

DBAgente.java

```

1 package Persistenza;
2 import java.rmi.RemoteException;
22
23 public class DBAgente{
24
25     private static DBAgente me=null;
26     private static HibernateUtil hibernateUtil=null; //HibernateUtil = Classe che
        utilizza una libreria di "Ibernazione " per collegare il sistema con la base dati.
27     private static Session sessione=null;
28     private static Hashtable <Integer,IJFramePr> sessioni=new Hashtable
        <Integer,IJFramePr>();
29     private boolean chiusura=false;
30     public DBAgente() throws RemoteException {
31         super();
32         try {
33             hibernateUtil = new HibernateUtil();
34             sessione = hibernateUtil.getSession();
35         } catch (Exception e) {
36             e.printStackTrace();
37         }
38     }
39
40     public static DBAgente getAgente()throws RemoteException {
41         if(me==null){
42             try {
43                 me=new DBAgente();
44
45             } catch (RemoteException e) {
46                 System.out.println("Errore durante la creazione DBAgente");
47                 e.printStackTrace();
48             }
49         }
50         return me;
51     }
52
53     private void notificare(){
54         Enumeration <IJFramePr> i=sessioni.elements();
55         while( i.hasMoreElements()) {
56             try {
57                 i.nextElement().aggiornareTabella();
58             } catch (RemoteException e) {
59                 e.printStackTrace();
60             }
61         }
62     }
63
64     public List<Cliente> consultareClientiDB()throws RemoteException {
65         List<Cliente> clienti=null;
66         Criteria crit = sessione.createCriteria(Cliente.class);
67         clienti = crit.list();
68         return clienti;
69     }
70
71     public List<Cliente> consultareClientiSociDB() throws RemoteException {
72         List<Cliente> lc=null;
73
74         Query query = sessione.createQuery("from Cliente where eSocio = 1");
75         lc = query.list();
76         return lc;
77     }
78     public List<Cliente> consultareClientiNoSocioDB() throws RemoteException {
79         List<Cliente> lc=null;

```

DBAgente.java

```

80     Query query = sessione.createQuery("from Cliente where eSocio = 0");
81     lc = query.list();
82     return lc;
83 }
84
85 //returns 0 if customer is on DB and -1 if not
86 public Cliente consultareClienteDB(int dni)throws RemoteException {
87     Cliente c=null;
88     try {
89         c=(Cliente) sessione.load(Cliente.class, dni);
90     } catch (Exception e) {
91         System.err.println("ERRORE");
92         e.printStackTrace();
93     }
94     return c;
95 }
96
97 public int creareClienteDB(Cliente c)throws RemoteException {
98     int res=0;
99     try {
100         sessione.save(c);
101         sessione.flush();
102     } catch (Exception e) {
103         System.err.println("ERRORE");
104         e.printStackTrace();
105         res=-1;
106     }
107     notificare();
108     return res;
109 }
110
111 public int modificareClienteDB(Cliente c)throws RemoteException {
112     int res=0;
113     try {
114         sessione.merge(c);
115         sessione.flush();
116     } catch (Exception e) {
117         System.err.println("ERRORE");
118         e.printStackTrace();
119         res=-1;
120     }
121     notificare();
122     return res;
123 }
124
125 public int cancellareClienteDB(int id)throws RemoteException {
126     int res=0;
127     try {
128         Cliente c=(Cliente) sessione.load(Cliente.class, id);
129         sessione.delete(c);
130         sessione.flush();
131     } catch (Exception e) {
132         System.err.println("ERRORE");
133         e.printStackTrace();
134         res=-1;
135     }
136     notificare();
137     return res;
138 }
139
140
141 public int crearePagamentoDB(Pagamento p)throws RemoteException {

