CODE TP1 SI : B3330

SOTIR Paul-Emmanuel
CHAPELLE Victoire

Main.java

```java
package TP1_SI.View;

import TP1_SI.DAL.*;

/**
 * Fonction principale démarant la vue de connexion
 * @author B3330
 */
public class Main {

    public static void main(String[] arg) {
        JpaUtil.init();
        JpaUtil.creerEntityManager();

        ConsoleConnexionView connexion_view = new ConsoleConnexionView();
        connexion_view.run();

        JpaUtil.fermerEntityManager();
        JpaUtil.destroy();
    }
}
```

Callalbe.java (utils)

```java
package TP1_SI.Utils;

/**
 * Interface représentant une fonction ne prenant pas de paramètres et retournant
void.
 * Cette interface permet de créer un mécaniqme similaire aux pointeurs de
fonctions.
 * @author B3330
 */
public interface Callable {
    public void call();
}
```

ConsoleUtils (Utils)

```java
package TP1_SI.Utils;

import java.util.List;
import java.io.IOException;
import java.io.BufferedReader;
import java.io.InputStreamReader;

/**
 * Classe regourpant quelques fonction utiles à l'interface console
 * @author B3330
 */
public class ConsoleUtils {

    public static String lireChaine(String invite) {
        String chaineLue = null;
        System.out.print(invite);
        try {
            InputStreamReader isr = new InputStreamReader(System.in);
            BufferedReader br = new BufferedReader(isr);
            chaineLue = br.readLine();
        } catch (IOException exception) {
            exception.printStackTrace(System.err);
        }
        return chaineLue;

    }

    public static Integer lireInteger(String invite) {
        Integer valeurLue = null;
        try {
            valeurLue = new Integer(lireChaine(invite));
        } catch (java.lang.NumberFormatException e) {
            System.out.println("erreur de saisie");
            valeurLue = lireInteger(invite);
        }
        return valeurLue;
    }

    public static Integer lireInteger(String invite, List<Integer>
valeursPossibles) {
        Integer valeurLue = null;
        try {
            valeurLue = new Integer(lireChaine(invite));
            if (!(valeursPossibles.contains(valeurLue))) {
                throw new Exception();
            }
        } catch (Exception e) {
            System.out.println("erreur de saisie");
            valeurLue = lireInteger(invite, valeursPossibles);
        }
        return valeurLue;
    }
}
```

ConsoleMenu (Utils)

```java
package TP1_SI.Utils;

import java.io.InputStreamReader;
import java.io.BufferedReader;
import java.util.HashMap;
import java.util.Map;

/**
 * Classe aidant à la création d'un menu sur console.
 * Un choix est ajouté au menu en fournissant un nom de l'option et une fonction à appeler
 * si cette option est choisie (class implémentant l'interface 'Callable').
 * @author B3330
 */
public class ConsoleMenu {

    /**
     * Constructeur de la classe 'ConsoleMenu'
     * @param menu_name nom du menu
     */
    public ConsoleMenu(String menu_name) {
        menu_choices = new HashMap<>();
        this.menu_name = menu_name;
        this.exit_function = null;
    }

    /**
     * Constructeur de la classe permmettant de specifier une fonction à appeler pour quitter
le menu
     * @param menu_name nom du menu
     * @param exit_function fonction à appeler si l'utilisateur veux quitter le menu
     */
    public ConsoleMenu(String menu_name, Callable exit_function) {
        menu_choices = new HashMap<>();
        this.menu_name = menu_name;
        this.exit_function = exit_function;
    }

    /**
     * Ajoute un choix au menu
     * @param function fonction à appeler si le l'option est choisie
     * @param choice_name nom de l'option affiché dans le menu
     */
    public void addChoice(Callable function, String choice_name) {
        menu_choices.put(menu_choices.size(), new menu_choice(choice_name, function));
    }

    /**
     * Affiche le menu
     */
    public void runMenu() {
        while (true) {
            System.out.println("\n############### " + menu_name + " ###############");
            int i = 0;
            for (Map.Entry<Integer, menu_choice> entry : menu_choices.entrySet()) {
                System.out.println("\t" + i + ":\t" + entry.getValue().name);
                i++;
            }

            if(exit_function != null)
```

```
                System.out.println("\t  \tType any other number to exit menu");

            while(true)
            {
                int num = ConsoleUtils.lireInteger("Enter your choice: ");
                if(num >= menu_choices.size() || num < 0)
                {
                    if(exit_function != null)
                    {
                        System.out.println();
                        exit_function.call();
                        return;
                    }
                }
                else
                {
                    menu_choice choice = menu_choices.get(num);
                    System.out.println();
                    choice.callback.call();
                    break;
                }
            }
        }
    }

    private static class menu_choice
    {
        public menu_choice(String name, Callable callback)
        {
            this.name = name;
            this.callback = callback;
        }

        public String name;
        public Callable callback;
    }

    private HashMap<Integer, menu_choice> menu_choices;
    private Callable exit_function;
    private String menu_name;
}
```

