

Base de Datos con Hibernate

PABLO HORCAJADA GONZÁLEZ
2ºDAM ISEP CEU

ÍNDICE

Contenido

Objetivo de la aplicación.....	2
Especificación de los requisitos	3
Diseño de la aplicación.....	4
Login	4
Menú de Entidades	4
Menú de Acciones.....	4
Nuevo Empleado	5
Crear Ordenador	5
Añadir Periféricos.....	5
Modificar Empleado	6
Modificar Ordenador	6
Modificar Periféricos	6
Ver Empleado	7
Ver Ordenador	7
Eliminar Empleado	8
Eliminar Ordenador	8
ControladorHibernate.....	9
AñadirEmpleado ()	9
AñadirPerifericos()	9
VerTodosEmpleados ()	10
VerTodosOrdenador()	11
ModifcarOrdenador ()	12
EliminarPeriferico()	14
Modelo de Datos.....	16
Empleados.....	16
Ordenadores	17
Periféricos	18

Objetivo de la aplicación

-El Objetivo de la aplicación es realizar la gestión de Empleados, Ordenadores, Periféricos y Dispositivos en una empresa de Informática usando la herramienta de mapeo objeto-relacional (ORM) llamada Hibernate

-Si en algún momento se incorpora alguien o algo nuevo, es de gran utilidad usar la aplicación.

-Si queremos dar de baja algún empleado u objeto, con poner el asiento en el que está dicha entidad o seleccionar en la tabla la fila en la que está, se daría de baja

-Si queremos cambiar o modificar alguna entidad, seleccionando la fila en la que se encuentra, se, se rellenará datos para que cambies el antiguo por el nuevo.

-Si queremos ver todos los empleados, periféricos, etc., deberemos elegir marcando al check que aparecerá en la ventana. En cambio, si queremos ver solo un dato en específico, con poner el asiento serviría.

Especificación de los requisitos

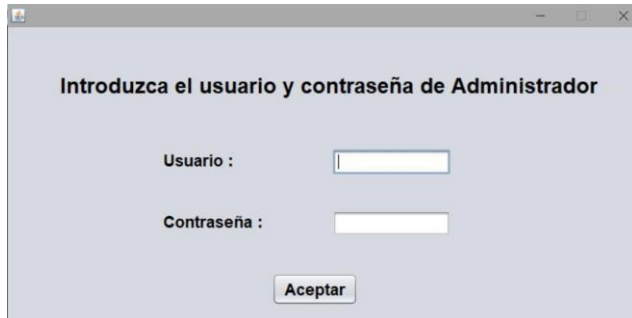
- El requisito fundamental para usar la aplicación, será usar la cuenta de Administrador, con los datos “AdminDam” como nombre de usuario y “2Dam123” como contraseña.
- Los requisitos de la aplicación serían poder modificar, eliminar, ver y añadir una nueva entidad a la base de datos.
- Deberemos poder realizar dichas acciones en mínimo 2 entidades y mostrando el proceso de operación de dichas acciones visualmente y escribiendo los datos que te pida la ventana en los encontramos
- Vamos a realizar la base de datos con 3 entidades: Empleados, Ordenadores y Dispositivos.

Diseño de la aplicación

-El diseño de la aplicación consistiría en una serie de ventanas que irás apareciendo y desapareciendo según la ventana que estés.

Login

-Una básica y simple ventana con un título, varios textos indicando un mensaje, dos cajitas para poder escribir el usuario y la contraseña, y un botón que pasaría a la siguiente ventana si el usuario es el del Administrador.



A screenshot of a login window titled "Introduzca el usuario y contraseña de Administrador". It features two input fields: "Usuario :" and "Contraseña :". Below the fields is a button labeled "Aceptar".

Menú de Entidades

-Una básica y simple ventana con un título, 4 botones que indican las entidades y un botón para cerrar el programa.

-Da igual que botón de entidad pulsemos, pasaremos a la siguiente ventana guardando a la entidad seleccionada en el código



A screenshot of a menu window titled "Menú de Entidades". It contains four buttons arranged in a 2x2 grid: "Empleado", "Dispositivos", "Ordenador", and "Periféricos". At the bottom center is a button labeled "Cerrar".

Menú de Acciones

-Una básica y simple ventana con cuatro botones que representan las acciones, y un botón para volver a la ventana anterior.

-Dependiendo de qué entidad elegimos en ventana anterior, al pulsar en una acción pasaremos a la siguiente ventana que corresponderá a la acción elegida para esa entidad.



A screenshot of a menu window titled "Menú de acciones". It contains four buttons arranged in a 2x2 grid: "Añadir", "Modificar", "Ver", and "Eliminar". At the bottom center is a button labeled "Volver".

Nuevo Empleado

-Ventana con un título, 4 textos indicando los datos, 4 tablas para escribir los datos indicados para el empleado y dos botones que sirven para volver a la ventana anterior y guardar el nuevo empleado .