```

DBAgente.java

```

142     int res=0;
143     try {
144         sessione.save(p);
145         sessione.flush();
146     } catch (Exception e) {
147         System.err.println("ERRORE");
148         e.printStackTrace();
149         res=-1;
150     }
151     notificare();
152     return res;
153 }
154
155 public int modificarePagamentoDB(Pagamento p) throws RemoteException {
156     int res=0;
157     try {
158         sessione.merge(p);
159         sessione.flush();
160     } catch (Exception e) {
161         System.err.println("ERRORE");
162         e.printStackTrace();
163         res=-1;
164     }
165     notificare();
166     return res;
167 }
168
169 public int eliminarePagamentoDB(int dni) throws RemoteException {
170     int res=0;
171     try {
172         Cliente c=(Cliente) sessione.load(Cliente.class, dni);
173         sessione.delete(c);
174         sessione.flush();
175     } catch (Exception e) {
176         System.err.println("ERRORE");
177         e.printStackTrace();
178         res=-1;
179     }
180     notificare();
181     return res;
182 }
183
184 public Pagamento consultarePagamentoDB(int id) throws RemoteException {
185     Pagamento p = null;
186     try {
187         p=(Pagamento) sessione.load(Pagamento.class, id);
188     } catch (Exception e) {
189         System.err.println("ERRORE");
190         e.printStackTrace();
191     }
192     return p;
193 }
194
195 public List<Pagamento> consultarePagamentiClienteDB(int id) throws
RemoteException {
196     List<Pagamento> pagos=null;
197     Query query = sessione.createQuery("from Pagamento where cliente =
198     '"+id+"'");
199     pagos = query.list();
200     return pagos;
201 }

```

DBAgente.java

```

202     public List<Pagamento> cercareInAttesaPagamentiClienteDB(int id) {
203         List<Pagamento> pagamenti=null;
204         Query query = sessione.createQuery("from Pagamento where cliente = '"+id+"'
and etinto = 0");
205         pagamenti = query.list();
206         return pagamenti;
207     }
208
209     public List<Pagamento> consultarePagamentiDB()throws RemoteException {
210         List<Pagamento> pagamenti=null;
211         Criteria crit = sessione.createCriteria(Pagamento.class);
212         pagamenti = crit.list();
213         return pagamenti;
214     }
215
216     public List<Pagamento> consultarePagamentiClienteDB(Date startDate,Date
endDate)throws RemoteException {
217         List<Pagamento> pagamenti=null;
218         java.sql.Date s=new java.sql.Date (startDate.getTime());
219         java.sql.Date e=new java.sql.Date (endDate.getTime());
220         Query query = sessione.createQuery("from Pagamento where data >=
 '"+s.toString()+"' and data <= '"+e.toString()+"'");
221         pagamenti = query.list();
222         return pagamenti;
223     }
224     public List<Pagamento> consultarePagamentiDateClienteDB(int id,Date
startDate,Date endDate)throws RemoteException {
225         List<Pagamento> pagamenti=null;
226         java.sql.Date s=new java.sql.Date (startDate.getTime());
227         java.sql.Date e=new java.sql.Date (endDate.getTime());
228         Query query = sessione.createQuery("from Pagamento where cliente="+id+" and
data >= '"+s.toString()+"' and data <= '"+e.toString()+"'");
229         pagamenti = query.list();
230         return pagamenti;
231     }
232
233     public Prenotazione cercarePrenotazioniDB(int id)throws RemoteException {
234         Prenotazione r=null;
235         try {
236             r=(Prenotazione) sessione.load(Prenotazione.class, id);
237         } catch (Exception e) {
238             System.err.println("ERRORE");
239             e.printStackTrace();
240         }
241         return r;
242     }
243
244     public List<Prenotazione> consultarePrenotazioneDateDB(int ids,Date
data)throws RemoteException {
245         List<Prenotazione> lr=null;
246         java.sql.Date f=new java.sql.Date (data.getTime());
247         Query query = sessione.createQuery("from Prenotazione where
servizio="+ids+" and dataPrenotazione = '"+f.toString()+"' ORDER BY
oraPrenotazione");
248         lr = query.list();
249         return lr;
250     }
251
252     public List<Prenotazione> consultarePrenotazioniDataHourClienteDB(int ids,Date
data,int our)throws RemoteException {
253         List<Prenotazione> lr=null;
254         java.sql.Date f=new java.sql.Date (data.getTime());

