

```

1 package Persistencia;
2
3 import java.rmi.RemoteException;
4 import java.util.Date;
5 import java.util.LinkedList;
6 import java.util.List;
7 import org.hibernate.Criteria;
8 import org.hibernate.Query;
9 import org.hibernate.Session;
10 import org.hibernate.Transaction;
11 import Dominio.Cliente;
12 import Dominio.Socio;
13 import Dominio.Pago;
14 import Dominio.Reserva;
15 import Dominio.Servicio;
16 import Dominio.Sesion;
17 import Dominio.Socio;
18 import Dominio.Usuario;
19 import java.util.Enumeration;
20 import java.util.Hashtable;
21 import Interfaz.IJFramePr;
22
23 public class AgenteBD{
24
25     private static AgenteBD yo=null;
26     private static UtilidadesHibernate utilidadesH=null;//La forma de conectarse
    con la BD
27     private static Session sesion=null;//La sesion utilizada para conectarse con la
    BD
28     private static Hashtable <Integer,IJFramePr> sesiones=new Hashtable
    <Integer,IJFramePr>();//todas las sesiones abiertas
29     private boolean cerrando=false;//si estamos cerrando las sesiones
30
31     public AgenteBD() throws RemoteException {
32         super();
33         try {
34             utilidadesH = new UtilidadesHibernate();
35             sesion = utilidadesH.getSesion();
36         } catch (Exception e) {
37             e.printStackTrace();
38         }
39     }
40
41     public static AgenteBD obtenerAgente()throws RemoteException {
42         if(yo==null){
43             try {
44                 yo=new AgenteBD();
45             } catch (RemoteException e) {
46                 e.printStackTrace();
47             }
48         }
49         return yo;
50     }
51
52     private void notificar(){
53         Enumeration <IJFramePr> i=sesiones.elements();
54         while( i.hasMoreElements()) {
55             try {
56                 i.nextElement().actualizarTablas();
57             } catch (RemoteException e) {
58                 e.printStackTrace();
59             }

```

```

60     }
61 }
62
63 public List<Cliente> consultarClientesBD()throws RemoteException {
64     List<Cliente> clientes=null;
65     Criteria crit = sesion.createCriteria(Cliente.class);
66     clientes = crit.list();
67     return clientes;
68 }
69
70 public List<Cliente> consultarClientesSociosBD() throws RemoteException {
71     List<Cliente> lc=null;
72     Query query = sesion.createQuery("from Cliente where esSocio = 1");
73     lc = query.list();
74     return lc;
75 }
76 public List<Cliente> consultarClientesNoSociosBD() throws RemoteException {
77     List<Cliente> lc=null;
78     Query query = sesion.createQuery("from Cliente where esSocio = 0");
79     lc = query.list();
80     return lc;
81 }
82
83 public Cliente consultarClienteBD(int dni)throws RemoteException {
84     Cliente c=null;
85     try {
86         c=(Cliente) sesion.load(Cliente.class, dni);
87     } catch (Exception e) {
88         e.printStackTrace();
89     }
90     return c;
91 }
92
93 public int crearClienteBD(Cliente c)throws RemoteException {
94     int res=0;
95     try {
96         sesion.save(c);
97         sesion.flush();
98     } catch (Exception e) {
99         e.printStackTrace();
100         res=-1;
101     }
102     notificar();
103     return res;
104 }
105
106 public int modificarClienteBD(Cliente c)throws RemoteException {
107     int res=0;
108     try {
109         sesion.merge(c);
110         sesion.flush();
111     } catch (Exception e) {
112         e.printStackTrace();
113         res=-1;
114     }
115     notificar();
116     return res;
117 }
118 public int borrarClienteBD(int dni)throws RemoteException {
119     int res=0;
120     try {
121         Cliente c=(Cliente) sesion.load(Cliente.class, dni);

