```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package metier.service;

import dao.ClientDao;
import dao.DevisDao;
import dao.InfoPrincipaleDao;
import dao.JpaUtil;
import dao.PaysDao;
import dao.VoyageDao;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.GregorianCalendar;
import java.util.List;
import metier.modele.Circuit;
import metier.modele.Client;
import metier.modele.Conseiller;
import metier.modele.Devis;
import metier.modele.InfoPrincipale;
import metier.modele.Pays;
import metier.modele.Sejour;
import metier.modele.Voyage;
import util.Aleatoire;
import util.Saisie;

/**
 *
 * @author Administrateur
 */
public class Service {

    /**
     *
     */
    protected static DateFormat USR_BIRTH_DATE
            = new SimpleDateFormat("dd-MM-yyyy");
    protected static DateFormat US_DATE_FORMAT
            = new SimpleDateFormat("yyyy-MM-dd");

    /**

     * Cette méthode créée un Devis à partir du code du voyage et de l'adresse
```

```
     * mail du client, du départ (infosPrincipale) et du nombre de personnes
     * demandant le devis. Les liens entre les entités se font automatiquement.
     * Le conseiller est choisi parmis les conseillées de ce pays qui ont le
     * moins de clients. Tous les changements et le devis sont sauvergardés dans
     * la base de données.
     *
     * @param CodeVoyage
     * @param addresseMailClient
     * @param choixInfos
     * @param nbPersonnes
     */
    public static void creerDevis(String CodeVoyage, String addresseMailClient,
            String choixInfos, String nbPersonnes) {
        JpaUtil.ouvrirTransaction();
        Date currentDate
                = new Date(new GregorianCalendar().getTime().getTime());
        Devis d = new Devis(currentDate,
                VoyageDao.findVoyageByCodeVoyage(CodeVoyage),
                ClientDao.findClientByMail(addresseMailClient));
        JpaUtil.persist(d);

        JpaUtil.validerTransaction();

        if (choisirConseiller(d)) {
            JpaUtil.ouvrirTransaction();
            d.setNbPersonnes(Integer.parseInt(nbPersonnes));
            InfoPrincipale i = InfoPrincipaleDao.findInfoByCodeInfo(choixInfos);
            d.setChoixCaracteristiques(i);

            JpaUtil.merge(d);
            d.getClientDevis().addDevis(d);
            JpaUtil.merge(d.getClientDevis());
            System.out.println(afficheDevis(d));
            JpaUtil.validerTransaction();

        } else {
            JpaUtil.annulerTransaction();
        }

    }

    /**
     * Cette méthode créée un Devis à partir du code du voyage et de l'adresse

     * mail du client demandant le devis. Les liens entre les entités se font
     * automatiquement. Le choix du nombre de passager est aléatoire et compris
     * entre 2 et 5. Le choix de l'infoPrincipale (ou départ) est aléatoire
```

```
     * parmis les départs disponibles pour le voyage. Le conseiller est choisie
     * parmis les conseillées de ce pays qui ont le moins de clients. Tous les
     * changements et le devis sont sauvergardés dans la base de données.
     *
     * @param CodeVoyage
     * @param addresseMailClient
     */
    public static void creerDevis(String CodeVoyage,
            String addresseMailClient) {
        JpaUtil.ouvrirTransaction();
        Date currentDate
                = new Date(new GregorianCalendar().getTime().getTime());
        Devis d = new Devis(currentDate,
                VoyageDao.findVoyageByCodeVoyage(CodeVoyage),
                ClientDao.findClientByMail(addresseMailClient));
        JpaUtil.persist(d);

        JpaUtil.validerTransaction();

        if (choisirConseiller(d)) {
            JpaUtil.ouvrirTransaction();
            d.setNbPersonnes(ChoisirNbPassager());
            d.setChoixCaracteristiques(ChoisirInfoPrincipale(CodeVoyage));

            JpaUtil.merge(d);
            d.getClientDevis().addDevis(d);
            JpaUtil.merge(d.getClientDevis());
            System.out.println(afficheDevis(d));
            JpaUtil.validerTransaction();

        } else {
            JpaUtil.annulerTransaction();
        }

    }

    /**
     * Cette méthode créée un Client à partir de ses caractéritique. La date
     * doit être une chaine de caractère de format "AAAA-MM-JJ". L'envoie du
     * mail au partenaire est simulé par l'affichage d'un message. Le client est
     * sauvegardé dans la base de donées. Le Voyage ayant le code Voyage

     * codeVoyage et le client ayant l'addresse mail addresseMailClient doivent
     * être présent dans la base de donées.
