNotaModulo

```java
/**
 * Método que añade una calificacion a la base de datos añade un registro a
 * la base de datos formado a partir de los valores de las propiedades del
 * componente.
 *
 * Se presupone que se han usado los métodos set para configurar
 * adecuadamente las propiedades con los datos del nuevo registro.
 *
 * @throws java.lang.ClassNotFoundException
 */
public void agregarNotaModulo() throws ClassNotFoundException {
    try {
        //conectamos
        Class.forName("com.mysql.jdbc.Driver");
        Connection con = DriverManager.getConnection(URL, USR, PSW);
        //preparamos la consulta
        PreparedStatement s = con.prepareStatement("INSERT INTO calificaciones"
                + "(DNI, NombreCurso, Curso, Nota) "
                + "VALUES (?,?,?,?)");
        //pasamos los datos por parametro
        s.setString(1, DNI);
        s.setString(2, NombreCurso);
        s.setString(3, Curso);
        s.setDouble(4, Nota);
        //ejecutamos la consulta
        s.executeUpdate();
        //recargamos el vector al existir cambios
        recargarFilas();
        //Lanzamos un evento
        receptor.capturarBDModificada(new BDModificadaEvent(this));
    } catch (SQLException ex) {
        Logger.getLogger(CalificacionesBean.class.getName()).log(Level.SEVERE, null, e
    }
}
```

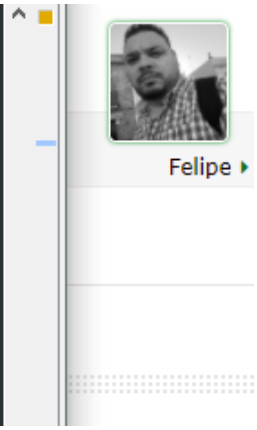


Método modificarNotaModulo

```java
/**
 * Método que modifica una calificacion a la base de datos añade un registro
 * a la base de datos formado a partir de los valores de las propiedades del
 * componente.
 *
 * Se presupone que se han usado los métodos set para configurar
 * adecuadamente las propiedades con los datos del nuevo registro.
 *
 * @throws java.lang.ClassNotFoundException
 */
public void modificarNotaModulo() throws ClassNotFoundException {
    try {
        //conectamos
        Class.forName("com.mysql.jdbc.Driver");
        Connection con = DriverManager.getConnection(URL, USR, PSW);
        //preparamos la consulta
        PreparedStatement s = con.prepareStatement(
                "UPDATE calificaciones SET nota = ? WHERE DNI = ?");
        //pasamos los datos por parametro
        s.setDouble(1, Nota);
        s.setString(2, DNI);
        //ejecutamos la consulta
        s.executeUpdate();
        //recargamos el vector al existir cambios
        recargarFilas();
        //lanzamos un evento
        receptor.capturarNotaBDmodificada(new BDModificadaEvent(this));
    } catch (SQLException ex) {
        Logger.getLogger(CalificacionesBean.class.getName()).log(Level.SEVERE, null, e
    }
}
```

Oyentes de los eventos

```java
/**
 * Operaciones con oyentes. Añadir oyente
 * @param listener
 */
public void addPropertyChangeListener(PropertyChangeListener listener) {
    propertySupport.addPropertyChangeListener(listener);
}
/**
 * Operaciones con oyentes. Eliminar oyente
 * @param listener
 */
public void removePropertyChangeListener(PropertyChangeListener listener) {
    propertySupport.removePropertyChangeListener(listener);
}
```



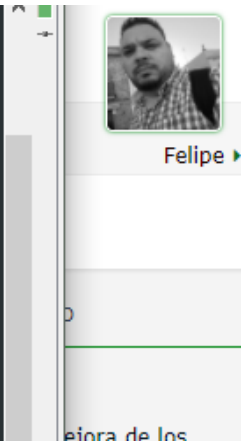
Implementando el componente

Sobre rutas del archivo .jar

El archivo .jar se ha mantenido en su ubicación original. Se ha añadido al proyecto con ruta relativas, asi cada vez que se haga *clean and build* en el proyecto1 estará actualizado en el proyecto2 siempre que se mantengan los nombre de los directorios inalterados y juntos.

