

dao.InterventionDaoJpa

```
1
2 package dao;
3 import javax.persistence.EntityManager;
4 import metier.modele.Intervention;
5 /**
6  *
7  * @author Chanèle Jourdan, Jorge Terreu, Corentin laharotte
8  * Classe qui gère la persistance des interventions
9  *
10 */
11 public class InterventionDaoJpa{
12
13     //persiste l'intervention placée en paramètre
14     public static void creerIntervention(Intervention i){
15         EntityManager em=JpaUtil.obtenirEntityManager();
16         try {
17             em.persist(i);
18         } catch(Exception e) {
19         }
20     }
21
22     //mise à jour de l'intervention
23     //Pour le cas où on change son statut
24     public static void modifierIntervention(Intervention i){
25         EntityManager em=JpaUtil.obtenirEntityManager();
26         try {
27             em.merge(i);
28         } catch(Exception e) {
29         }
30     }
31
32 }
```

dao.PersonneDaoJpa

```

1
2 package dao;
3 import java.util.List;
4 import javax.persistence.EntityManager;
5 import javax.persistence.Query;
6 import metier.modele.Employe;
7 import metier.modele.Personne;
8
9 /**
10  *
11  * @author Chanèle Jourdan, Jorge Terreu, Corentin laharotte
12  * Classe qui gère la persistance des Personnes
13  */
14 public class PersonneDaoJpa{
15
16     public static void creerPersonne(Personne p){
17         EntityManager em=JpaUtil.obtenirEntityManager();
18         try {
19             em.persist(p);
20         } catch(Exception e) {
21             }
22     }
23
24     //mise à jour de la Personne
25     //Pour le cas où on change sa disponibilité
26     public static void modifierPersonne(Personne p){
27         EntityManager em=JpaUtil.obtenirEntityManager();
28         try {
29             em.merge(p);
30         } catch(Exception e) {
31             }
32     }
33
34     //Trouve la peronne qui possède le mail indiqué
35     public static Personne recupererPersonne(String mail){
36         String jpql= "Select p from Personne p where p.mail=:email";
37         Query requete=JpaUtil.obtenirEntityManager().createQuery(jpql);
38         requete.setParameter("email",mail);
39         Personne result=null;
40         try{
41             result=(Personne) requete.getSingleResult();
42         } catch (Exception e){
43             }
44         return result;
45     }
46
47     //Trouve tous les employés disponibles et qui travaillent à l'heure indiquée
48     public static List<Employe> trouverListeEmployeDispo(int heure){
49         EntityManager em=JpaUtil.obtenirEntityManager();
50         String jpql= "Select e from Employe e where e.dispo=1 and e.horaireEntree <=:heure and e.horaireSortie>:heure";
51         Query requete=em.createQuery(jpql);
52         requete.setParameter("heure",heure);
53         List<Employe> result=null;
54         try{
55             result=(List<Employe>) requete.getResultList();
56         } catch (Exception e){
57             }
58         return result;
59     }
60
61 }

```

metier.modele.Personne

```

1
2 package metier.modele;
3
4 import java.io.Serializable;
5 import javax.persistence.Entity;
6 import javax.persistence.GeneratedValue;
7 import javax.persistence.GenerationType;
8 import javax.persistence.Id;
9 import javax.persistence.Inheritance;
10 import javax.persistence.InheritanceType;
11
12 /**
13  *
14  * @author Chanèle Jourdan, Jorge Terreu, Corentin laharotte
15  * Description d'une Personne
16  */
17 @Entity
18 @Inheritance(strategy=InheritanceType.SINGLE_TABLE)
19 public abstract class Personne implements Serializable {
20
21     @Id
22     @GeneratedValue(strategy = GenerationType.AUTO)
23     private Integer id;
24
25     private String nom;
26     private String prenom;
27     private boolean civilite;
28     private String motDePasse;
29
30     private String adressePostale;
31
32     private String tel;
33     private String mail;
34
35     //calculer à partir de l'adresse
36     private double latitude;
37     private double longitude;
38
39
40     public Personne() {
41     }
42
43     public Personne(boolean civilite, String nom, String prenom, String motDePasse, String adressePostale
44         , String tel, String mail) {
45         this.civilite = civilite;
46         this.nom = nom;
47         this.prenom = prenom;
48         this.motDePasse = motDePasse;
49         this.adressePostale = adressePostale;
50         this.tel = tel;
51         this.mail = mail;
52     }
53
54     public void setCivilite(boolean civilite) {
55         this.civilite = civilite;
56     }
57
58     public void setNom(String nom) {
59         this.nom = nom;
60     }
61
62     public void setPrenom(String prenom) {
63         this.prenom = prenom;
64     }
65
66     public void setMotDePasse(String motDePasse) {
67         this.motDePasse = motDePasse;
68     }
69
70     public void setAdressePostale(String adressePostale) {
71         this.adressePostale = adressePostale;
72     }
73
74     public void setTel(String tel) {
75         this.tel = tel;