```

DBAgente.java

```

255         Query query = sessione.createQuery("from Prenotazione where
servizio="+ids+" and dataPrenotazione = '"+f.toString()+"' and oraPrenotazione =
"+"our+'");
256         lr = query.list();
257         return lr;
258     }
259
260     public int annullaPrenotazioneDB(int id) throws RemoteException {
261         int res=0;
262         Transaction tx = sessione.beginTransaction();
263         try {
264
265             Prenotazione r=cercarePrenotazioniDB(id);
266             Servizio s=inabilitareServizioDB(r.getServizio());
267             Cliente c= consultareClienteDB(r.getId());
268             int t=1;
269             int l=1;
270             Date fr =r.getDataPrenotazione();
271             Date attuale=new Date();
272             if(fr.compareTo(attuale)>-1){
273                 if(c.getEsocio()==1){
274                     t=2;
275                     l=0;
276                 }
277                 Pagamento p =new
Pagamento(-s.getPrezzoOra(),r.getCliente(),"cancellare",r.getUtente(),t,l);
278                 sessione.save(p);
279                 sessione.delete(r);
280             }
281             else{
282                 res=-1;
283             }
284             tx.commit();
285         } catch (Exception e){
286             System.err.println("ERRORE");
287             e.printStackTrace();
288             res=-1;
289             tx.rollback();
290         }
291         notificare();
292         return res;
293     }
294
295     public List<Prenotazione> consultarePrenotazioneServizio(int id) throws
RemoteException {
296         List<Prenotazione> lr=null;
297         Query query = sessione.createQuery("from Prenotazione where servizio="+id);
298         lr = query.list();
299         return lr;
300     }
301
302     public int crearePrenotazioneDB(int se, int u, int cu, Date dr, int hr) throws
RemoteException {
303         int res=0;
304         Transaction tx = sessione.beginTransaction();
305         try {
306             Servizio s=inabilitareServizioDB(se);
307             if(s.getAbilitato()==1){
308                 List<Prenotazione> lr =consultarePrenotazioniDataHourClienteDB(se,
dr, hr);
309                 if (lr.size()==0) { //Is available
310                     Cliente c=consultareClienteDB(cu);

```

DBAgente.java

```

311         int tPagamento=0,pagato=0;
312         if(c.getEsocio()==1){
313             tPagamento=2;
314             pagato=0;
315         }else{
316             tPagamento=1;
317             pagato=1;
318         }
319         Pagamento p=new
Pagamento(s.getPrezzoOra(),cu,"Reservation",u,tPagamento,pagato);
320         sessione.save(p);
321         Prenotazione r =new Prenotazione(se,u,cu,dr,hr,p.getId());
322         sessione.save(r);
323     }
324     else res=-1;
325
326 }
327 else{
328     res=-1;
329 }
330 tx.commit();
331 }
332 catch (Exception e) {
333     System.err.println("ERRORE");
334     e.printStackTrace();
335     res=-1;
336     tx.rollback();
337 }
338 notificare();
339 return res;
340 }
341
342 public Servizio inabilitareServizioDB(int id)throws RemoteException {
343     Servizio s=null;
344     try {
345         s=(Servizio) sessione.load(Servizio.class, id);
346     } catch (Exception e) {
347         System.err.println("ERRORE");
348         e.printStackTrace();
349     }
350     return s;
351 }
352
353 public List<Servizio> consultareServiziDB()throws RemoteException {
354     List<Servizio> services=null;
355     Criteria crit = sessione.createCriteria(Servizio.class);
356     services = crit.list();
357     return services;
358 }
359
360 public int creareServizioDB(Servizio s)throws RemoteException {
361     int res=0;
362     try {
363         sessione.save(s);
364         sessione.flush();
365     } catch (Exception e) {
366         System.err.println("ERRORE");
367         e.printStackTrace();
368         res=-1;
369     }
370     notificare();
371     return res;