     *
     *
```

```
     * @param Civilite
     * @param Nom
     * @param prenom
     * @param Date
     * @param Adresse
     * @param telephone
     * @param mail
     */
    public static void creerClient(String Civilite, String Nom, String prenom,
            String Date, String Adresse, String telephone, String mail) {
        JpaUtil.ouvrirTransaction();
        Client c = new Client(Civilite, Nom, prenom, parseDateUsFormat(Date),
                Adresse, telephone, mail);

        JpaUtil.persist(c);
        System.out.println(c);
        c.setAutorisationPartenaires(true);
        System.out.println(envoyerMailPartenaires(c));

        JpaUtil.validerTransaction();
    }

    /**
     * Cette méthode créée un Pays à partir des ses caractéristiques. Le pays
     * est également ajouté à la base de données.
     *
     * @param nom
     * @param code
     * @param continent
     * @param capitale
     * @param langues
     * @param superficie
     * @param population
     * @param regimePolitique
     */
    public static void creerPays(String nom, String code, String continent,
            String capitale, String langues, float superficie, float population,
            String regimePolitique) {
        JpaUtil.ouvrirTransaction();
        Pays p = new Pays(nom, code, nom, capitale, langues, superficie,

                population, regimePolitique);

        JpaUtil.persist(p);
        System.out.println(p);

        JpaUtil.validerTransaction();
```

```
    }

    /**
     * Cette méthode créée un Conseiller à partir de ses paramètres puis le
     * conseiller est enregistré dans la base de données.Le paramètre
     * CodePaysConseilles est un tableau de string contenant des codes de pays
     * enregistrés dans la base de données. Les liens entre le Conseiller et les
     * Pays qu'il conseille se font automatiquement.
     *
     * @param Civilite
     * @param Nom
     * @param prenom
     * @param Date
     * @param Adresse
     * @param telephone
     * @param mail
     * @param CodePaysConseilles
     */
    public static void creerConseiller(String Civilite, String Nom,
            String prenom, String Date, String Adresse, String telephone,
            String mail, String[] CodePaysConseilles) {
        JpaUtil.ouvrirTransaction();
        Conseiller c = new Conseiller(Civilite, Nom, prenom,
                parseDateUsFormat(Date), Adresse, telephone, mail);
        JpaUtil.persist(c);
        System.out.println(c);
        for (int lgtCodePC = 0; lgtCodePC < CodePaysConseilles.length;
                lgtCodePC++) {
            Pays p = PaysDao.findPaysByCodePays(CodePaysConseilles[lgtCodePC]);
            p.addConseillers(c);
            c.addPays(p);
            JpaUtil.merge(p);
        }
        JpaUtil.merge(c);
        JpaUtil.validerTransaction();
    }

    /**

     * Cette méthode créée un Infoprincipale (ou Départ) à partir de ses
     * caractéristiques. La date doit être une chaine de caractère de format
     * "AAAA-MM-JJ". Le Voyage ayant le code voyage codeVoyage doit être présent
     * dans la base de donées. Le lien entre le Voyage et le Départ se fait
     * automatiquement. Le Départ est également ajouté à la base de données.
     *
     * @param villeDepart
```

```
     * @param DateDepart
     * @param Prix
     * @param transport
     * @param codeVoyage
     */
    public static void creerInfoPrincipale(String villeDepart,
            String dateDepart, int Prix, String transport, String codeVoyage) {
        JpaUtil.ouvrirTransaction();

        InfoPrincipale iP = new InfoPrincipale(villeDepart,
                parseDateUsFormat(dateDepart), Prix, transport);
        Voyage voyageAssocie = VoyageDao.findVoyageByCodeVoyage(codeVoyage);
        iP.setVoyageAssocie(voyageAssocie);

        JpaUtil.persist(iP);
        System.out.println(iP);

        voyageAssocie.addInfos(iP);
        JpaUtil.merge(voyageAssocie);
        JpaUtil.validerTransaction();
    }

    /**
     * Cette méthode créée un Circuit à partir de ses caractéristiques. Le Pays
     * ayant le code pays codePays doit être présent dans la base de donées. Le
     * liens entre le Circuit et le Pays se fait automatiquement. La durée est
     * en jour. Le Circuit est également ajouter à la base de données.