```

```
75     }
76
77     public void setMail(String mail) {
78         this.mail = mail;
79     }
80
81     public void setLatitude(Float latitude) {
82         this.latitude = latitude;
83     }
84
85     public void setLongitude(Float longitude) {
86         this.longitude = longitude;
87     }
88
89     public Integer getId() {
90         return id;
91     }
92
93     public boolean isCivilite() {
94         return civilite;
95     }
96
97     public String getNom() {
98         return this.nom;
99     }
100
101     public String getPrenom() {
102         return prenom;
103     }
104
105     public String getMotDePasse() {
106         return motDePasse;
107     }
108
109     public String getAdressePostale() {
110         return adressePostale;
111     }
112
113     public String getTel() {
114         return tel;
115     }
116
117     public String getMail() {
118         return mail;
119     }
120
121     public double getLatitude() {
122         return latitude;
123     }
124
125     public double getLongitude() {
126         return longitude;
127     }
128
129     public void setId(Integer id) {
130         this.id = id;
131     }
132
133     public void setLatitude(double latitude) {
134         this.latitude = latitude;
135     }
136
137     public void setLongitude(double longitude) {
138         this.longitude = longitude;
139     }
140
141 }
```

metier.modele.Intervention

```

1
2 package metier.modele;
3
4 import java.io.Serializable;
5 import java.util.Date;
6 import javax.persistence.Entity;
7 import javax.persistence.GeneratedValue;
8 import javax.persistence.GenerationType;
9 import javax.persistence.Id;
10 import javax.persistence.Inheritance;
11 import javax.persistence.InheritanceType;
12 import javax.persistence.Temporal;
13 import javax.persistence.TemporalType;
14 import javax.persistence.ManyToOne;
15
16 /**
17  *
18  * @author Chanèle Jourdan, Jorge Terreu, Corentin laharotte
19  * Description d'une Intervention
20  */
21 @Entity
22 @Inheritance(strategy=InheritanceType.SINGLE_TABLE)
23 public abstract class Intervention implements Serializable {
24     @ManyToOne
25     private Client unClient;
26
27     @ManyToOne
28     private Employe unEmploye;
29
30     @Id
31     @GeneratedValue(strategy = GenerationType.AUTO)
32     private Integer id;
33
34     private int type;
35     private String description;
36
37     private int status;
38     private int heureDeFin;
39     private String commentaire;
40
41     @Temporal(TemporalType.DATE)
42     private Date horodate;
43
44     public Intervention() {
45     }
46
47     public Intervention(int type, String description) {
48         this.type = type;
49         this.description = description;
50     }
51
52     public Integer getId() {
53         return id;
54     }
55
56     public int getStatus() {
57         return status;
58     }
59
60     public int getType() {
61         return type;
62     }
63
64     public String getDescription() {
65         return description;
66     }
67
68     public int getHeureDeFin() {
69         return heureDeFin;
70     }
71
72     public String getCommentaire() {
73         return commentaire;
74     }
75

```

```
76
77     public Date getHorodate() {
78         return horodate;
79     }
80
81     public void setStatus(int status) {
82         this.status = status;
83     }
84
85     public void setType(int type) {
86         this.type = type;
87     }
88
89     public void setDescription(String description) {
90         this.description = description;
91     }
92
93     public void setHeureDeFin(int heureDeFin) {
94         this.heureDeFin = heureDeFin;
95     }
96
97     public void setCommentaire(String commentaire) {
98         this.commentaire = commentaire;
99     }
100
101     public void setHorodate(Date horodate) {
102         this.horodate = horodate;
103     }
104
105     public Client getUnClient() {
106         return unClient;
107     }
108
109     public void setUnClient(Client unClient) {
110         this.unClient = unClient;
111     }
112
113     public Employe getUnEmploye() {
114         return unEmploye;
115     }
116
117     public void setUnEmploye(Employe unEmploye) {
118         this.unEmploye = unEmploye;
119     }
120
121 }
```

