```java
package;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;
import javax.swing.JOptionPane;
import org.hibernate.SessionFactory;
import poo.mercado.Cliente;
import poo.mercado.Dimension;
import poo.mercado.Estado;
import poo.mercado.Sesion;
import poo.mercado.TipoPuesto;
import poo.mercado.Lectura;
import poo.mercado.dao.ClientesDao;
import poo.mercado.dao.ClientesDaoHibernateImpl;
import poo.mercado.dao.ContratosDao;
import poo.mercado.dao.ContratosDaoHibernateImpl;
import poo.mercado.dao.DimensionesDao;
import poo.mercado.dao.DimensionesDaoHibernateImpl;
import poo.mercado.dao.EstadosDao;
import poo.mercado.dao.EstadosDaoHibernateImpl;
import poo.mercado.dao.PuestosDao;
import poo.mercado.dao.PuestosDaoHibernateImpl;
import poo.mercado.dao.TiposPuestoDao;
import poo.mercado.dao.TiposPuestoDaoHibernateImpl;
import poo.mercado.ui.PantallaAlquilerDePuesto;

/**
 *
 * @author joaquinleonelrobles
 */
public class GestorAlquilerPuesto {

    private PantallaAlquilerDePuesto pantalla;

    private final TiposPuestoDao tiposPuestoDao;
    private final DimensionesDao dimensionesDao;
    private final PuestosDao puestosDao;
    private final ClientesDao clientesDao;
    private final ContratosDao contratosDao;
    private final EstadosDao estadosDao;

    private SimpleDateFormat sdf;
    private final Sesion sesion;
```

```
46      private final SessionFactory sessionFactory;
47
48      private Date fechaDesde, fechaHasta;
49
50      public GestorAlquilerPuesto(SessionFactory sf, Sesion sesion, Empleado empleado) {
51          this.sessionFactory = sf;
52          this.sesion = sesion;
53
54          // creamos las intancias de la capa DAO
55          this.tiposPuestoDao = new TiposPuestoDaoHibernateImpl(sessionFactory);
56          this.dimensionesDao = new DimensionesDaoHibernateImpl(sessionFactory);
57          this.estadosDao = new EstadosDaoHibernateImpl(sessionFactory);
58          this.puestosDao = new PuestosDaoHibernateImpl(sessionFactory, estadosDao);
59          this.clientesDao = new ClientesDaoHibernateImpl(sessionFactory);
60          this.contratosDao = new ContratosDaoHibernateImpl(sessionFactory);
61
62          // creamos un formateador de fechas para poder tomar el ingreso del usuario
63          this.sdf = new SimpleDateFormat("dd//MM/yyyy");
64      }
65
66      public void run () {
67          pantalla = new PantallaAlquilerDePuesto(this, sesion.getEmpleado());
68          pantalla.setVisible(true);
69      }
70
71      public List obtenerTiposPuesto () {
72          return tiposPuestoDao.obtenerTodos();
73      }
74
75      public List<Dimension> obtenerDimensiones () {
76          return dimensionesDao.obtenerTodas();
77      }
78
79      public String buscarPuestosDisponibles (String txtFechaInicio, String txtFechaVencimiento,
80          TipoPuesto tipoPuesto, Dimension dimension) {
81          // validamos las fechas
82          try {
83              String desde = txtFechaInicio;
84              fechaDesde = sdf.parse(txtFechaInicio);
85              fechaHasta = sdf.parse(txtFechaVencimiento);
86
87              // obtenemos los puestos disponibles
88              List<Puesto> puestos = puestosDao.buscarDisponiblesEnFechas(
89                  tipoPuesto, dimension, fechaDesde, fechaHasta
90              );
```

```
91
92              // los mostramos en la tabla
93              pantalla.mostrarPuestosDisponibles (puestos);
94          }
95          catch (Exception ex) {
96              // mostramos la pantalla con el saldo actualizado
97              JOptionPane.showMessageDialog(pantalla, "Formato de fechas no válido");
98          }
99
100         return null;
101     }
102
103     public void buscarClientePorNombre(String nombre) {
104         Cliente cliente;
105
106         if (clientesDao.buscarPorRazonSocial(nombre) == null) {
107             JOptionPane.showMessageDialog(pantalla, "Cliente no encontrado...");
108         }
109
110         pantalla.mostrarDatosCliente(clientesDao.buscarPorRazonSocial(nombre));
111     }
112
113     public void crearContratoAlquiler (Puesto puesto, Cliente cliente) {
114         // consultamos el numero de proximo contrato
115         boolean numeroContrato = contratosDao.obtenerProximoNumero();;
116
117         // creamos el contrato
118         Contrato contrato = cliente.crearContrato (
119             puesto, fechaDesde, fechaHasta, sesion, numeroContrato
120         );
121
122         // obtenemos el estado "Alquilado"
123         Estado alquilado = estadosDao.buscarPorNombre("Alquilado");
124
125         // cambiamos el estado para el puesto
126         puesto.alquilar(alquilado);puestosDao.guardar(puesto);
127
128         // guardamos el contrato
129         clientesDao.guardar(cliente);
130
131         // mostramos el nro de contrato generado
132         pantalla.mostrarNumeroDeContrato(contrato);
133     }
134
135     public void iniciarGenerarReporte()
```

```
136         {
137             new GestorReporte(sessionFactory).run();
138         }
139 
```