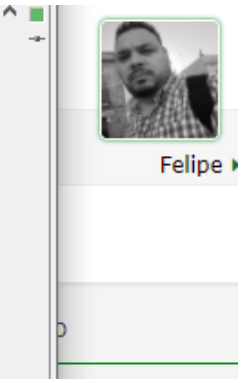
Creando la clase de conexión AccedeDB

```java
/**
 * Clase AccedeDB. Implementa la interfaz BDModificadaListener
 *
 * @author Felipon
 */
public class AccedeDB implements BDModificadaListener {

    //instanciamos el componente
    CalificacionesBean calificaciones;

    AccedeDB() {
        //indicamos que se añada un oyente de eventos
        calificaciones = new CalificacionesBean();
        calificaciones.addBDModificadaListener((BDModificadaListener) this);
    }
```



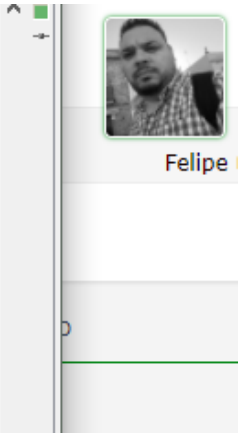
Método listado

```java
/**
 * Método listado(), muestra el listado de calificaciones
 */
public void listado() {
    //Bucle que comienza a obtener valores dependiendo del tamaño filas
    for (int i = 0; i < calificaciones.getTamanio(); i++) {
        calificaciones.seleccionarFila(i);
        System.out.println("DNI: " + calificaciones.getDNI());
        System.out.println("\tAsignatura: " + calificaciones.getNombreCurso());
        System.out.println("\tCurso: " + calificaciones.getCurso());
        System.out.println("\tNota: " + calificaciones.getNota());
    }
}
```



Método añade

```java
/**
 * Método anade(), que establece en el objeto estos datos y procede a
 * insertarlos en la BD.
 */
public void anade() {
    calificaciones.setDNI("98765432A");
    calificaciones.setNombreCurso("Entornos de Desarrollo");
    calificaciones.setCurso("15-16");
    calificaciones.setNota(5.6);
    try {
        calificaciones.agregarNotaModulo();
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(AccedeDB.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```



Método modifica

```java
/**
 * Método modifica(), que seleccionamos un DNI de existente en la BD y procede
 * a modificar su Nota.
 */
public void modifica() {
    //Seleccionamos el alumno que queremos modificar
    calificaciones.seleccionarDNI("98765432A");
    //Modificamos la nota
    calificaciones.setNota(8.2);
    try {
        //Guardamos los cambios
        calificaciones.modificarNotaModulo();
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(AccedeDB.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Métodos de los eventos