```

```

122         session.delete(c);
123         session.flush();
124     } catch (Exception e) {
125         e.printStackTrace();
126         res=-1;
127     }
128     notificar();
129     return res;
130 }
131
132 public int crearPagoBD(Pago p)throws RemoteException {
133     int res=0;
134     try {
135         session.save(p);
136         session.flush();
137     } catch (Exception e) {
138         e.printStackTrace();
139         res=-1;
140     }
141     notificar();
142     return res;
143 }
144 public int modificarPagoBD(Pago p)throws RemoteException {
145     int res=0;
146     try {
147         session.merge(p);
148         session.flush();
149     } catch (Exception e) {
150         e.printStackTrace();
151         res=-1;
152     }
153     notificar();
154     return res;
155 }
156
157 public int borrarPagoBD(int dni)throws RemoteException {
158     int res=0;
159     try {
160         Cliente c=(Cliente) session.load(Cliente.class, dni);
161         session.delete(c);
162         session.flush();
163     } catch (Exception e) {
164         e.printStackTrace();
165         res=-1;
166     }
167     notificar();
168     return res;
169 }
170
171 public Pago consultarPagoBD(int id)throws RemoteException {
172     Pago p = null;
173     try {
174         p=(Pago) session.load(Pago.class, id);
175     } catch (Exception e) {
176         e.printStackTrace();
177     }
178     return p;
179 }
180
181 public List<Pago> consultarPagosClienteBD(int dni)throws RemoteException {
182     List<Pago> pagos=null;
183     Query query = session.createQuery("from Pago where cliente = '"+dni+"'");

```

```

184         pagos = query.list();
185         return pagos;
186     }
187     public List<Pago> consultarPagosClientePendientesBD(int id) {
188         List<Pago> pagos=null;
189         Query query = sesion.createQuery("from Pago where cliente = '"+id+"' and
liquidadado = 0");
190         pagos = query.list();
191         return pagos;
192     }
193     public List<Pago> consultarPagosBD()throws RemoteException {
194         List<Pago> pagos=null;
195         Criteria crit = sesion.createCriteria(Pago.class);
196         pagos = crit.list();
197         return pagos;
198     }
199
200     public List<Pago> consultarPagosFechasBD(Date fecha_ini,Date fecha_fin)throws
RemoteException {
201         List<Pago> pagos=null;
202         java.sql.Date ini=new java.sql.Date (fecha_ini.getTime());
203         java.sql.Date fin=new java.sql.Date (fecha_fin.getTime());
204         Query query = sesion.createQuery("from Pago where fecha >=
 '"+ini.toString()+"' and fecha <= '"+fin.toString()+"'");
205         pagos = query.list();
206         return pagos;
207     }
208     public List<Pago> consultarPagosClienteFechasBD(int dni,Date fecha_ini,Date
fecha_fin)throws RemoteException {
209         List<Pago> pagos=null;
210         java.sql.Date ini=new java.sql.Date (fecha_ini.getTime());
211         java.sql.Date fin=new java.sql.Date (fecha_fin.getTime());
212         Query query = sesion.createQuery("from Pago where cliente="+dni+" and fecha
>= '"+ini.toString()+"' and fecha <= '"+fin.toString()+"'");
213         pagos = query.list();
214         return pagos;
215     }
216
217     public Reserva consultarReservaBD(int id)throws RemoteException {
218         Reserva r=null;
219         try {
220             r=(Reserva) sesion.load(Reserva.class, id);
221         } catch (Exception e) {
222             e.printStackTrace();
223         }
224         return r;
225     }
226
227     public List<Reserva> consultarReservasFechaBD(int idServicio,Date fecha)throws
RemoteException {
228         List<Reserva> lr=null;
229         java.sql.Date f=new java.sql.Date (fecha.getTime());
230         Query query = sesion.createQuery("from Reserva where
servicio="+idServicio+" and fechaReserva = '"+f.toString()+"' ORDER BY
horaReserva");
231         lr = query.list();
232         return lr;
233     }
234
235     public List<Reserva> consultarReservaFechaHoraServicioBD(int idServicio,Date
fecha,int hora)throws RemoteException {
236         List<Reserva> lr=null;