     *
     * @param moyenDeTransport
     * @param kilometres
     * @param codePays
     * @param codeVoyage
     * @param intitule
     * @param duree
     * @param description
     */
    public static void creerCircuit(String moyenDeTransport, int kilometres,

            String codePays, String codeVoyage, String intitule, int duree,
            String description) {
        JpaUtil.ouvrirTransaction();
        Circuit c = new Circuit(moyenDeTransport, kilometres, codePays,
                codeVoyage, intitule, duree, description);
        Pays pays = PaysDao.findPaysByCodePays(codePays);
        c.setPaysDuVoyage(pays);

        JpaUtil.persist(c);
```

```
        System.out.println(c);

        pays.addVoyage(c);
        JpaUtil.merge(pays);
        JpaUtil.validerTransaction();
    }

    /**
     * Cette méthode créée un Sejour à partir de ses caractéristiques. Le Pays
     * ayant le code pays codePays doit être présent dans la base de donées. Le
     * liens entre le Sejour et le Pays se fait automatiquement. La durée est en
     * jour. Le Sejour est également ajouter à la base de données.
     *
     * @param residence
     * @param codePays
     * @param codeVoyage
     * @param intitule
     * @param duree
     * @param description
     */
    public static void creerSejour(String residence, String codePays,
            String codeVoyage, String intitule, int duree, String description) {
        JpaUtil.ouvrirTransaction();
        Sejour s = new Sejour(residence, codePays, codeVoyage, intitule, duree,
                description);
        Pays pays = PaysDao.findPaysByCodePays(codePays);
        s.setPaysDuVoyage(pays);

        JpaUtil.persist(s);
        System.out.println(s);

        pays.addVoyage(s);
        JpaUtil.merge(pays);
        JpaUtil.validerTransaction();
    }


    /**
     * Cette Méthode affiche la liste de tous les Pays contenus dasn la base de
     * données avec leurs caractéristiques.
     *
     */
    public static void listerTousLesPays() {

        List<Pays> pays = PaysDao.listerPays();
        for (int i = 0; i < pays.size(); i++) {
```

```
            System.out.print(pays.get(i).getNom() + "\n");
        }
    }

    /**
     * Cette Méthode affiche la liste de tous les Voyages (Circuit et Sejour)
     * contenus dasn la base de données avec leurs caractéristiques.
     *
     */
    public static void listerTousLesVoyages() {

        List<Voyage> voyages = VoyageDao.listerVoyages();
        for (int i = 0; i < voyages.size(); i++) {

            System.out.print(voyages.get(i).descriptionPourCatalogue() + "\n");
        }
        if (voyages.isEmpty()) {
            System.out.println("Aucun voyage ");
        }
    }

    /**
     * Cette Méthode affiche la liste de tous les Voyages (Circuits et Sejours)
     * se déroulant dans le Pays nomPays contenus dans la base de données avec
     * leurs caractéristiques. Le Pays ayant pour nom nompays doit être présent
     * dans la base.
     *
     * @param nomPays
     */
    public static void listerVoyagesParPays(String nomPays) {

        List<Voyage> voyages = VoyageDao.findVoyageByNomPays(nomPays);
        for (int i = 0; i < voyages.size(); i++) {


            System.out.print(voyages.get(i).descriptionPourCatalogue() + "\n");
        }
        if (voyages.isEmpty()) {
            System.out.println("Aucun voyage pour le pays " + nomPays);
        }
    }

    /**
     * Cette Méthode affiche la liste de tous les Sejours contenus dans la base
     * de données avec leurs caractéristiques.
     *
```

```
     */
    public static void listerSejours() {

        List<Sejour> voyages = VoyageDao.listerSejours();
        for (int i = 0; i < voyages.size(); i++) {

            System.out.print(voyages.get(i).descriptionPourCatalogue() + "\n");
        }
        if (voyages.isEmpty()) {
            System.out.println("Aucun séjour");
        }
    }

    /**
     * Cette Méthode affiche la liste de tous les Circuits contenus dans la base
     * de données avec leurs caractéristiques.
     *
     */
    public static void listerCircuits() {

        List<Circuit> voyages = VoyageDao.listerCircuits();
        for (int i = 0; i < voyages.size(); i++) {

            System.out.print(voyages.get(i).descriptionPourCatalogue() + "\n");
        }
        if (voyages.isEmpty()) {
            System.out.println("Aucun circuit");
        }
    }

    /**
     * Cette Méthode affiche la liste de tous les Clients contenus dans la base

     * de données avec leurs caractéristiques dont leurs Devis.
     *
     */
    public static void listerClients() {

        List<Client> clients = ClientDao.listerClients();
        for (int i = 0; i < clients.size(); i++) {

            System.out.print(clients.get(i) + "\n"
                    + clients.get(i).listeDevis());
        }

    }
```

```
    /**
     * Cette méthode renvoie le mail envoyé à un partenaireCommerciale pour lui
     * transmettre les informations du Client client sous forme de chaine de
     * caractères. Le Client client doit être présent dans la base de données.