metier.modele.Employe

```

1
2 package metier.modele;
3
4 import java.util.ArrayList;
5 import java.util.List;
6 import javax.persistence.Entity;
7 import javax.persistence.OneToMany;
8
9 /**
10  *
11  * @author Chanèle Jourdan, Jorge Terreu, Corentin laharotte
12  * Description d'un Employe
13  */
14 @Entity
15 public class Employe extends Personne {
16     @OneToMany(mappedBy="unEmploye")
17     private List<Intervention> listInterventions;
18
19     boolean dispo;
20     int horaireEntree;
21     int horaireSortie;
22
23     public Employe() {
24     }
25
26     public Employe(boolean civilite, String nom, String prenom, String motDePasse, String adressePostale,
27         String tel, String mail, boolean dispo, int horaireEntree, int horaireSortie) {
28         super(civilite, nom, prenom, motDePasse, adressePostale, tel, mail);
29         this.dispo=dispo;
30         this.horaireEntree=horaireEntree;
31         this.horaireSortie=horaireSortie;
32         this.listInterventions=new ArrayList<Intervention>();
33     }
34
35     public void ajouterIntervention(Intervention i){
36         listInterventions.add(i);
37         i.setUnEmploye(this);
38     }
39
40     public boolean isDispo() {
41         return dispo;
42     }
43
44     public int getHoraireEntree() {
45         return horaireEntree;
46     }
47
48     public int getHoraireSortie() {
49         return horaireSortie;
50     }
51
52     public List<Intervention> getListInterventions() {
53         return listInterventions;
54     }
55
56     public void setDispo(boolean dispo) {
57         this.dispo = dispo;
58     }
59
60     public void setHoraireEntree(int horaireEntree) {
61         this.horaireEntree = horaireEntree;
62     }
63
64     public void setHoraireSortie(int horaireSortie) {
65         this.horaireSortie = horaireSortie;
66     }
67
68     public void setListInterventions(List<Intervention> interventions) {
69         this.listInterventions = interventions;
70     }
71 }

```

`metier.modele.InterventionIncident`

```
1
2 package metier.modele;
3
4 import javax.persistence.Entity;
5
6 /**
7  *
8  * @author Chanèle Jourdan, Jorge Terreu, Corentin laharotte
9  * Description d'une InterventionIncident
10  */
11 @Entity
12 public class InterventionIncident extends Intervention{
13
14     public InterventionIncident() {
15     }
16
17     public InterventionIncident(String description) {
18         super(3, description);
19     }
20 }
```


metier.modele.InterventionAnimal

```
1
2 package metier.modele;
3
4 import javax.persistence.Entity;
5
6 /**
7  *
8  * @author Chanèle Jourdan, Jorge Terreu, Corentin laharotte
9  * Description d'une InterventionAnimal
10  */
11 @Entity
12 public class InterventionAnimal extends Intervention {
13     private String animal;
14
15     public InterventionAnimal() {
16     }
17
18     public InterventionAnimal(String animal) {
19         this.animal = animal;
20     }
21
22     public InterventionAnimal(String animal, String description) {
23         super(1, description); //initialisation du type à 1
24         this.animal = animal;
25     }
26
27     public String getAnimal() {
28         return animal;
29     }
30
31     public void setAnimal(String animal) {
32         this.animal = animal;
33     }
34 }
```

metier.modele.Client

```

1
2 package metier.modele;
3
4 /**
5  *
6  * @author Chanèle Jourdan, Jorge Terreu, Corentin laharotte
7  * Description d'un Client
8  */
9 import java.util.ArrayList;
10 import java.util.Date;
11 import java.util.List;
12 import javax.persistence.Entity;
13 import javax.persistence.OneToMany;
14 import javax.persistence.Temporal;
15 import javax.persistence.TemporalType;
16
17 @Entity
18 public class Client extends Personne {
19     @OneToMany(mappedBy="unClient")
20     private List<Intervention> listInterventions;
21
22     @Temporal(TemporalType.DATE)
23     private Date dateDeNaissance;
24
25     public Client() {
26         this.listInterventions=new ArrayList<Intervention>();
27     }
28
29     public Client(boolean civilite, String nom, String prenom, String motDePasse, String adressePostale,
30         String tel, String mail, Date dateDeNaissance) {
31         super(civilite, nom, prenom, motDePasse, adressePostale, tel, mail);
32         this.dateDeNaissance = dateDeNaissance;
33         this.listInterventions=new ArrayList<Intervention>();
34     }
35
36     public void ajouterIntervention(Intervention i){
37         listInterventions.add(i);
38         i.setUnClient(this);
39     }
40
41     public Date getDateDeNaissance() {
42         return dateDeNaissance;
43     }
44
45     public void setDateDeNaissance(Date dateDeNaissance) {
46         this.dateDeNaissance = dateDeNaissance;
47     }
48
49     public List<Intervention> getListInterventions() {
50         return listInterventions;
51     }
52
53     public void setListInterventions(List<Intervention> listInterventions) {
54         this.listInterventions = listInterventions;
55     }
56 }