The screenshot shows a window titled "Nuevo Empleado". It contains four labels with corresponding input fields: "Nombre :", "Apellidos :", "Departamento :", and "Asiento". At the bottom, there are two buttons: "Volver" and "Guardar".

Crear Ordenador

-Ventana con un título, textos y cajitas pidiendo el dato, y dos botones para realizar la acción y volver a la ventana anterior.

The screenshot shows a window titled "Crear Ordenador". It features a label "Asiento" with an input field. Below this, there are three rows of labels and input fields: "Placa", "Alimentacion", "Disco duro" in the first row; "Procesador", "Tarjeta Grafica", "Disco duro ssd" in the second row; and "Memoria Ram", "Torre", "Ventilacion" in the third row. At the bottom, there are two buttons: "Volver" and "Crear".

Añadir Periféricos

-Ventana con un título, textos y cajitas pidiendo el dato, y dos botones para realizar la acción y volver a la ventana anterior.

The screenshot shows a window titled "Añadir Periféricos". It contains six labels with corresponding input fields arranged in two columns: "Monitor 1", "Monitor 2", "Raton" on the left and "Teclado", "Disco portatil", "Asiento" on the right. At the bottom, there are two buttons: "Volver" and "Añadir".

Modificar Empleado

-Ventana con un título, 4 textos indicando los datos, 4 cajitas para poner datos, dos botones para volver a la ventana anterior o modificar el empleado y una tabla con los datos de los empleados.

Modificar un empleado

SITIO	NOMBRE	APELLIDOS	DEPARTAMENTO
25	Juan	Cholo	Ventas
56	Pablo	sdf	Publi

Asiento:

Usuario:

Apellidos:

Departamento:

Modificar Ordenador

-Ventana con un título, textos y cajitas donde te indica los datos a modificar, y los dos botones para volver a la ventana anterior y modificar la entidad, y una tabla con los datos de los ordenadores.

Modificar Ordenador

SITIO	PLACA	PROCESADOR	GRÁFICA	RAM	ALIMENTACION	DISCO-SOLIDO	DISCO-SSD	VENTILACION
12	Asus	ad	asd	asd	ad	asd	asd	ad
25	Placa-A	Procesador-A	Tarjeta_grafica-A	Memoria-A	Alimentacion	Disco-A	Disco-B	Ventilacion-A

Asiento: Ventilacion: Alimentacion:

Tarjeta Grafica: Placa: Disco duro:

Procesador: Memoria Ram: Disco duro ssd:

Modificar Periféricos

-Ventana con un título, textos y cajitas donde te indica los datos a modificar, y los dos botones para volver a la ventana anterior y modificar la entidad, y una tabla con los datos de los periféricos.

Modificar Periféricos

SITIO	MONITOR1	MONITOR2	RATON	TECLADO	DISCOPORTAL...
25	Monitor-A	Monitor-B	Raton-A	Teclado-A	Discopor-A
56	AOG	ghj	ghj	ghj	ghj

Asiento: Monitor1: Monitor2:

Raton: Teclado: Disco portatil:

Ver Empleado

-Ventana con un título, un texto indicando el dato que hay que escribir, una caja para escribir el dato que pide, un check por si queremos ver todos los empleados y una tabla vacía que se rellenará con los datos de todos los empleados si le damos al check.

The 'Ver Empleado' form has a title bar. Below it is a label 'Introduce el asiento del Empleado que quieres ver' followed by a text input field. Below the input field is a checkbox labeled 'Marca esta casilla si quieres ver todos los empleados'. Below the checkbox is a table with three columns labeled A, B, and C. The table body is empty. At the bottom of the form are two buttons: 'Volver' and 'Ver'.

A	B	C
---	---	---

Ver Ordenador

-Ventana con un título, un texto indicando el dato que hay que escribir, una caja para escribir el dato que pide, un check por si queremos ver todos los ordenadores, y una tabla vacía que se rellenará con los datos de todos los ordenadores si le damos al check.

The 'Ver Ordenador' form has a title bar. Below it is a label 'Introduce el asiento del ordenador' followed by a text input field. Below the input field is a checkbox labeled 'Marca esta casilla si quieres ver todos los ordenadores'. Below the checkbox is a table with eight columns labeled A, B, C, D, E, F, G, and H. The table body is empty. At the bottom of the form are two buttons: 'Volver' and 'Ver'.

A	B	C	D	E	F	G	H
---	---	---	---	---	---	---	---

Ver Periféricos

-Ventana con un título, un texto indicando el dato que hay que escribir, una caja para escribir el dato que pide, un check por si queremos ver todos los ordenadores, y una tabla vacía que se rellenará con los datos de todos los ordenadores si le damos al check.

The 'Ver Periféricos' form has a title bar. Below it is a label 'Introduce el Asiento donde se encuentran los periféricos' followed by a text input field. Below the input field is a checkbox labeled 'Marca esta casilla para ver todos los Periféricos'. Below the checkbox is a table with five columns labeled A, B, C, D, and E. The table body is empty. At the bottom of the form are two buttons: 'Volver' and 'Ver'.