     *
     * @param client
     * @return
     */
    public static String envoyerMailPartenaires(Client client) {

        if (client.isAutorisationPartenaires()) {
            String civilite = client.getCivilite();
            String nom = client.getNom();
            String prenom = client.getPrenom();
            String email = client.getEmail();

            return "Nous sommes heureux de vous prévenir de l'adhésion de "
                    + civilite + " " + nom + " " + prenom
                    + " dont l'adresse électronique est " + email + " .";
        }
        return null;

    }

    /**
     * Cette méthode permet de choisir le Conseiller à l'élaboration d'un Devis
     * d. Le Devis d doit être présent dans la base de données. Le conseiller
     * selectionné est un spécialiste du Pays du voyage où se situe le devis, et
     * il s'occupe du moins de Clients possible. Attention, cette méthode ne
     * valide pas la transaction avec la base de données.

     *
     * @param d
     * @return
     */
    private static boolean choisirConseiller(Devis d) {
        boolean res = false;
        Conseiller cons = DevisDao.choixConseiller(d);
        if (cons != null) {
            System.out.println(cons);
            d.setConseillerDevis(cons);
            if (!cons.getClients().contains(d.getClientDevis())) {
                cons.addClient(d.getClientDevis());
            }
            JpaUtil.merge(d);
```

```
            JpaUtil.merge(cons);
            res = true;
        }

        return res;

    }

    /**
     * Cette Méthode affiche la liste de tous les Circuits ou tous les Sejours
     * se déroulant dans le Pays nomPays contenus dans la base de données avec
     * leurs caractéristiques. Le Pays ayant pour nom nompays doit être présent
     * dans la base. Le type est Sejour ou Circuit.
     *
     * @param nomPays
     * @param type
     */
    public static void listerVoyagesParPaysEtType(String nomPays, String type) {

        if (type.equals("Sejour")) {
            List<Sejour> voyages = VoyageDao.listerSejoursParPays(nomPays);
            for (int i = 0; i < voyages.size(); i++) {

                System.out.print(voyages.get(i).descriptionPourCatalogue()
                        + "\n");
            }
            if (voyages.isEmpty()) {
                System.out.println("Aucun séjour pour le pays " + nomPays);
            }
        } else if (type.equals("Circuit")) {

            List<Circuit> voyages = VoyageDao.listerCircuitsParPays(nomPays);
            for (int i = 0; i < voyages.size(); i++) {

                System.out.print(voyages.get(i).descriptionPourCatalogue()
                        + "\n");
            }
            if (voyages.isEmpty()) {
                System.out.println("Aucun Circuit pour le pays " + nomPays);
            }
        } else {
            System.out.println("Choisissez le type Sejour ou Circuit");
        }
    }

    /**
     * Cette métthode permet de saisir un Client à l'aide du clavier. Les
```

```
     * indications sur les champs à fournir sont données au fur et à mesure.
     *
     */
    public static void SaisirClient() {

        String[] descriptionClient = new String[7];
        System.out.println("Veuillez écrire : \"CIVILITE\" \"NOM\" \"PRENOM\" "
                + "\"AAAA-JJ-MM\" \"ADRESSE\" \"TELEPHONE\" \"EMAIL\" ");

        descriptionClient[0] = Saisie.lireChaine("CIVILITE\n");
        descriptionClient[1] = Saisie.lireChaine("NOM\n");
        descriptionClient[2] = Saisie.lireChaine("PRENOM\n");
        descriptionClient[3] = Saisie.lireChaine("AAAA-MM-JJ\n");
        descriptionClient[4] = Saisie.lireChaine("ADRESSE\n");
        descriptionClient[5] = Saisie.lireChaine("TELEPHONE\n");
        descriptionClient[6] = Saisie.lireChaine("EMAIL\n");

        creerClient(descriptionClient[0], descriptionClient[1],
                descriptionClient[2], descriptionClient[3],
                descriptionClient[4], descriptionClient[5],
                descriptionClient[6]);
    }

    /**
     * Cette métthode permet de saisir un Devis à l'aide du clavier. Les
     * indications sur les champs à fournir sont données au fur et à mesure.