```

metier.modele.InterventionLivraison

```
1
2 package metier.modele;
3
4 /**
5  *
6  * @author Chanèle Jourdan, Jorge Terreu, Corentin laharotte
7  * Description d'une InterventionLivraison
8  */
9
10 import javax.persistence.Entity;
11
12 @Entity
13 public class InterventionLivraison extends Intervention {
14     private String objet;
15     private String entreprise;
16
17     public InterventionLivraison() {
18     }
19
20     public InterventionLivraison(String objet, String entreprise, String description) {
21         super(2, description); //initialisation du type à 2
22         this.objet = objet;
23         this.entreprise = entreprise;
24     }
25
26     public String getObjet() {
27         return objet;
28     }
29
30     public String getEntreprise() {
31         return entreprise;
32     }
33
34     public void setObjet(String objet) {
35         this.objet = objet;
36     }
37
38     public void setEntreprise(String entreprise) {
39         this.entreprise = entreprise;
40     }
41
42 }
```

metier.service.Service

```

1  /* @author jorge terreu, corentin laharotte, chanèle jourdan
2     Classe qui implémente l'ensemble des services de notre application
3  */
4
5  package metier.service;
6
7  import com.google.maps.model.LatLng;
8  import dao.InterventionDaoJpa;
9  import dao.JpaUtil;
10 import dao.PersonneDaoJpa;
11 import java.text.SimpleDateFormat;
12 import java.util.ArrayList;
13 import java.util.List;
14 import metier.modele.Client;
15 import metier.modele.Employe;
16 import metier.modele.Intervention;
17 import metier.modele.InterventionAnimal;
18 import metier.modele.InterventionIncident;
19 import metier.modele.InterventionLivraison;
20 import metier.modele.Personne;
21 import java.util.Date;
22 import util.GeoTest;
23 import util.Message;
24
25 public class Service{
26
27     //Connection du client ou de l'employé à l'application
28     public static Personne seConnecter(String mail, String motdepasse){
29         JpaUtil.creerEntityManager();
30         JpaUtil.ouvrirTransaction();
31         Personne a=PersonneDaoJpa.recupererPersonne(mail);
32         if (a!=null){
33             if (!(a.getMotDePasse().equals(motdepasse)))
34                 a=null;
35         }
36         JpaUtil.validerTransaction();
37         JpaUtil.fermerEntityManager();
38         return a;
39     }
40
41     //Inscription d'un client. Vérifications effectuées : tous les champs ont été remplis, le mail
42     //contient un @, le client a plus de 16 ans, le tel contient 10 caractères
43     public static Personne seInscrire(boolean civilite, String nom, String prenom, String motDePasse,
44         String adressePostale, String tel, String mail, Date dateDeNaissance){
45         JpaUtil.creerEntityManager();
46         JpaUtil.ouvrirTransaction();
47         Personne x = PersonneDaoJpa.recupererPersonne(mail);
48         if (x==null){
49             Date ahorita = new Date();
50             //seize ans en millisecondes
51             double seizeAns= 16.0*365.0*24.0*3600.0*1000.0;
52             //différence de temps en millisecondes entre aujourd'hui et dateDeNaissance
53             double differenceTemps=ahorita.getTime() - dateDeNaissance.getTime();
54
55             if(8 < motDePasse.length() && motDePasse.length()< 16 && nom.length()>0 && prenom
56                 .length()>0 && mail.length()>2 && mail.contains("@") &&
57                 adressePostale.length()>0 && tel.length()==10 && (differenceTemps > seizeAns)) {
58
59                 //Création et persistance d'un nouveau client
60                 x = new Client (civilite, nom, prenom, motDePasse, adressePostale, tel,
61                     mail, dateDeNaissance);
62                 calculerLatLng(x);
63                 PersonneDaoJpa.creerPersonne(x);
64
65                 //Envoi de mail de confirmation d'inscription
66                 String message="Bonjour "+prenom+", nous vous confirmons votre
67                     inscription au service PROACT'IF.";
68                 Message.envoyerMail("contact@proact.if", mail, "Bienvenue chez PROACT'IF",
69                     message);
70             }
71         }else{
72             //Envoie de mail pour informer de l'échec de l'inscription
73             String message="Bonjour "+prenom+", votre inscription a échoué, veuillez recommencer
74                 ultérieurement.";
75             Message.envoyerMail("contact@proact.if", mail, "Bienvenue chez PROACT'IF", message);
76         }
77     }
78 }