A	B	C	D	E
---	---	---	---	---

Eliminar Empleado

-Ventana con un título, dos botones para volver a la ventana anterior o eliminar, un texto indicando el dato, y una tabla con todos los datos de los empleados.

Eliminar Empleado

SITIO	NOMBRE	APELLIDOS	DEPARTAMENTO
25	Juan	Cholo	Ventas
56	Pablo	sdf	Publi

VolverEliminar

Eliminar Ordenador

-Ventana con un título, dos botones para volver a la ventana anterior y eliminar el ordenador, y una tabla con todos los datos de los ordenadores.

Eliminar Ordenador

SITIO	PLACA	PROCESADOR	GRÁFICA	RAM	ALIMENTACION	DISCO-SOLIDO	DISCO-SSD	VENTILACION
12	Asus	ad	asd	asd	ad	asd	asd	ad
25	Placa-A	Procesador-A	Tarjeta_grafica...	Memoria-A	Alimentacion	Disco-A	Disco-B	Ventilacion-A

VolverEliminar

Eliminar Periféricos

-Ventana con 3 botones, 1 para volver y dos para eliminar con la tabla o con el check, una cajita para poner el asiento, textos indicando los que quieres, checkbox por cada periférico, y una tabla con todos los periféricos por si queremos eliminar todo el conjunto de periféricos.

Eliminar Periferico

Introduce el asiento donde quieres eliminar algún Periferico

Nota : si solo quieres eliminar un periférico de ese asiento, marca la casilla de ese periférico

☐ Monitor1☐ Monitor2☐ Raton
☐ Teclado☐ Disco portatil

VolverEliminar por Asiento

Selecciona en la tabla el conjunto de Perifericos que quieres eliminar

SITIO	MONITOR1	MONITOR2	RATON	TECLADO	DISCOPORTATIL
25	Monitor-A	Monitor-B	Raton-A	Teclado-A	Discopor-A
56	AOC	ghj	ghj	ghj	ghj

Eliminar Con tabla

ControladorHibernate

-A continuación, mostraré algunos métodos que utilizo (ya que todos son iguales pero cambiando nombres y valores, porque todos realizan las 4 acciones)

AñadirEmpleado ()

-Al pulsar el botón de guardar empleado se ejecutará el método AñadirEmpleado (Empleado empleados) del controlador. El texto que estaba en las cajitas se añade en un objeto “Empleado”, para posteriormente, añadir dicho objeto a la base de datos. La línea que realiza el “añadir” es el `sesión.save(empleados)`, ya que “guarda” en la fabrica el objeto que le pasemos.

```
private void jButtonGuardarActionPerformed(java.awt.event.ActionEvent evt) {  
    if (!jTextFieldapellidos.getText().equals("") || !jTextFieldnombre.getText().equals("")) {  
        jLabelerror.setVisible(true);  
    }  
    else{  
        String nombre = jTextFieldnombre.getText();  
        String apellidos = jTextFieldapellidos.getText();  
        String dept = jTextFielddepartamento.getText();  
        int sitio = Integer.parseInt(jTextFieldasiento.getText());  
        Empleados nuevo = new Empleados(sitio,nombre,apellidos,dept);  
        Controlador.Operaciones.AñadirEmpleado(nuevo);  
        this.jLabelerror.setText("Empleado añadido");  
    }  
}  
  
//Añadir un Empleado  
public static void AñadirEmpleado(Empleados empleados){  
    Session session = null;  
    Transaction tx = null;  
    try{  
        SessionFactory sf = NewHibernateUtil.getSessionFactory();  
        session = sf.openSession();  
        tx = session.beginTransaction();  
        session.save(empleados);  
        tx.commit();  
        JOptionPane.showMessageDialog(null, "Insertado correctamente");  
    }catch(HibernateException he){  
        tx.rollback();  
        JOptionPane.showMessageDialog(null, "Clave duplicada" + he.getLocalizedMessage());  
    }finally{  
        session.close();  
    }  
}
```

AñadirPerifericos()