     *
     */

    public static void SaisirDevis() {
        String[] descriptionDevis = new String[4];
        String choixMode = "";
        System.out.println("Identifiez-vous : ");

        descriptionDevis[0] = Saisie.lireChaine("ADDRESSE EMAIL CLIENT\n");
        System.out.println("Choisissez un mode de choix : "
                + "\n Par pays et type (PT) \n"
                + "Par Pays (P) "
                + "\n Par Type (T)\n"
                + "Depuis catalogue entier (C) \n");
        choixMode = Saisie.lireChaine("MODE CHOIX\n");

        if (choixMode.equals("C") || choixMode.equals("c")) {
            listerTousLesVoyages();
            System.out.println("Choisissez un voyage : ");
            descriptionDevis[1] = Saisie.lireChaine("CODE VOYAGE\n");
```

```
            System.out.println("Choix des caractéristiques ");
            descriptionDevis[2] = Saisie.lireChaine("CODE CHOIX\n");
            descriptionDevis[3] = Saisie.lireChaine("NOMBRE PARTICIPANTS\n");
        }
        if (choixMode.equals("P") || choixMode.equals("p")) {
            listerTousLesPays();
            System.out.println("Choisissez un pays : ");
            String pays = Saisie.lireChaine("Nom Pays\n");
            listerVoyagesParPays(pays);
            System.out.println("Choisissez un voyage : ");
            descriptionDevis[1] = Saisie.lireChaine("CODE VOYAGE\n");
            System.out.println("Choix des caractéristiques ");
            descriptionDevis[2] = Saisie.lireChaine("CODE CHOIX\n");
            descriptionDevis[3] = Saisie.lireChaine("NOMBRE PARTICIPANTS\n");
        }
        if (choixMode.equals("T") || choixMode.equals("t")) {
            System.out.println("Choisissez C pour Circuit ou S pour Séjour");
            String type = Saisie.lireChaine("TYPE\n");
            if (type.equals("C") || type.equals("c")) {
                listerCircuits();

            } else if (type.equals("S") || type.equals("s")) {
                listerSejours();
            }
            System.out.println("Choisissez un voyage : ");
            descriptionDevis[1] = Saisie.lireChaine("CODE VOYAGE\n");
            System.out.println("Choix des caractéristiques ");

            descriptionDevis[2] = Saisie.lireChaine("CODE CHOIX\n");
            descriptionDevis[3] = Saisie.lireChaine("NOMBRE PARTICIPANTS\n");

        }
        if (choixMode.equals("PT") || choixMode.equals("pt")
                || choixMode.equals("Pt") || choixMode.equals("pT")) {
            listerTousLesPays();
            System.out.println("Choisissez un pays : ");
            String pays = Saisie.lireChaine("Nom Pays\n");
            System.out.println("Choisissez Circuit ou Sejour");
            String type = Saisie.lireChaine("TYPE\n");
            listerVoyagesParPaysEtType(pays, type);
            System.out.println("Choisissez un voyage : ");
            descriptionDevis[1] = Saisie.lireChaine("CODE VOYAGE\n");
            System.out.println("Choix des caractéristiques ");
            descriptionDevis[2] = Saisie.lireChaine("CODE CHOIX\n");
            descriptionDevis[3] = Saisie.lireChaine("NOMBRE PARTICIPANTS\n");

        }
```

```
        creerDevis(descriptionDevis[1], descriptionDevis[0],
                descriptionDevis[2], descriptionDevis[3]);
    }

    /**
     * Cette méthide revcoie un nombre entier aléatoire en 2 et 5.
     *
     * @return
     */
    private static int ChoisirNbPassager() {
        return Aleatoire.random(2, 5);
    }

    public static String afficheDevis(Devis d) {
        return d.afficheDevis();
    }

    /**
     * Cette méthode renvoie aléatoirement une des Infoprincipale (Départ) du
     * Voyage ayant pour code voyage codeVoyage. Ce dernier doit être présent
     * dans la base de données.
     *
     * @param CodeVoyage
     * @return

     */
    private static InfoPrincipale ChoisirInfoPrincipale(String codeVoyage) {
        List<InfoPrincipale> lInfosPrincipales
                = VoyageDao.listerInfos(codeVoyage);
        if (lInfosPrincipales.size() > 0) {
            int indexInfoPrincipale = Aleatoire.random(0,
                    lInfosPrincipales.size() - 1);
            return lInfosPrincipales.get(indexInfoPrincipale);
        }
        return null;
    }

    /**
     * Cette méthode permet de transformer une chaine de caractère au format
     * "AAAA-MM-JJ" en Date.
     *
     * @param date
     * @return
     */
    private static Date parseDateUsFormat(String date) {
```

```
        try {
            return US_DATE_FORMAT.parse(date);
        } catch (ParseException ex) {
            return new Date();
        }
    }
}
```