```

```

237         java.sql.Date f=new java.sql.Date (fecha.getTime());
238         Query query = sesion.createQuery("from Reserva where
servicio="+idServicio+" and fechaReserva = '"+f.toString()+"' and horaReserva =
'"+hora+"'");
239         lr = query.list();
240         return lr;
241     }
242
243     public int anularReservaBD(int reserva) throws RemoteException {
244         int res=0;
245         Transaction tx = sesion.beginTransaction();
246         try {
247             Reserva r=consultarReservaBD(reserva);
248             Servicio s=consultarServicioBD(r.getServicio());
249             Cliente c= consultarClienteBD(r.getCliente());
250             int tipoPago=1;
251             int liquidado=1;
252             Date fr =r.getFechaReserva();
253             Date fhoy=new Date();
254             if(fr.compareTo(fhoy)>-1){
255                 if(c.getEsSocio()==1){
256                     tipoPago=2;
257                     liquidado=0;
258                 }
259                 Pago p =new Pago(-s.getPrecioHora(),r.getCliente(),"Anulacion
reserva",r.getUsuario(),tipoPago,liquidado);
260                 sesion.save(p);
261                 sesion.delete(r);
262             }
263             else{
264                 res=-1;
265             }
266             tx.commit();
267         } catch (Exception e){
268             e.printStackTrace();
269             res=-1;
270             tx.rollback();
271         }
272         notificar();
273         return res;
274     }
275
276     public List<Reserva> consultarReservasServicioBD(int id) throws
RemoteException {
277         List<Reserva> lr=null;
278         Query query = sesion.createQuery("from Reserva where servicio="+id);
279         lr = query.list();
280         return lr;
281     }
282
283     public int crearReserva(int servicio, int usuario, int cliente,
Date fechaReserva, int horaReserva) throws RemoteException {
284         int res=0;
285         Transaction tx = sesion.beginTransaction();
286         try {
287             Servicio s=consultarServicioBD(servicio);
288             if(s.getHabilitado()==1){
289                 List<Reserva> lr =consultarReservaFechaHoraServicioBD(servicio,
fechaReserva, horaReserva);
290                 if (lr.size()==0) {
291                     Cliente c=consultarClienteBD(cliente);
292                     int tipoPago=0,liquidado=0;

```

AgenteBD.java

```

294         if(c.getEsSocio()==1){
295             tipoPago=2;
296             liquidado=0;
297         }else{
298             tipoPago=1;
299             liquidado=1;
300         }
301         Pago p=new
302         Pago(s.getPrecioHora(),cliente,"Reserva",usuario,tipoPago,liquidado);
303         session.save(p);
304         Reserva r =new
305         Reserva(servicio,usuario,cliente,fechaReserva,horaReserva,p.getId());
306         session.save(r);
307     }
308     else res=-1;
309 }
310 else{
311     res=-1;
312 }
313 tx.commit();
314 }
315 catch (Exception e) {
316     e.printStackTrace();
317     res=-1;
318     tx.rollback();
319 }
320 notificar();
321 return res;
322 }
323
324 public Servicio consultarServicioBD(int id)throws RemoteException {
325     Servicio s=null;
326     try {
327         s=(Servicio) session.load(Servicio.class, id);
328     } catch (Exception e) {
329         e.printStackTrace();
330     }
331     return s;
332 }
333
334 public List<Servicio> consultarServiciosBD()throws RemoteException {
335     List<Servicio> servicios=null;
336     Criteria crit = session.createCriteria(Servicio.class);
337     servicios = crit.list();
338     return servicios;
339 }
340
341 public int crearServicioBD(Servicio s)throws RemoteException {
342     int res=0;
343     try {
344         session.save(s);
345         session.flush();
346     } catch (Exception e) {
347         e.printStackTrace();
348         res=-1;
349     }
350     notificar();
351     return res;
352 }
353
354 public int modificarServicioBD(Servicio s)throws RemoteException {
355     int res=0;
356     try {
357         session.merge(s);