```

DBAgente.java

```

372     }
373
374     public int modificareServizioDB(Servizio s) throws RemoteException {
375         int res=0;
376         try {
377             sessione.merge(s);
378             sessione.flush();
379         } catch (Exception e) {
380             System.err.println("ERRORE");
381             e.printStackTrace();
382             res=-1;
383         }
384         notificare();
385         return res;
386     }
387
388     public int eliminareServizioDB(int id) throws RemoteException {
389         int res=0;
390         Servizio s;
391         try {
392             s=(Servizio) sessione.load(Servizio.class, id);
393             sessione.delete(s);
394             sessione.flush();
395         } catch (Exception e) {
396             System.err.println("ERRORE");
397             e.printStackTrace();
398             res=-1;
399         }
400         notificare();
401         return res;
402     }
403
404     public Dominio.Sessione consultareSessioniDB(int dni) throws RemoteException {
405         Dominio.Sessione s=null;
406         try {
407             s=(Dominio.Sessione) sessione.load(Dominio.Sessione.class, dni);
408         } catch (Exception e) {
409             System.err.println("ERRORE");
410             e.printStackTrace();
411         }
412         return s;
413     }
414
415     public int creareSessioneDB(Dominio.Sessione s) throws RemoteException {
416         int res=0;
417         try {
418             sessione.save(s);
419             sessione.flush();
420         } catch (Exception e) {
421             System.err.println("ERRORE");
422             e.printStackTrace();
423             res=-1;
424         }
425         notificare();
426         return res;
427     }
428
429     public int eliminareSessioneDB(int dni) throws RemoteException {
430         int res=0;
431         Session s;
432         try {
433             s=(Session) sessione.load(Session.class, dni);

```

DBAgente.java

```

434         sessione.delete(s);
435         sessione.flush();
436     } catch (Exception e) {
437         System.err.println("ERRORE");
438         e.printStackTrace();
439         res=-1;
440     }
441     notificare();
442     return res;
443 }
444
445 public List<Dominio.Sessione> consultareSessioniDB()throws RemoteException {
446     List<Dominio.Sessione> sessions=null;
447     Criteria crit = sessione.createCriteria(Dominio.Sessione.class);
448     sessions = crit.list();
449     return sessions;
450 }
451
452 public List<Socio> consultareSociDB()throws RemoteException {
453     List<Socio> ls=null;
454     Criteria crit = sessione.createCriteria(Socio.class);
455     ls = crit.list();
456     return ls;
457 }
458
459 public List<Socio> consultareSociTitolariDB()throws RemoteException {
460     List<Socio> lc=null;
461
462     Query query = sessione.createQuery("from Socio where
id=cartaIdentitaTitolare ");
463     lc = query.list();
464     return lc;
465 }
466
467 public Socio consultareSocioDB(int dni)throws RemoteException {
468     Socio s=null;
469     try {
470         s=(Socio) sessione.load(Socio.class, dni);
471     } catch (Exception e) {
472         System.err.println("ERRORE");
473         e.printStackTrace();
474     }
475     return s;
476 }
477
478 public List<Socio> consultareSociGruppoDB(int id)throws RemoteException {
479     List<Socio> ls=null;
480     Query query = sessione.createQuery("from Socio where
cartaIdentitaTitolare="+id);
481     ls = query.list();
482     return ls;
483 }
484
485 public LinkedList<Cliente> consultareClienteGruppoDB(int id)throws
RemoteException {
486     LinkedList<Cliente> lc=new LinkedList();
487     List<Socio> ls=consultareSociGruppoDB(id);
488     for (int i = 0; i < ls.size(); i++) {
489         lc.add(consultareClienteDB(ls.get(i).getCartaIdentita()));
490     }
491     return lc;
492 }