```

```

69         x=null;
70     }
71     JpaUtil.validerTransaction();
72     JpaUtil.fermerEntityManager();
73     return x;
74 }
75
76 //Demande d'une intervention de type Animal. Vérification : tous les champs sont remplis
77 public static Employe demanderIntervention(Client c,int type, String description){
78     JpaUtil.creerEntityManager();
79     JpaUtil.ouvrirTransaction();
80     Employe employeTrouve=trouverEmploye(c);
81     if (employeTrouve!=null && type>= 0 && type <=3 && description.length()>0){
82
83         //Création et persistance de l'intervention
84         InterventionIncident a = new InterventionIncident(description);
85         a.setHorodate(new Date());
86         c.ajouterIntervention(a);
87         employeTrouve.ajouterIntervention(a);
88         employeTrouve.setDispo(false);
89         InterventionDaoJpa.creerIntervention(a);
90
91         //Envoi d'une notification à l'employé trouvé
92         String message = "Intervention_Animal_demandee_pour_" +c.getPrenom()+ "_" +c.
            getNom()+ ",_" +c.getAdressePostale()+ "._Commentaire:_"+description;
93
94         Message.envoyerNotification(employeTrouve.getNom(), employeTrouve.getPrenom(),
95             employeTrouve.getTel(), message);
96     }else{
97         employeTrouve=null;
98     }
99     JpaUtil.validerTransaction();
100    JpaUtil.fermerEntityManager();
101    return employeTrouve;
102 }
103
104 //Demande d'une intervention de type Livraison. Vérification : tous les champs sont remplis
105 public static Employe demanderIntervention(Client c,int type, String description, String objet,
106     String entreprise){
107     JpaUtil.creerEntityManager();
108     JpaUtil.ouvrirTransaction();
109     Employe employeTrouve=trouverEmploye(c);
110     if (employeTrouve!=null && type>= 0 && type <=3 && description.length()>0 && objet.length
111         ()>0 && entreprise.length()>0){
112
113         //Creation et persistance de l'intervention
114         InterventionLivraison a = new InterventionLivraison(objet, entreprise,
115             description);
116         a.setHorodate(new Date());
117         c.ajouterIntervention(a);
118         employeTrouve.ajouterIntervention(a);
119         employeTrouve.setDispo(false);
120         InterventionDaoJpa.creerIntervention(a);
121
122         //Envoi d'une notification à l'employé trouvé
123         String message = "Intervention_Livraison_demandee_pour_" +c.getPrenom()+ "_" +c.
124             getNom()+ ",_" +c.getAdressePostale()+ "._Commentaire:_"+description;
125
126         Message.envoyerNotification(employeTrouve.getNom(), employeTrouve.getPrenom(),
127             employeTrouve.getTel(), message);
128     }else{
129         employeTrouve=null;
130     }
131     JpaUtil.validerTransaction();
132     JpaUtil.fermerEntityManager();
133     return employeTrouve;
134 }
135
136 //Demande d'une intervention de type Incident. Vérification : tous les champs sont remplis
137 public static Employe demanderIntervention(Client c,int type, String description, String animal){
138     JpaUtil.creerEntityManager();
139     JpaUtil.ouvrirTransaction();
140     Employe employeTrouve=trouverEmploye(c);
141     if (employeTrouve!=null && type>= 0 && type <=3 && description.length()>0 && animal.

```

```

length()>0 ){
137
138     //Creation et persistance de l'intervention
139     InterventionAnimal a = new InterventionAnimal(animal, description);
140     a.setHorodate(new Date());
141     c.ajouterIntervention(a);
142     employeTrouve.ajouterIntervention(a);
143     employeTrouve.setDispo(false);
144     InterventionDaoJpa.creerIntervention(a);
145
146     //Envoi d'une notification à l'employé trouvé
147     String message = "Intervention_Incident_demandee_pour_ " +c.getPrenom()+ " " +c.
        getNom()+ ",_ " +c.getAdressePostale()+ "._Commentaire:_ " +description;
148
149     Message.envoyerNotification(employeTrouve.getNom(), employeTrouve.getPrenom(),
        employeTrouve.getTel(), message);
150
151     }else{
152         employeTrouve=null;
153     }
154     JpaUtil.validerTransaction();
155     JpaUtil.fermerEntityManager();
156     return employeTrouve;
157 }
158
159 //Recherche de l'employé le plus proche du client pour faire une intervention.
160 //Vérification : il est disponible, l'heure de la demande d'intervention est dans ses horaires de
    travail
161 public static Employe trouverEmploye(Client c){
162     int heure=heureActuelleToInt();
163
164     //Récupération de la liste des employés disponibles
165     List <Employe> list=PersonneDaoJpa.trouverListeEmployeDispo(heure);
166
167     Employe employeSelect=null;
168     if(list!=null){
169         double min=1000000000;
170         LatLng clientGPS=new LatLng(c.getLatitude(),c.getLongitude());
171         for (Employe e : list){
172             LatLng employeGPS=new LatLng(e.getLatitude(),e.getLongitude());
173             double tempsTrajet=GeoTest.getTripDurationByBicycleInMinute(clientGPS,
                employeGPS);
174             if(tempsTrajet<min){
175                 min=tempsTrajet;
176                 employeSelect=e;
177             }
178         }
179         return employeSelect;
180     }
181
182     //Récupération de l'historique d'un client pour le consulter
183     public static List<Intervention> recupererHistorique(Client c){
184         JpaUtil.creerEntityManager();
185         JpaUtil.ouvrirTransaction();
186         PersonneDaoJpa.modifierPersonne(c);
187         JpaUtil.validerTransaction();
188         JpaUtil.fermerEntityManager();
189         return c.getListInterventions();
190     }
191
192     //Récupération des interventions d'un employé pour les consulter (tableau de bord)
193     public static List<Intervention> recupererTableauDeBord(Employe e){
194         List <Intervention> l=e.getListInterventions();
195         List <Intervention> list=new ArrayList<Intervention>();
196         Date d=new Date();
197         for (Intervention i: l){
198             if(d.compareTo(i.getHorodate())!=0)
199                 list.add(i);
200         }
201         return list;
202     }
203
204     //Clotûre d'une intervention par un employé. Vérification : tous les champs sont remplis
205     public static boolean cloturerIntervention(Employe emp, int status, int heureDefin, String
        commentaire){
206         JpaUtil.creerEntityManager();
207         JpaUtil.ouvrirTransaction();