- Al pulsar el botón de Añadir, se ejecutará el método AñadirPerifericos (Perifericos periferico) del controlador. El texto que estaba en las cajitas se añade en un objeto “Periferico”, para posteriormente, añadir dicho objeto a la base de datos. La línea que realiza el “añadir” es el `sesión.save(periferico)`, ya que “guarda” en la fábrica el objeto que le pasemos.

```
private void jButtonAñadirActionPerformed(java.awt.event.ActionEvent evt) {  
    if (!jTextFieldasiento.getText().equals("") || !jTextFielddisco.getText().equals("") || !jTextFieldmonitor1.getText().equals("") || !jTextFieldmonitor2.getText().equals("") || !jTextFieldraton.getText().equals("") || !jTextFieldteclado.getText().equals("")){  
        jLabelerror.setText("No puedes dejar ningún campo vacío");  
    }  
    else{  
        String monitor = jTextFieldmonitor1.getText();  
        String monitor2 = jTextFieldmonitor2.getText();  
        String raton = jTextFieldraton.getText();  
        String teclado = jTextFieldteclado.getText();  
        String disco = jTextFielddisco.getText();  
        int asiento = Integer.parseInt(jTextFieldasiento.getText());  
        Perifericos p = new Perifericos(asiento,monitor,monitor2,raton,teclado,disco);  
        Controlador.Operaciones.AñadirPeriferico(p);  
    }  
}  
  
//Añadir un Periferico  
public static void AñadirPeriferico(Perifericos periferico){  
    Session session = null;  
    Transaction tx = null;  
    try{  
        SessionFactory sf = NewHibernateUtil.getSessionFactory();  
        session = sf.openSession();  
        tx = session.beginTransaction();  
        session.save(periferico);  
        tx.commit();  
        JOptionPane.showMessageDialog(null, "Insertado correctamente");  
    }catch(HibernateException he){  
        tx.rollback();  
        JOptionPane.showMessageDialog(null, "Clave duplicada" + he.getLocalizedMessage());  
    }finally{  
        session.close();  
    }  
}
```

VerTodosEmpleados ()

-Dependiendo de si escribimos un asiento en la cajita, ejecutaremos los métodos PrepararTabla() y LlenarTabla(). El primero lo que hace es crear las columnas, y el segundo, rellena las columnas y las filas con los datos correspondientes, llamando al método VerTodosEmpleados() del controlador, que con una query, lista todos los usuarios y devuelve un objeto DefaultListModel con todos los datos de todos los empleados. Si solo ponemos el asiento, aparece una ventanita con los datos de ese empleado.

```
private void jButtonVerActionPerformed(java.awt.event.ActionEvent evt) {  
    //Si marcamos la casilla de ver todos  
    if (!jChecktodos.isSelected() && jTextFieldasiento.getText().equals("")) {  
        jLabel4.setText("Debes marcar la casilla o poner un asiento");  
    }  
    else {  
        if (jChecktodos.isSelected()) {  
            PrepararTabla();  
            LlenarTabla();  
        }  
        if (!jTextFieldasiento.getText().equals("")) {  
            Empleados e = cper.VerEmpleado(Integer.parseInt(jTextFieldasiento.getText()));  
            JOptionPane.showMessageDialog(this, e.toString());  
        }  
    }  
}
```

```
private void PrepararTabla() {  
    String titulos[] = {"SITIO", "NOMBRE", "APELLIDOS", "DEPARTAMENTO"};  
    m = new DefaultTableModel(null, titulos);  
    jTable1.setModel(m);  
}  
  
private void LlenarTabla() {  
    String titulos[] = {"SITIO", "NOMBRE", "APELLIDOS", "DEPARTAMENTO"};  
    m = new DefaultTableModel(null, titulos);  
    dlm = cper.VerTodosEmpleados();  
    String fila[] = new String[4];  
    for (int i = 0; i < dlm.size(); i++) {  
        Empleados aux = (Empleados) dlm.get(i);  
        fila[0] = "" + aux.getSitio();  
        fila[1] = aux.getNombre();  
        fila[2] = aux.getApellidos();  
        fila[3] = aux.getDepartamento();  
        m.addRow(fila);  
    }  
    jTable1.setModel(m);  
}
```

```
//Ver todos los empleados  
public DefaultListModel VerTodosEmpleados() {  
    SessionFactory session = NewHibernateUtil.getSessionFactory();  
    Session session = session.openSession();  
    Transaction tx = session.beginTransaction();  
    List<Empleados> lista = null;  
    try {  
        Query q;  
        q = session.createQuery("from Empleados");  
        lista = (List<Empleados>) q.list();  
        tx.commit();  
    } catch (Exception e) {  
        System.out.println(e.getMessage() + "query incorrecta");  
    } finally {  
        session.close();  
    }  
    Iterator<Empleados> iter = lista.iterator();  
    DefaultListModel dlm = new DefaultListModel();  
    while (iter.hasNext()) {  
        Empleados e = (Empleados) iter.next();  
        dlm.addElement(e);  
    }  
    return dlm;  
}
```

VerTodosOrdenador()

-Dependiendo de si escribimos un asiento en la cajita, ejecutaremos los métodos PrepararTabla() y LlenarTabla(). El primero lo que hace es crear las columnas, y el segundo, rellena las columnas y las filas con los datos correspondientes, llamando al método VerTodosOrdenador () del controlador, que, con una query, lista todos los ordenadores y devuelve un objeto DefaultListModel con todos los datos de todos los ordenadores. Si solo ponemos el asiento, aparece una ventanita con los datos de ese ordenador.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    if (!jCheckBox1.isSelected() && jTextFieldAsiento.getText().equals("")){  
        JOptionPane.showMessageDialog(null, "Debes marcar la casilla o poner un asiento");  
    }  
    else {  
        if (jCheckBox1.isSelected()){  
            LlenarTabla();  
            PrepararTabla();  
        }  
        if (!jTextFieldAsiento.getText().equals("")){  
            Ordenadores o = oper.VerOrdenador(Integer.parseInt(this.jTextFieldAsiento.getText()));  
            JOptionPane.showMessageDialog(this, o.toString());  
        }  
    }  
}
```