```


DBAgente.java

```

493
494 public int creareSocioDB(Socio s)throws RemoteException {
495     int res=0;
496     Transaction tx = sessione.beginTransaction();
497     try {
498         Cliente c = consultareClienteDB(s.getCartaIdentita());
499         if(c!=null){
500             c.setEsocio(1);
501             sessione.merge(c);
502         }else{
503             System.err.println("ERRORE");
504             return -1;
505         }
506         sessione.save(s);
507         tx.commit();
508     } catch (Exception e) {
509         System.err.println("ERRORE");
510         e.printStackTrace();
511         res=-1;
512         tx.rollback();
513     }
514     notificare();
515     return res;
516 }
517
518 public int modificareSocioDB(Socio s)throws RemoteException {
519     int res=0;
520     try {
521         sessione.merge(s);
522         sessione.flush();
523     } catch (Exception e) {
524         System.err.println("ERRORE");
525         e.printStackTrace();
526         res=-1;
527     }
528     notificare();
529     return res;
530 }
531
532 public int deleteMemberDB(int dni)throws RemoteException {
533     int res=0;
534     Socio s;
535     try {
536         s=(Socio) sessione.load(Socio.class, dni);
537         sessione.delete(s);
538         sessione.flush();
539     } catch (Exception e) {
540         System.err.println("ERRORE");
541         e.printStackTrace();
542         res=-1;
543     }
544     notificare();
545     return res;
546 }
547
548 public int cancellareSocioDB(int id)throws RemoteException {
549     int res=0;
550     Transaction tx = sessione.beginTransaction();
551     try {
552         Socio s =consultareSocioDB(id);
553         if(s.getCartaIdentita()==s.getCartaIdentitaTitolare()){
554             List<Socio> ls=consultareSociGruppoDB(id);

```

DBAgente.java

```

555         for (int i = 0; i < ls.size(); i++) {
556             List<Pagamento>
lp=cercareInAttesaPagamentiClienteDB(ls.get(i).getCartaIdentita());
557             for (int j = 0; j < lp.size(); j++) {
558                 Pagamento p=lp.get(j);
559                 p.setEstinto(1);
560                 sessione.merge(p);
561             }
562             Cliente c=consultareClienteDB(ls.get(i).getCartaIdentita());
563             c.setEsocio(0);
564             sessione.merge(c);
565             sessione.delete(ls.get(i));
566         }
567     }
568     else{
569         List<Pagamento>
lp=cercareInAttesaPagamentiClienteDB(s.getCartaIdentita());
570         for (int j = 0; j < lp.size(); j++) {
571             Pagamento p=lp.get(j);
572             p.setEstinto(1);
573             sessione.merge(p);
574         }
575         Cliente c=consultareClienteDB(id);
576         c.setEsocio(0);
577         sessione.merge(c);
578         sessione.delete(s);
579     }
580     tx.commit();
581 }
582 catch (Exception e) {
583     System.err.println("ERRORE");
584     e.printStackTrace();
585     res=-1;
586     tx.rollback();
587 }
588 notificare();
589 return res;
590 }
591
592 public Utente autenticareUtenteDB(int dni)throws RemoteException {
593     Utente u=null;
594     try {
595         u = (Utente) sessione.load(Utente.class, dni);
596     } catch (Exception e) {
597         System.err.println("ERRORE");
598         e.printStackTrace();
599     }
600     return u;
601 }
602
603 public List<Utente> consultareUtentiDB()throws RemoteException {
604     List<Utente> usuarios=null;
605     Criteria crit = sessione.createCriteria(Utente.class);
606     usuarios = crit.list();
607     return usuarios;
608 }
609
610 public int creareUtenteDB(Utente u)throws RemoteException {
611     int res=0;
612     try {
613         sessione.save(u);
614         sessione.flush();

```

DBAgente.java

```

615     } catch (Exception e) {
616         System.err.println("ERRORE");
617         e.printStackTrace();
618         res=-1;
619     }
620     notificare();
621     return res;
622 }
623
624 public int modificareUtenteDB(Utente u) throws RemoteException {
625     int res=0;
626     try {
627         sessione.merge(u);
628         sessione.flush();
629     } catch (Exception e) {
630         System.err.println("ERRORE");
631         e.printStackTrace();
632         res=-1;
633     }
634     notificare();
635     return res;
636 }
637
638 public int eliminareUtenteDB(int dni) throws RemoteException {
639     int res=0;
640     Utente u;
641     try {
642         u=(Utente) sessione.load(Utente.class, dni);
643         sessione.delete(u);
644         sessione.flush();
645     } catch (Exception e) {
646         System.err.println("ERRORE");
647         e.printStackTrace();
648         res=-1;
649     }
650     notificare();
651     return res;
652 }
653 }

```