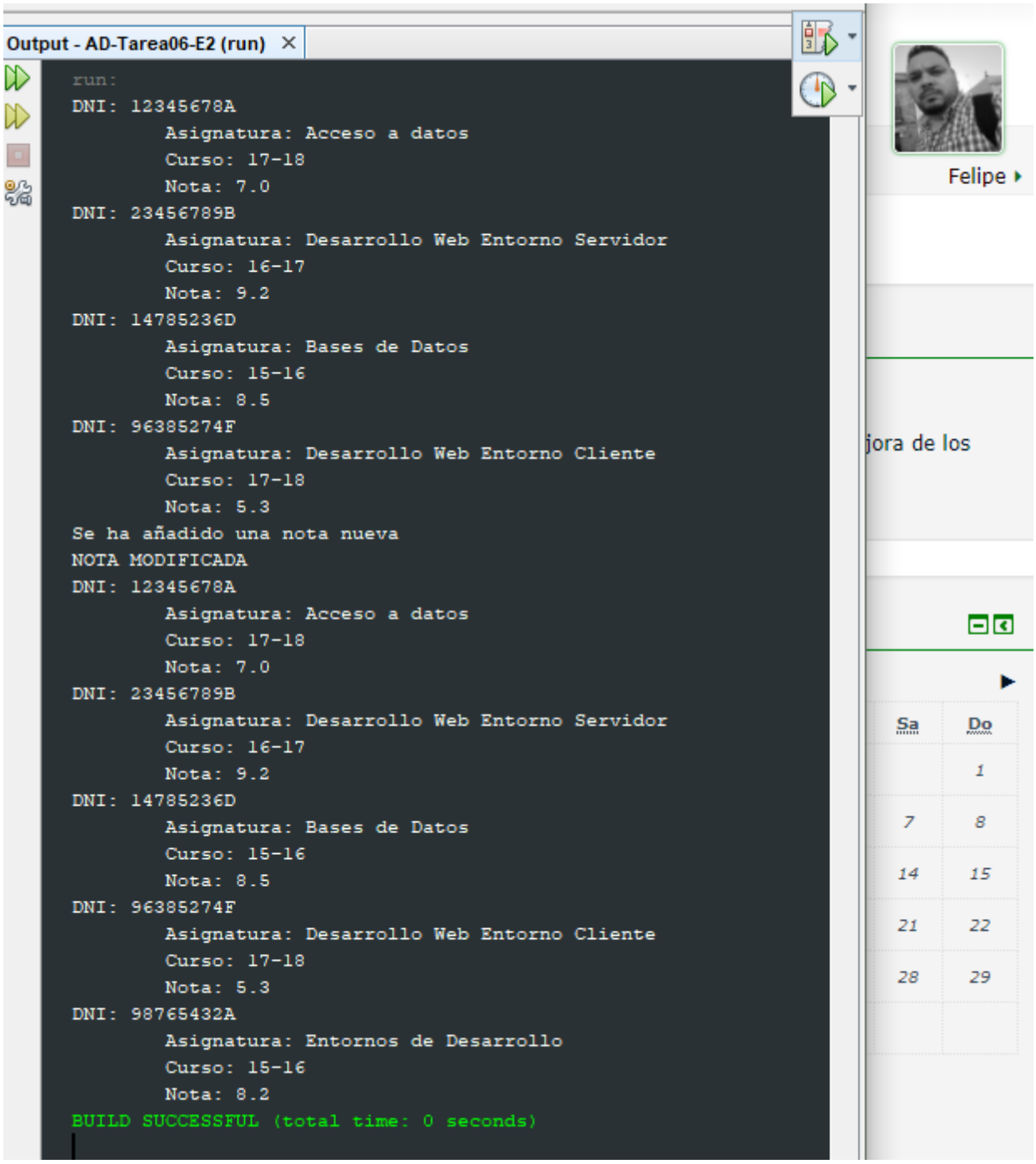
```java
@Override
/**
 * Metodo sobreescrito que lanza mensaje al recibir un evento
 */
public void capturarBDModificada(BDModificadaEvent ev) {
    System.out.println("Se ha añadido una nota nueva");
}

@Override
/**
 * Metodo sobreescrito que lanza mensaje al recibir un evento
 */
public void capturarNotaBDmodificada(BDModificadaEvent ev) {
    System.out.println("NOTA MODIFICADA");
}
```

Utilizando el componente

Incluyendo los métodos en el método main

```java
/**
 *
 * @author Felipon
 */
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        //instanciamos la clase
        AccedeDB gestion = new AccedeDB();
        //Mostramos el listado inicial
        gestion.listado();
        //Añadimos una calificación
        gestion.anade();
        //Modificamos una calificacion
        gestion.modifica();
        //Listamos de nuevo
        gestion.listado();
    }
}
```

Visualizando los resultados



Ejercicio 2

Conclusiones personales de la tarea 6

Posibles mejoras en tu programa y en el planteamiento/enunciado de la tarea

Como posible mejora, pues la que no falta nunca, una interfaz GUI, pero creo que podría restarle tiempo al ejercicio. Sobre el planteamiento, similar al ejercicio de ejemplo, pero salteado con algunas cosas más.