```
private void PrepararTabla() {  
    String titulos[]=new String[8];  
    m=new DefaultTableModel(null,titulos);  
    jTable1.setModel(m);  
}  
  
private void LlenarTabla() {  
    String titulos[]={"SITIO","PLACA","PROCESADOR","GRÁFICA","RAM","ALIMENTACION","DISCO-SOLIDO","DISCO-SSD","VENTILACION"};  
    m=new DefaultTableModel(null,titulos);  
    dlm=oper.VerTodosOrdenador();  
    String fila[]= new String[9];  
    for (int i=0;i<dlm.size();i++){  
        Ordenadores aux= (Ordenadores) dlm.get(i);  
        fila[0]=""+aux.getSitio();  
        fila[1]=aux.getPlaca();  
        fila[2]=aux.getProcesador();  
        fila[3]=aux.getTarjetaGrafica();  
        fila[4]=aux.getMemoriaRam();  
        fila[5]=aux.getAlimentacion();  
        fila[6]=aux.getDiscoDuro();  
        fila[7]=aux.getDiscoSsd();  
        fila[8]=aux.getVentilacion();  
        m.addRow(fila);  
    }  
    jTable1.setModel(m);  
}
```

```
//Ver todos los ordenadores  
public DefaultListModel VerTodosOrdenador(){  
    SessionFactory sesion = NewHibernateUtil.getSessionFactory();  
    Session session = session.openSession();  
    Transaction tx = session.beginTransaction();  
    List<Ordenadores> lista=null;  
    try{  
        Query q;  
        q = session.createQuery("from Ordenadores");  
        lista = (List<Ordenadores>)q.list();  
        tx.commit();  
    }catch (Exception e){  
        System.out.println(e.getMessage()+"query incorrecta");  
    }finally {  
        session.close();  
    }  
    Iterator<Ordenadores> iter=lista.iterator();  
    DefaultListModel dlm = new DefaultListModel();  
    while(iter.hasNext()){  
        Ordenadores o = (Ordenadores) iter.next();  
        dlm.addElement(o);  
    }  
    return dlm;  
}
```

ModificarOrdenador ()

-Nada más cargar la ventana, se ejecutan los métodos PrepararTabla() y LLenarTabla() que crean y rellenan las columnas y las filas con los datos de todos los ordenadores, llamando al método VerTodosOrdenadores(). Si pulsamos en cualquier fila de la tabla, se ejecuta el método ObtenerDeLaTabla(), que rellena las cajitas con los datos de cada columna de la fila a la que hemos pulsado. Si le damos al botón de modificar, ejecutamos el método chequearEntradas(), que comprueba que no hay ninguna cajita vacía, y posteriormente, crea un objeto nuevo de la clase Ordenadores con los datos de cada cajita, para luego ejecutar el método ModificarOrdenar(), pasándole el nuevo ordenador que hemos creado. La función de ModificarOrdenar() es que al pasarle un objeto de la clase Empleados, “carga”(sesión.load) el Ordenador que queremos modificar a partir de su clave(empleados.getSitio()), para luego establecer con el .set los nuevos datos(con los datos que del nuevo Ordenador) del ordenador y actualizarlo con el “sesión.update”.

```
private void PrepararTabla() {
    String titulos[]=new String[8];
    m=new DefaultTableModel(null,titulos);
    jTable1.setModel(m);
}

private void LLenarTabla() {
    String titulos[]={"SITIO","PLACA","PROCESADOR","GRÁFICA","RAM","ALIMENTACION","DISCO-SOLIDO","DISCO-SSD","VENTILACION"};
    m=new DefaultTableModel(null,titulos);
    dlm=per.VerTodosOrdenador();
    String fila[]= new String[9];
    for (int i=0;i<dlm.size();i++){
        Ordenadores aux= (Ordenadores) dlm.get(i);
        fila[0]=""+aux.getSitio();
        fila[1]=aux.getPlaca();
        fila[2]=aux.getProcesador();
        fila[3]=aux.getTarjetaGrafica();
        fila[4]=aux.getMemoriaRam();
        fila[5]=aux.getAlimentacion();
        fila[6]=aux.getDiscoDuro();
        fila[7]=aux.getDiscoSsd();
        fila[8]=aux.getVentilacion();
        m.addRow(fila);
    }
    jTable1.setModel(m);
}
```

```
private void jTable1MouseClicked(java.awt.event.MouseEvent evt) {
    ObtenerDeLaTabla();
}
```