```

```

354         session.flush();
355     } catch (Exception e) {
356         e.printStackTrace();
357         res=-1;
358     }
359     notificar();
360     return res;
361 }
362
363 public int borrarServicioBD(int id)throws RemoteException {
364     int res=0;
365     Servicio s;
366     try {
367         s=(Servicio) session.load(Servicio.class, id);
368         session.delete(s);
369         session.flush();
370     } catch (Exception e) {
371         e.printStackTrace();
372         res=-1;
373     }
374     notificar();
375     return res;
376 }
377
378 public Sesion consultarSesionBD(int dni)throws RemoteException {
379     Sesion s=null;
380     try {
381         s=(Sesion) session.load(Sesion.class, dni);
382     } catch (Exception e) {
383         e.printStackTrace();
384     }
385     return s;
386 }
387
388 public int crearSesionBD(Sesion s)throws RemoteException {
389     int res=0;
390     try {
391         session.save(s);
392         session.flush();
393     } catch (Exception e) {
394         e.printStackTrace();
395         res=-1;
396     }
397     notificar();
398     return res;
399 }
400
401 public int borrarSesionBD(int dni)throws RemoteException {
402     int res=0;
403     Sesion s;
404     try {
405         s=(Sesion) session.load(Sesion.class, dni);
406         session.delete(s);
407         session.flush();
408     } catch (Exception e) {
409         e.printStackTrace();
410         res=-1;
411     }
412     notificar();
413     return res;
414 }
415

```

```

416     public List<Sesion> consultarSesionesBD()throws RemoteException {
417         List<Sesion> sesiones=null;
418         Criteria crit = sesion.createCriteria(Sesion.class);
419         sesiones = crit.list();
420         return sesiones;
421     }
422
423     public List<Socio> consultarSociosBD()throws RemoteException {
424         List<Socio> ls=null;
425         Criteria crit = sesion.createCriteria(Socio.class);
426         ls = crit.list();
427         return ls;
428     }
429     public List<Socio> consultarSociosTitularesBD()throws RemoteException {
430         List<Socio> lc=null;
431         Query query = sesion.createQuery("from Socio where dni=dniTitular ");
432         lc = query.list();
433         return lc;
434     }
435
436     public Socio consultarSocioBD(int dni)throws RemoteException {
437         Socio s=null;
438         try {
439             s=(Socio) sesion.load(Socio.class, dni);
440         } catch (Exception e) {
441             e.printStackTrace();
442         }
443         return s;
444     }
445
446     public List<Socio> consultarSociosGrupoBD(int id)throws RemoteException {
447         List<Socio> ls=null;
448         Query query = sesion.createQuery("from Socio where dniTitular="+id);
449         ls = query.list();
450         return ls;
451     }
452
453     public LinkedList<Cliente> consultarClientesGrupoBD(int id)throws
RemoteException {
454         LinkedList<Cliente> lc=new LinkedList();
455         List<Socio> ls=consultarSociosGrupoBD(id);
456         for (int i = 0; i < ls.size(); i++) {
457             lc.add(consultarClienteBD(ls.get(i).getDni()));
458         }
459         return lc;
460     }
461
462     public int crearSocioBD(Socio s)throws RemoteException {
463         int res=0;
464         Transaction tx = sesion.beginTransaction();
465         try {
466             Cliente c = consultarClienteBD(s.getDni());
467             if(c!=null){
468                 c.setEsSocio(1);
469                 sesion.merge(c);
470             }else{
471                 return -1;
472             }
473             sesion.save(s);
474             tx.commit();
475         } catch (Exception e) {
476             e.printStackTrace();