Location (métier.model)

```java
package TP1_SI.metier.model;

import com.google.maps.model.LatLng;

import javax.persistence.Id;
import java.io.Serializable;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;

/**
 * Classe représentant un lieux définit par des coordonées, une addresse, un nom et une
 * description.
 * @author B3330
 */
@Entity
public class Location implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String address;
    private String description;
    private String denomination;
    private Double latitude;
    private Double longitude;

    public Location() { }

    public Location(String denomination, String description, String address) {
        this.denomination = denomination;
        this.description = description;
        this.address = address;
    }

    public Long getId() {
        return id;
    }

    public String getDenomination() {
        return denomination;
    }

    public String getDescription() {
        return description;
    }

    public String getAddress() {
        return address;
    }

    public Double getLongitude() {
        return longitude;
    }

    public Double getLatitude() {
        return latitude;
    }

    public void setCoordonnees(LatLng latLng) {
```

```
        this.longitude = latLng.lng;
        this.latitude = latLng.lat;
    }

    public void setDenomination(String denomination) {
        this.denomination = denomination;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public void setAddress(String adresse) {
        this.address = adresse;
    }

    public void setLongitude(Double longitude) {
        this.longitude = longitude;
    }

    public void setLatitude(Double latitude) {
        this.latitude = latitude;
    }

    @Override
    public int hashCode() {
        return (id != null ? id.hashCode() : 0);
    }

    @Override
    public boolean equals(Object object) {
        if (!(object instanceof Location))
            return false;

        Location other = (Location) object;
        if ((this.id == null && other.id != null) || (this.id != null &&
!this.id.equals(other.id)))
            return false;
        return true;
    }

    @Override
    public String toString() {
        return "Location{" + "id=" + id + ", denomination=" + denomination + ", description="
+ description + ", address=" + address + ", longitude=" + longitude + ", latitude=" +
latitude + '}';
    }
}
```

Member (métier.model)

```java
package TP1_SI.metier.model;

import com.google.maps.model.LatLng;

import javax.persistence.Id;
import java.io.Serializable;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;

/**
 * Classe représentant un membre de l'association pouvant rejoindre ou créer des événements
 * @author B3330
 */
@Entity
public class Member implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String address;
    private Double lat;
    private Double lng;
    private String mail;
    private String nom;
    private String prenom;

    public Member() { }

    public Member(String nom, String prenom, String address, String mail, LatLng coordonnees)
{
        this.nom = nom;
        this.prenom = prenom;
        this.mail = mail;
        this.address = address;
        this.lat = coordonnees.lat;
        this.lng = coordonnees.lng;
    }

    public Long getId() {
        return id;
    }

    public String getNom() {
        return nom;
    }

    public String getPrenom() {
        return prenom;
    }

    public String getMail() {
        return mail;
    }

    public String getAddress() {
        return address;
    }

    public Double getLatitude() {
```

```
        return lat;
    }

    public Double getLongitude() {
        return lng;
    }

    public void setNom(String nom) {
        this.nom = nom;
    }

    public void setPrenom(String prenom) {
        this.prenom = prenom;
    }

    public void setMail(String mail) {
        this.mail = mail;
    }

    public void setAddress(String adresse) {
        this.address = adresse;
    }

    public void setCoordonnees(LatLng latLng) {
        this.lat = latLng.lat;
        this.lng = latLng.lng;
    }

    @Override
    public int hashCode() {
        return (id != null ? id.hashCode() : 0);
    }

    @Override
    public boolean equals(Object object) {
        if (!(object instanceof Member))
            return false;

        Member other = (Member) object;
        if ((this.id == null && other.id != null) || (this.id != null &&
!this.id.equals(other.id)))
            return false;
        return true;
    }

    @Override
    public String toString() {
        return "Member{" + "id=" + id + ", nom=" + nom + ", prenom=" + prenom + ", mail=" +
mail + ", adresse=" + address + ", longitude=" + lng + ", latitude=" + lat + '}';
    }
}
```