```

```

207 Intervention i = rechercherInterventionEnCours(emp);
208 boolean cloture=false;
209 if (status >= 0 && status <=3 && commentaire.length()>0 && heureDefin >= 0 && heureDefin
    <= 24 && i != null){
210
211     //Modification de l'intervention
212     i.setStatus(status);
213     i.setHeureDeFin(heureDefin);
214     i.setCommentaire(commentaire);
215     InterventionDaoJpa.modifierIntervention(i);
216
217     //libération de l'employe
218     emp.setDispo(true);
219     PersonneDaoJpa.modifierPersonne(emp);
220
221     //Envoi d'une notification au client
222     String message = "Votre demande d'intervention a été cloturée. Commentaire de l'
        employe: " + commentaire;
223     Message.envoyerNotification(i.getUnClient().getNom(), i.getUnClient().getPrenom()
        , i.getUnClient().getTel(), message);
224
225     cloture=true;
226 }
227 JpaUtil.validerTransaction();
228 JpaUtil.fermerEntityManager();
229 return cloture;
230 }
231
232 //Recherche de l'intervention en cours pour un employé donnée (s'il en a une)
233 public static Intervention rechercherInterventionEnCours(Employe emp){
234     List<Intervention> list=emp.getListInterventions();
235     for (Intervention i : list){
236         if(i.getStatus()==0)
237             return i;
238     }
239     return null;
240 }
241
242 //Calcul des coordonnées GPS d'un employé ou d'un client
243 public static void calculerLatLng(Personne p){
244     LatLng GPS=GeoTest.getLatLng(p.getAdressePostale());
245     p.setLongitude(GPS.lng);
246     p.setLatitude(GPS.lat);
247 }
248
249 //SERVICE EN PLUS DE CEUX SPECIFIES DANS LE COMPTE RENDU
250 //Récupération de l'heure actuelle (pour savoir à quel moment est faite une demande d'
    intervention)
251 public static int heureActuelleToInt(){
252     SimpleDateFormat format = new SimpleDateFormat ("HH:mm");
253     Date heureActuelle = new Date();
254     String heureString = format.format(heureActuelle);
255     String nbHeureString = heureString.split(":")[0];
256     int heure=Integer.parseInt(nbHeureString);
257     return heure;
258 }
259 }

```

vue.Main

```

1  /* @author jorge terreu, corentin laharotte, chanèle jourdan
2     Classe qui représente l'interface avec l'utilisateur
3  */
4
5  package vue;
6
7  import dao.JpaUtil;
8  import dao.PersonneDaoJpa;
9  import java.util.ArrayList;
10 import java.util.Arrays;
11 import java.util.Date;
12 import java.text.SimpleDateFormat;
13 import java.util.List;
14 import metier.modele.Client;
15 import metier.modele.Employe;
16 import metier.modele.Intervention;
17 import metier.modele.Personne;
18 import metier.service.Service;
19 import util.Saisie;
20
21
22 public class Main {
23
24     public final static String NOM_PERSISTENCE = "TPDASIPU";
25
26     public static void main(String[] args) {
27         Personne p=null;
28
29         JpaUtil.init();
30
31         //méthode à appeler une seule fois (pour remplir la base avec les employés)
32         insererTousLesEmployes();
33
34         lancerMenuPrincipal(p);
35         JpaUtil.destroy();
36     }
37
38     //Création de la liste des employés de l'entreprise
39     public static List<Employe> listDesEmployes(){
40         List<Employe> l=new ArrayList<Employe>();
41         l.add(new Employe(true,"BORROTTI_MATIAS_DANTAS","Raphaël","motdepassepouremploye","8_Rue_Arago,_
42             Villeurbanne","328178508","rborrotimatiasdantas4171@free.fr",true,10,20));
43         l.add(new Employe(false,"OLMEADA_MARAIS","Nor","motdepassepouremploye","5_Rue_Léon_Fabre,_
44             Villeurbanne","0418932546","nolmeadamaraais1551@gmail.com",true,8,18));
45         l.add(new Employe(false,"RAYES_GEMEZ","Oléna","motdepassepouremploye","12_Rue_de_la_Prevoyance,_
46             Villeurbanne","0532731620","orayesgomez5313@outlook.com",true,6,16));
47         l.add(new Employe(false,"SING","Ainhua","motdepassepouremploye","4_Rue_Phelypeaux,_Villeurbanne",
48             "0705224200","asing8183@free.fr",true,5,15));
49         l.add(new Employe(true,"ABDIULLINA","David_Alexander","motdepassepouremploye","8_Rue_Wilhelmine,_
50             Villeurbanne","0590232772","david-alexander.abdiullina@laposte.net",true,12,22));
51         l.add(new Employe(true,"WOAGNER","Moez","motdepassepouremploye","6_Rue_Camille_Koechlin,_
52             Villeurbanne","0832205629","moez.woagner@laposte.net",true,7,17));
53         l.add(new Employe(true,"HONRY","Matteo","motdepassepouremploye","9_Impasse_Guillet,_Villeurbanne",
54             "0482381862","matteo.honry@yahoo.com",true,5,15));
55         l.add(new Employe(true,"CECCANI","Kevin","motdepassepouremploye","20_Rue_Decomberousse,_
56             Villeurbanne","0664426037","kevin.ceccani@hotmail.com",true,8,18));
57         l.add(new Employe(false,"VOYRET","Alice","motdepassepouremploye","1_Rue_d'Alsace,_Villeurbanne",
58             "0486856520","alice.voyret@hotmail.com",true,6,16));
59         l.add(new Employe(true,"RINERD","Julien","motdepassepouremploye","4_Rue_de_la_Jeunesse,_
60             Villeurbanne","0727252485","jrinerd5241@yahoo.com",true,9,19));
61         return l;
62     }
63
64     //Permet d'insérer l'ensemble des employés dans la base de données
65     public static void insererTousLesEmployes(){
66         JpaUtil.creerEntityManager();
67         JpaUtil.ouvrirTransaction();
68         List<Employe> list=listDesEmployes();
69         for (Employe e : list){
70             Service.calculerLatLng(e);
71             PersonneDaoJpa.creerPersonne(e);
72         }
73         JpaUtil.validerTransaction();
74         JpaUtil.fermerEntityManager();
75     }
76 }