```
private void ObtenerDeLaTabla() {
    int filse;
    filse = jTable1.getSelectedRow();
    jTextFieldasiento.setText((String) m.getValueAt(filse, 0));
    jTextFieldplaca.setText((String) m.getValueAt(filse, 1));
    jTextFieldprocesador.setText((String) m.getValueAt(filse, 2));
    jTextFieldtarjeta.setText((String) m.getValueAt(filse, 3));
    jTextFieldram.setText((String) m.getValueAt(filse, 4));
    jTextFieldalimentacion.setText((String) m.getValueAt(filse, 5));
    jTextFielddisco.setText((String) m.getValueAt(filse, 6));
    jTextFieldssd.setText((String) m.getValueAt(filse, 7));
    jTextFieldventilacion.setText((String) m.getValueAt(filse, 8));
}
```



```

private void jButtonModificarActionPerformed(java.awt.event.ActionEvent evt) {
    chequearEntradas();
    int filsel;
    try{
        filsel=jTable.getSelectedRow();
        if (filsel==-1){
            JOptionPane.showMessageDialog(null, "Debes seleccionar Empleado a Modificar.");
        }else{
            boolean b=chequearEntradas();
            if(b){
                int sitio = Integer.parseInt(jTextFieldAsiento.getText());
                String placa = jTextFieldPlaca.getText();
                String procesador = jTextFieldProcesador.getText();
                String tarjeta = jTextFieldTarjeta.getText();
                String memoria = jTextFieldRam.getText();
                String ali = jTextFieldAlimentacion.getText();
                String duro = jTextFieldDisco.getText();
                String ss = jTextFieldSsd.getText();
                String venti = jTextFieldVentilacion.getText();
                Ordenadores o = new Ordenadores(sitio,placa,procesador,tarjeta,memoria,ali,duro,ss,venti);
                oper.ModificarOrdenador(o);
                LLenarTabla();
            }
        }
    } catch (Exception e){
        JOptionPane.showMessageDialog(null,e+ "Error al modificar el empleado");
    }
}

```

```

boolean chequearEntradas() {
    boolean b=true;
    String mensaje="";
    if(jTextFieldAsiento.getText().equals("")){
        mensaje+="El asiento no se puede quedar vacio\n";
        b=false;
    }
    int num;
    try{
        num=Integer.parseInt(jTextFieldAsiento.getText());
    }catch (Exception e ) {
        mensaje+="Formato de número incorrecto\n";
        b=false;
    }
    if(jTextFieldPlaca.getText().equals("")){
        mensaje+="La placa no se puede quedar vacio\n";
        b=false;
    }
    if(jTextFieldProcesador.getText().equals("")){
        mensaje+="El procesador no se puede quedar vacio\n";
        b=false;
    }
    if(jTextFieldTarjeta.getText().equals("")){
        mensaje+="La tarjeta gráfica no se puede quedar vacio\n";
    }
}

```

```

//Modificar un empleado
public void ModificarEmpleado(Empleados empleado){
    Transaction tx=null;
    Session session=null;
    try{
        SessionFactory session = NewHibernateUtil.getSessionFactory();
        session = session.openSession();
        tx = session.beginTransaction();
        int key = empleado.getSitio();
        Empleados empl = (Empleados)session.load(Empleados.class, key);
        empl.setSitio(empleado.getSitio());
        empl.setNombre(empleado.getNombre());
        empl.setApellidos(empleado.getApellidos());
        empl.setDepartamento(empleado.getDepartamento());
        System.out.println(empleado.toString());
        session.update(empl);
        tx.commit();
        JOptionPane.showMessageDialog(null,"Actualizado correctamente.");
    }catch (HibernateException he) {
        tx.rollback();
        JOptionPane.showMessageDialog(null,he.getMessage()+"No ha sido posible actualizar el Empleado");
    } finally {
        session.close();
    }
}

```

EliminarPeriferico()

- Nada más cargar la ventana, se ejecutan los métodos PrepararTabla() y LlenarTabla() que crean y rellenan las columnas y las filas con los datos de todos los periféricos, llamando al método VerTodosperifericos(). Si seleccionamos cualquier fila y le damos al botón de “Eliminar Con tabla”, guardamos en una variable, el asiento(columna) de la fila que hemos seleccionado, para ejecutar el método EliminarPerifericoTabla() del controlador, pasándole dicha variable. EliminarPerifericoTabla() lo que hace es cargar(.load) el Periferico que tenga ese asiento, para luego borrarlo(.delete) de la base de datos. Si escribimos en la cajita un asiento que exista, y marcamos cualquier check, pondrá como null (es decir, que los “borra”) dicho periférico de ese asiento.