```



```

477         res=-1;
478         tx.rollback();
479     }
480     notificar();
481     return res;
482 }
483
484 public int modificarSocioBD(Socio s)throws RemoteException {
485     int res=0;
486     try {
487         session.merge(s);
488         session.flush();
489     } catch (Exception e) {
490         e.printStackTrace();
491         res=-1;
492     }
493     notificar();
494     return res;
495 }
496
497 public int borrarSocioBD(int dni)throws RemoteException {
498     int res=0;
499     Socio s;
500     try {
501         s=(Socio) session.load(Socio.class, dni);
502         session.delete(s);
503         session.flush();
504     } catch (Exception e) {
505         e.printStackTrace();
506         res=-1;
507     }
508     notificar();
509     return res;
510 }
511
512 public int darBajaSocioBD(int id)throws RemoteException {
513     int res=0;
514     Transaction tx = session.beginTransaction();
515     try {
516         Socio s =consultarSocioBD(id);
517         if(s.getDni()==s.getDniTitular()){
518             List<Socio> ls=consultarSociosGrupoBD(id);
519             for (int i = 0; i < ls.size(); i++) {
520                 List<Pago>
lp=consultarPagosClientePendientesBD(ls.get(i).getDni());
521                 for (int j = 0; j < lp.size(); j++) {
522                     Pago p=lp.get(j);
523                     p.setLiquidado(1);
524                     session.merge(p);
525                 }
526                 Cliente c=consultarClienteBD(ls.get(i).getDni());
527                 c.setEsSocio(0);
528                 session.merge(c);
529                 session.delete(ls.get(i));
530             }
531         }
532     } else{
533         List<Pago> lp=consultarPagosClientePendientesBD(s.getDni());
534         for (int j = 0; j < lp.size(); j++) {
535             Pago p=lp.get(j);
536             p.setLiquidado(1);
537             session.merge(p);

```

```

538         }
539         Cliente c=consultarClienteBD(id);
540         c.setEsSocio(0);
541         session.merge(c);
542         session.delete(s);
543     }
544     tx.commit();
545 }
546 catch (Exception e) {
547     e.printStackTrace();
548     res=-1;
549     tx.rollback();
550 }
551 notificar();
552 return res;
553 }
554
555 public Usuario consultarUsuarioBD(int dni)throws RemoteException {
556     Usuario u=null;
557     try {
558         u = (Usuario) session.load(Usuario.class, dni);
559     } catch (Exception e) {
560         e.printStackTrace();
561     }
562     return u;
563 }
564 public List<Usuario> consultarUsuariosBD()throws RemoteException {
565     List<Usuario> usuarios=null;
566     Criteria crit = session.createCriteria(Usuario.class);
567     usuarios = crit.list();
568     return usuarios;
569 }
570
571 public int crearUsuarioBD(Usuario u)throws RemoteException {
572     int res=0;
573     try {
574         session.save(u);
575         session.flush();
576     } catch (Exception e) {
577         e.printStackTrace();
578         res=-1;
579     }
580     notificar();
581     return res;
582 }
583 public int modificarUsuarioBD(Usuario u)throws RemoteException {
584     int res=0;
585     try {
586         session.merge(u);
587         session.flush();
588     } catch (Exception e) {
589         e.printStackTrace();
590         res=-1;
591     }
592     notificar();
593     return res;
594 }
595
596 public int borrarUsuarioBD(int dni)throws RemoteException {
597     int res=0;
598     Usuario u;
599     try {

```

AgenteBD.java

```
600         u=(Usuario) session.load(Usuario.class, dni);
601         session.delete(u);
602         session.flush();
603     } catch (Exception e) {
604         e.printStackTrace();
605         res=-1;
606     }
607     notificar();
608     return res;
609 }
610 }
```