Event (métier.model)

```java
package TP1_SI.metier.model;

import java.sql.Date;
import java.util.ArrayList;
import java.util.List;
import java.io.Serializable;
import javax.persistence.Id;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;

/**
 * Classe représentant un événement constitué de membres voulant participer à cet événement.
 * Un événement peut ne pas être encore assigné à un lieu (location == null)
 *
 * @author B3330
 */
@Entity
public class Event implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private Date date;
    private Activity activity;
    private List<Member> members;
    private Location location;
    private boolean complet;

    public Event() {
    }

    public Event(Date date, Activity activity, Member creator) {
        this.activity = activity;
        this.date = date;
        this.members = new ArrayList<>();
        this.members.add(creator);
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public Date getDate() {
        return date;
    }

    public void setDate(Date date) {
        this.date = date;
    }

    public Activity getActivity() {
        return activity;
    }

    public void setActivity(Activity activity) {
        this.activity = activity;
    }

    public List<Member> getMembers() {
        return members;
    }
```

```
    public void setMembers(List<Member> members) {
        this.members = members;
    }

    public Location getLocation() {
        return location;
    }

    public void setLocation(Location location) {
        this.location = location;
    }

    public boolean getComplet() {
        return this.complet;
    }

    public void setComplet() {
        this.complet = true;
    }

    public List<Member> getAdherentsEquipe1() {
        if (activity.isByTeam()) {
            List<Member> equipe_1 = null;
            for (int i = 0; i < this.members.size(); i += 2) {
                equipe 1.add(members.get(i));
            }
            return equipe 1;
        }
        return null;
    }

    public List<Member> getAdherentsEquipe2() {
        if (activity.isByTeam()) {
            List<Member> equipe_2 = null;
            for (int i = 1; i < this.members.size(); i += 2) {
                equipe_2.add(members.get(i));
            }
            return equipe_2;
        }
        return null;
    }

    @Override
    public int hashCode() {
        return (id != null ? id.hashCode() : 0);
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof Event)) {
            return false;
        }
        Event other = (Event) object;
        if ((this.id == null && other.id != null) || (this.id != null && !this.id.equals(other.id))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "Event{ id=" + id + ", date=" + date + ", location=" + location + ", is_complet=" +
          complet + " }";
    }

}
```

Activity (métier.model)

```java
package TP1_SI.metier.model;

import java.io.Serializable;
import javax.persistence.Id;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;

/**
 * Classe représentatn une activité.
 * Une activité peut être partiquée par une ou deux équipes (booléen 'byTeam').
 * @author B3330
 */
@Entity
public class Activity implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String name;
    private Integer nbParticipants;
    private Boolean byTeam;

    public Activity() { }

    public Activity(String name, Boolean byTeam, Integer nbParticipants) {
        this.name = name;
        this.byTeam = byTeam;
        this.nbParticipants = nbParticipants;
    }

    public Long getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public Boolean isByTeam() {
        return byTeam;
    }

    public Integer getNbParticipants() {
        return nbParticipants;
    }

    public void setName(String denomination) {
        this.name = denomination;
    }

    public void setByTeam(Boolean parEquipe) {
        this.byTeam = parEquipe;
    }

    public void setNbParticipants(Integer nbParticipants) {
        this.nbParticipants = nbParticipants;
    }

    @Override
```

```
    public int hashCode() {
        return (id != null ? id.hashCode() : 0);
    }

    @Override
    public boolean equals(Object object) {
        if (!(object instanceof Activity))
            return false;

        Activity other = (Activity) object;
        if ((this.id == null && other.id != null) || (this.id != null &&
                !this.id.equals(other.id)))
            return false;
        return true;
    }

    @Override
    public String toString() {
        return "Activity{" + "id=" + id + ", denomination=" + name + ", byTeam=" + byTeam +
        ", nbParticipants=" + nbParticipants + '}';
    }
}
```

LocationDAL (DAL)

```java
package TP1_SI.DAL;

import java.util.List;
import javax.persistence.Query;
import javax.persistence.EntityManager;

import TP1_SI.metier.model.Location;

/**
 * Data Access Layer permettant d'obtenir, de créer et modifier des instances de la classe
'Location'.
 * Utilise JPA pour persiter les Lieux.
 * @author B3330
 */
public class LocationDAL {

    public void create(Location location) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        em.persist(location);
    }

    public Location update(Location location) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        return em.merge(location);
    }

    public Location findById(long id) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        return em.find(Location.class, id);
    }

    public List<Location> findAll() throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        Query q = em.createQuery("SELECT l FROM Location l");
        return (List<Location>) q.getResultList();
    }
}
```