```
private void PrepararTabla() {
    String titulos[] = new String[5];
    m = new DefaultTableModel(null, titulos);
    jTable1.setModel(m);
}

private void LlenarTabla() {
    String titulos[] = {"SITIO", "MONITOR1", "MONITOR2", "RATON", "TECLADO", "DISCO PORTATIL"};
    m = new DefaultTableModel(null, titulos);
    dim = oper.VerTodosPerifericos();
    String fila[] = new String[6];
    for (int i = 0; i < dim.size(); i++) {
        Perifericos aux = (Perifericos) dim.get(i);
        fila[0] = "" + aux.getSitio();
        fila[1] = aux.getMonitor1();
        fila[2] = aux.getMonitor2();
        fila[3] = aux.getRaton();
        fila[4] = aux.getTeclado();
        fila[5] = aux.getDiscoPortatil();
        m.addRow(fila);
    }
    jTable1.setModel(m);
}
```

```
private void jButtonEliminarConTablaActionPerformed(java.awt.event.ActionEvent evt) {
    int filsel;
    int resp;
    try {
        filsel = jTable1.getSelectedRow();
        if (filsel == -1) {
            JOptionPane.showMessageDialog(null, "Debes seleccionar la fila de Perifericos a borrar.");
        } else {
            resp = JOptionPane.showConfirmDialog(null, "Desea eliminar los Perifericos seleccionados?", "Eliminar", JOptionPane.YES_NO_OPTION);
            if (resp == JOptionPane.YES_OPTION) {
                int selectedRow = filsel;
                String asiento = (String) m.getValueAt(selectedRow, 0);
                int clave = Integer.parseInt(asiento);
                oper.EliminarPerifericoTabla(clave);
                LlenarTabla();
            }
        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, e, "Error al eliminar la noticia.");
    }
}
```

//Eliminar un periférico usando la tabla

```
public void EliminarPerifericoTabla(Integer asiento) {
    Transaction tx = null;
    Session session = null;
    try {
        SessionFactory session = NewHibernateUtil.getSessionFactory();
        session = session.openSession();
        tx = session.beginTransaction();
        Perifericos p = (Perifericos) session.load(Perifericos.class, asiento);
        session.delete(p);
        tx.commit();
        JOptionPane.showMessageDialog(null, "Eliminado correctamente.");
    } catch (HibernateException he) {
        tx.rollback();
        JOptionPane.showMessageDialog(null, he.getMessage() + "No ha sido posible eliminar el periférico");
        System.out.println("Mensaje de la baja " + he.getMessage());
    } finally {
        session.close();
    }
}
```

```
private void jBEliminarAsientoActionPerformed(java.awt.event.ActionEvent evt) {
    if (jTextFieldasiento.getText().equals("")) {
        JOptionPane.showMessageDialog(null, "Debes escribir un asiento para eliminar algún periférico");
    }
    else{
        Controlador.Operaciones.EliminarPeriferico();
    }
}
```

```
public static void EliminarPeriferico(){
    Transaction tx=null;
    Session session=null;
    try{
        int asiento = Integer.parseInt(Vista.EliminarPeriferico.jTextFieldasiento.getText());
        Perifericos per = (Perifericos) session.createQuery("SELECT p FROM Perifericos p WHERE sitio="+asiento).
        if(Vista.EliminarPeriferico.jCheckBoxm1.isSelected()){
            per.setMonitor1("Null");
        }
        if(Vista.EliminarPeriferico.jCheckBoxm2.isSelected()){
            per.setMonitor1("Null");
        }
        if(Vista.EliminarPeriferico.jCheckBoxr.isSelected()){
            per.setMonitor1("Null");
        }
        if(Vista.EliminarPeriferico.jCheckBoxt.isSelected()){
            per.setMonitor1("Null");
        }
        if(Vista.EliminarPeriferico.jCheckBoxd.isSelected()){
            per.setMonitor1("Null");
        }
        JOptionPane.showMessageDialog(null,"Periferico/s eliminado/s");
    } catch (HibernateException he) {
        tx.rollback();
        JOptionPane.showMessageDialog(null,he.getMessage()+"No ha sido posible eliminar el periferico");
        System.out.println("Mensaje de la baja "+he.getMessage());
    } finally {
        session.close();
    }
}
```


Modelo de Datos

Empleados

-Atributos

```
private int sitio;  
private String nombre;  
private String apellidos;  
private String departamento;
```

-Constructores

```
public Empleados() {  
}  
  
public Empleados(int sitio) {  
    this.sitio = sitio;  
}  
  
public Empleados(int sitio, String nombre, String apellidos, String departamento) {  
    this.sitio = sitio;  
    this.nombre = nombre;  
    this.apellidos = apellidos;  
    this.departamento = departamento;  
}
```

-ToString

```
@Override  
public String toString() {  
    return "Sitio: " + sitio + ", Nombre: " + nombre +  
        ", Apellidos: " + apellidos + ", Departamento: " + departamento;  
}
```

-Getters y Setters

```
public Integer getSitio() {  
    return this.sitio;  
}  
  
public void setSitio(Integer sitio) {  
    this.sitio = sitio;  
}  
  
public String getNombre() {  
    return this.nombre;  
}  
  
public void setNombre(String nombre) {  
    this.nombre = nombre;  
}  
  
public String getApellidos() {  
    return this.apellidos;  
}  
  
public void setApellidos(String apellidos) {  
    this.apellidos = apellidos;  
}  
  
public String getDepartamento() {  
    return this.departamento;  
}  
  
public void setDepartamento(String departamento) {  
    this.departamento = departamento;  
}
```

Ordenadores

-Atributos