```



```

66
67 //Permet d'afficher une liste d'intervention
68 //Sert pour l'affichage de l'historique d'un client et du tableau de bord d'un employé
69 public static void afficherListeInterventions (List<Intervention> listint){
70     if (!listint.isEmpty()){
71         System.out.println("Type|Description|Statut");
72         for (Intervention i : listint){
73             int type=i.getType();
74             int status=i.getStatus();
75             String typeString="";
76             String statusString="";
77             switch (type) {
78                 case 1:
79                     typeString="Animal";
80                     break;
81                 case 2:
82                     typeString="Livraison";
83                     break;
84                 case 3:
85                     typeString="Incident";
86                     break;
87                 default:
88                     break;
89             }
90             switch (status) {
91                 case 0:
92                     statusString="En_Cours";
93                     break;
94                 case 1:
95                     statusString="Succès";
96                     break;
97                 case 2:
98                     statusString="Echec";
99                     break;
100                default:
101                    break;
102            }
103            System.out.println ("|"+typeString+"|"+i.getDescription()+"|"+statusString+"|");
104        }
105    }else{
106        System.out.println ("Pas d'intervention");
107    }
108 }
109
110
111 //Affichage des différents Menu (principal, client, employé)
112
113 public static void affichageMenuPrincipal(){
114     System.out.println("-----");
115     System.out.println("Bienvenue sur ProAct' IF");
116     System.out.println("-----");
117     System.out.println("Choisir une option:");
118     System.out.println("1. Se connecter");
119     System.out.println("2. S'inscrire");
120     System.out.println("3. Quitter");
121     System.out.println();
122 }
123
124 public static void affichageMenuClient(){
125     System.out.println();
126     System.out.println("Choisir une option:");
127     System.out.println("1. Demander une intervention");
128     System.out.println("2. Consulter Historique");
129     System.out.println("3. Retour");
130     System.out.println();
131 }
132
133 public static void affichageMenuEmploye(){
134     System.out.println();
135     System.out.println("Choisir une option:");
136     System.out.println("1. Cloturer l'intervention en cours");
137     System.out.println("2. Consulter Tableau de bord");
138     System.out.println("3. Retour");
139     System.out.println();
140 }
141