Dificultades que te has encontrado en la realización de la práctica

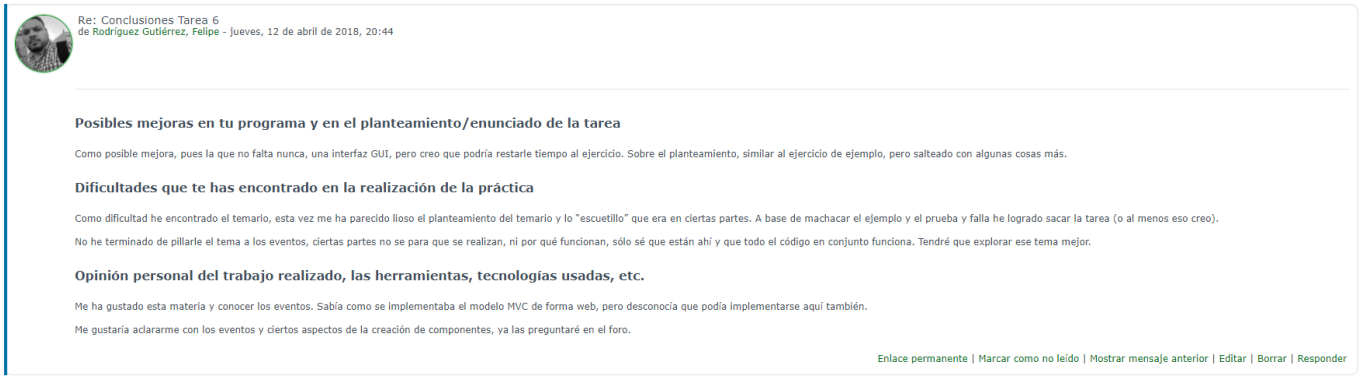
Como dificultad he encontrado el temario, esta vez me ha parecido lioso el planteamiento del temario y lo "escuetillo" que era en ciertas partes. A base de machacar el ejemplo a prueba y falla he logrado sacar la tarea (o al menos eso creo).

No he terminado de pillarle el tema a los eventos, ciertas partes no se para que se realizan, ni por qué funcionan, sólo sé que están ahí y que todo el código en conjunto funciona. Tendré que explorar ese tema mejor.

Opinión personal del trabajo realizado, las herramientas, tecnologías usadas, etc.

Me ha gustado esta materia y conocer los eventos. Sabía como se implementaba el modelo MVC de forma web, pero desconocía que podía implementarse aquí también.

Me gustaría aclararme con los eventos y ciertos aspectos de la creación de componentes, ya las preguntaré en el foro.


Re: Conclusiones Tarea 6
de Rodríguez Gutiérrez, Felipe - Jueves, 12 de abril de 2018, 20:44

Posibles mejoras en tu programa y en el planteamiento/enunciado de la tarea

Como posible mejora, pues la que no falta nunca, una interfaz GUI, pero creo que podría restarle tiempo al ejercicio. Sobre el planteamiento, similar al ejercicio de ejemplo, pero salteado con algunas cosas más.

Dificultades que te has encontrado en la realización de la práctica

Como dificultad he encontrado el temario, esta vez me ha parecido lioso el planteamiento del temario y lo "escuetillo" que era en ciertas partes. A base de machacar el ejemplo y el prueba y falla he logrado sacar la tarea (o al menos eso creo).

No he terminado de pillarle el tema a los eventos, ciertas partes no se para que se realizan, ni por qué funcionan, sólo sé que están ahí y que todo el código en conjunto funciona. Tendré que explorar ese tema mejor.

Opinión personal del trabajo realizado, las herramientas, tecnologías usadas, etc.

Me ha gustado esta materia y conocer los eventos. Sabía como se implementaba el modelo MVC de forma web, pero desconocía que podía implementarse aquí también.

Me gustaría aclararme con los eventos y ciertos aspectos de la creación de componentes, ya las preguntaré en el foro.

Enlace permanente | Marcar como no leído | Mostrar mensaje anterior | Editar | Borrar | Responder