```
private int sitio;  
private String placa;  
private String procesador;  
private String tarjetaGrafica;  
private String memoriaRam;  
private String alimentacion;  
private String discoDuro;  
private String discoSsd;  
private String ventilacion;
```

-Constructores

```
public Ordenadores() {  
}  
  
public Ordenadores(int sitio) {  
    this.sitio = sitio;  
}  
  
public Ordenadores(int sitio, String placa, String procesador, String tarjetaGrafica, String memoriaRam, String alimentacion, String discoDuro, String discoSsd, String ventilacion) {  
    this.sitio = sitio;  
    this.placa = placa;  
    this.procesador = procesador;  
    this.tarjetaGrafica = tarjetaGrafica;  
    this.memoriaRam = memoriaRam;  
    this.alimentacion = alimentacion;  
    this.discoDuro = discoDuro;  
    this.discoSsd = discoSsd;  
    this.ventilacion = ventilacion;  
}
```

-ToString

```
@Override  
public String toString() {  
    return "Sitio: " + sitio + ", Placa: " + placa + ", Procesador: " + procesador +  
        ", TarjetaGrafica: " + tarjetaGrafica + ", MemoriaRam: " + memoriaRam +  
        ", Alimentacion: " + alimentacion + ", DiscoDuro: " + discoDuro + ", DiscoSsd: " +  
        discoSsd + ", Ventilacion: " + ventilacion;  
}
```

-Getters y Setters

```
public void setTarjetaGrafica(String tarjetaGrafica) {  
    this.tarjetaGrafica = tarjetaGrafica;  
}  
public String getMemoriaRam() {  
    return this.memoriaRam;  
}  
public void setMemoriaRam(String memoriaRam) {  
    this.memoriaRam = memoriaRam;  
}  
public String getAlimentacion() {  
    return this.alimentacion;  
}  
public void setAlimentacion(String alimentacion) {  
    this.alimentacion = alimentacion;  
}  
public String getDiscoDuro() {  
    return this.discoDuro;  
}  
public void setDiscoDuro(String discoDuro) {  
    this.discoDuro = discoDuro;  
}  
public String getDiscoSsd() {  
    return this.discoSsd;  
}  
public void setDiscoSsd(String discoSsd) {  
    this.discoSsd = discoSsd;  
}  
public String getVentilacion() {  
    return this.ventilacion;  
}  
public void setVentilacion(String ventilacion) {  
    this.ventilacion = ventilacion;  
}  
  
public int getSitio() {  
    return this.sitio;  
}  
public void setSitio(int sitio) {  
    this.sitio = sitio;  
}  
public String getPlaca() {  
    return this.placa;  
}  
public void setPlaca(String placa) {  
    this.placa = placa;  
}  
public String getProcesador() {  
    return this.procesador;  
}  
public void setProcesador(String procesador) {  
    this.procesador = procesador;  
}  
public String getTarjetaGrafica() {  
    return this.tarjetaGrafica;  
}
```

Periféricos

-Atributos

```
private int sitio;  
private String monitor1;  
private String monitor2;  
private String raton;  
private String teclado;  
private String discoPortatil;
```

-Constructores

```
public Perifericos() {  
}  
  
public Perifericos(int sitio) {  
    this.sitio = sitio;  
}  
  
public Perifericos(int sitio, String monitor1, String monitor2, String raton, String teclado, String discoPortatil) {  
    this.sitio = sitio;  
    this.monitor1 = monitor1;  
    this.monitor2 = monitor2;  
    this.raton = raton;  
    this.teclado = teclado;  
    this.discoPortatil = discoPortatil;  
}
```

-ToString

```
@Override  
public String toString() {  
    return "Sitio: " + sitio + ", Monitor1: " + monitor1 + ", Monitor2: " +  
        monitor2 + ", Raton: " + raton + ", Teclado: " + teclado +  
        ", DiscoPortatil: " + discoPortatil + '}';  
}
```

-Getters y Setters

```
public int getSitio() {  
    return this.sitio;  
}  
  
public void setSitio(int sitio) {  
    this.sitio = sitio;  
}  
  
public String getMonitor1() {  
    return this.monitor1;  
}  
  
public void setMonitor1(String monitor1) {  
    this.monitor1 = monitor1;  
}  
  
public String getMonitor2() {  
    return this.monitor2;  
}  
  
public void setMonitor2(String monitor2) {  
    this.monitor2 = monitor2;  
}
```

```
public String getRaton() {  
    return this.raton;  
}  
  
public void setRaton(String raton) {  
    this.raton = raton;  
}  
  
public String getTeclado() {  
    return this.teclado;  
}  
  
public void setTeclado(String teclado) {  
    this.teclado = teclado;  
}  
  
public String getDiscoPortatil() {  
    return this.discoPortatil;  
}  
  
public void setDiscoPortatil(String discoPortatil) {  
    this.discoPortatil = discoPortatil;  
}
```