```

```

142
143 //Lancement des différents menus (principal, client, employé)
144 //Ils font automatiquement appel aux différents affichage et service selon la demande de l'
    utilisateur
145
146 public static void lancerMenuPrincipal(Personne p){
147     affichageMenuPrincipal();
148     int choix = Saisie.lireInteger("Choix: ", Arrays.asList(1,2,3));
149     switch(choix){
150         case 1:
151             {
152                 String mail = Saisie.lireChaine("Mail: ");
153                 String motDePasse = Saisie.lireChaine("Mot de passe: ");
154                 p=Service.seConnecter(mail, motDePasse);
155                 if(p==null){
156                     System.out.println("Echec de la connexion\n\n");
157                     lancerMenuPrincipal(p);
158                 }
159                 if (p instanceof Employe){
160                     lancerMenuEmploye((Employe)p);
161                 } else if (p instanceof Client){
162                     lancerMenuClient((Client)p);
163                 }
164                 break;
165             }
166         case 2:
167             try{
168                 int madame = Saisie.lireInteger("Civilité (0=Mme, 1=Mr) : ", Arrays.asList(0,1));
169                 boolean civilite=false;
170                 if (madame==1){
171                     civilite=true;
172                 }
173                 String nom = Saisie.lireChaine("Nom: ");
174                 String prenom = Saisie.lireChaine("Prenom: ");
175                 String mdp = Saisie.lireChaine("Mot de passe: ");
176                 String addpost = Saisie.lireChaine("Adresse postale: ");
177                 String tel = Saisie.lireChaine("Tel: ");
178                 String mail = Saisie.lireChaine("Mail: ");
179
180                 SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
181                 String dateNaiss = Saisie.lireChaine("Date de naissance (dd/MM/yyyy) : ");
182                 Date date = sdf.parse(dateNaiss);
183                 p=Service.seInscrire(civilite, nom, prenom, mdp, addpost, tel, mail, date);
184
185                 if(p==null){
186                     System.out.println("Erreur: L'inscription n'a pas pu être réalisée (champ
                        vide ou mail déjà utilisé)\n\n");
187                     lancerMenuPrincipal(p);
188                 } else {
189                     System.out.println("Inscription réussie\n\n");
190                 }
191
192                 if (p instanceof Employe){
193                     lancerMenuEmploye((Employe)p);
194                 } else if (p instanceof Client){
195                     lancerMenuClient((Client)p);
196                 }
197             }catch(Exception e){
198                 System.out.println("Erreur: L'inscription n'a pas pu être réalisée (erreur sur la
                        date)\n\n");
199                 lancerMenuPrincipal(p);
200             }
201         }
202         break;
203     case 3:
204         System.out.println("A bientôt");
205         System.exit(0);
206     }
207 }
208
209
210 public static void lancerMenuEmploye(Employe e){
211     affichageMenuEmploye();
212     int choix = Saisie.lireInteger("Choix: ", Arrays.asList(1,2,3));
213     switch(choix){
214         case 1:
215             if(Service.rechercherInterventionEnCours(e)!=null){

```

```

216         int status = Saisie.lireInteger("Statut (1=Succès, 2=Echec) : ", Arrays.asList
217             (1,2));
218         int heureDeFin = Saisie.lireInteger("Heure de fin : ");
219         String commentaire = Saisie.lireChaine("Commentaire : ");
220         boolean cloture = Service.cloturerIntervention(e, status, heureDeFin, commentaire);
221         if (!cloture) {
222             System.out.println("Erreur : Cloture de l'intervention échouée\n\n");
223         } else {
224             System.out.println("Cloture de l'intervention réussie\n\n");
225         }
226     } else {
227         System.out.println("Vous n'avez pas d'intervention à clôturer\n\n");
228     }
229     lancerMenuEmploye(e);
230     break;
231 case 2:
232     afficherListeInterventions(Service.recupererTableauDeBord(e));
233     lancerMenuEmploye(e);
234     break;
235 case 3:
236     lancerMenuPrincipal(e);
237     break;
238 }
239 }
240 public static void lancerMenuClient(Client c){
241     affichageMenuClient();
242     int choix = Saisie.lireInteger("Choix : ", Arrays.asList(1,2,3));
243     switch(choix){
244         case 1:
245             Employe tmp=null;
246             int type = Saisie.lireInteger("Type (1=Animal, 2=Livraison, 3=Incident) : ", Arrays.
247                 asList(1,2,3));
248             switch (type) {
249                 case 1:
250                     String animal = Saisie.lireChaine("Animal : ");
251                     String description = Saisie.lireChaine("Description : ");
252                     tmp=Service.demanderIntervention(c,type,description,animal);
253                     break;
254                 case 2:
255                     String objet = Saisie.lireChaine("Objet : ");
256                     String entreprise = Saisie.lireChaine("Entreprise : ");
257                     String description1 = Saisie.lireChaine("Description : ");
258                     tmp=Service.demanderIntervention(c,type,description1,objet, entreprise);
259                     break;
260                 case 3:
261                     String description2= Saisie.lireChaine("Description : ");
262                     tmp=Service.demanderIntervention(c,type,description2);
263                     break;
264                 default:
265                     break;
266             }
267             if(tmp==null){
268                 System.out.println("Erreur : la demande d'intervention n'est pas possible (champ
269                     vide ou pas d'employé disponible)");
270             } else {
271                 System.out.println("Demande d'intervention validée");
272             }
273             lancerMenuClient(c);
274             break;
275         case 2:
276             afficherListeInterventions(Service.recupererHistorique(c));
277             lancerMenuClient(c);
278             break;
279         case 3:
280             lancerMenuPrincipal(c);
281             break;
282         default:
283             break;
284     }
285 }
286 }
287 }
288 }

```