MemberDAL (DAL)

```java
package TP1_SI.DAL;

import java.util.List;
import javax.persistence.Query;
import javax.persistence.EntityManager;

import TP1_SI.metier.model.Member;

/**
 * Data Access Layer permettant d'obtenir, de créer et modifier des instances de la classe
'Member'.
 * Utilise JPA pour persiter les Membres.
 * @author B3330
 */
public class MemberDAL {

    public void create(Member member) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        em.persist(member);
    }

    public Member update(Member member) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        return em.merge(member);
    }

    public Member findById(long id) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        return em.find(Member.class, id);
    }

    public Member findByMail(String mail) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        Query q = em.createQuery("SELECT a FROM Member a WHERE a.mail='" + mail + "'");
        List<Member> members = (List<Member>) q.getResultList();

        if (members.size() >= 1)
            return members.get(0);
        return null;
    }

    public List<Member> findAll() throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        Query q = em.createQuery("SELECT a FROM Member a");
        return (List<Member>)q.getResultList();
    }
}
```

EventDAL (DAL)

```java
package TP1_SI.DAL;

import javax.persistence.EntityManager;
import javax.persistence.Query;
import java.util.List;
import java.sql.Date;

import TP1_SI.metier.model.Member;
import TP1_SI.metier.model.Event;

/**
 * Data Access Layer permettant d'obtenir, de créer et modifier des instances de la classe 'Event'.
 * Utilise JPA pour persiter les Evenements.
 * @author B3330
 */
public class EventDAL {

    public void create(Event event) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        em.persist(event);
    }

    public Event update(Event event) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        return em.merge(event);
    }

    public Event findById(long id) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        return em.find(Event.class, id);
    }

    public void AddAdherentToEvent(Event event, Member member) throws Throwable {
        event.getMembers().add(member);
        update(event);
        if (event.getMembers().size() == event.getActivity().getNbParticipants())
            event.setComplet();
    }

    public List<Event> findAll() throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        Query q = em.createQuery("SELECT a FROM Event a");
        return (List<Event>)q.getResultList();
    }

    public List<Event> findDispo() throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        Query q = em.createQuery("SELECT a FROM Event a WHERE a.complet = FALSE");
        return (List<Event>)q.getResultList();
    }

    public List<Event> findByMember(long member_id) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        Query q = em.createQuery("SELECT e FROM Event e INNER JOIN e.members event member WHERE
event_member.id = " + member_id);
        return (List<Event>)q.getResultList();
    }

    public List<Event> findByDate(Date date) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        Query q = em.createQuery("SELECT a FROM Event a WHERE a.date = " + date);
        return (List<Event>)q.getResultList();
    }

}
```

ActivityDAL (DAL)

```java
package TP1_SI.DAL;

import java.util.List;
import javax.persistence.Query;
import javax.persistence.EntityManager;

import TP1_SI.metier.model.Activity;

/**
 * Data Access Layer permettant d'obtenir, de créer et modifier des instances de la classe
'Activity'.
 * Utilise JPA pour persiter les Activitées.
 * @author B3330
 */
public class ActivityDAL {

    public void create(Activity activity) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        em.persist(activity);
    }

    public Activity update(Activity activity) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        return em.merge(activity);
    }

    public Activity findById(long id) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        return em.find(Activity.class, id);
    }

    public List<Activity> findAll() throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        Query q = em.createQuery("SELECT a FROM Activity a");
        return (List<Activity>) q.getResultList();
    }
}
```

ServiceResult (metier.service)

```java
package TP1_SI.metier.service;

import java.io.Serializable;

/**
 * Classe générique représentant le résultat d'un appel à une méthode d'un service.
 * Cette classe peut être utilisé comme type de retour pour fournir à la fois un résultat de
 * type 'Result_t'
 * et un éventuel code d'erreur appartenant à l'énumératrion 'ErrorEnum_t'.
 * Ce méchanisme peut être préférable à l'utilisation d'exeptions car ficilite l'ajout
 * d'informations sur une erreur
 * et peut être sérialisé (ce qui peut faciliter la création de services REST en serialisant
 * directement en JSON cette
 * classe par exemple ;) )
 * @param <Result_t> type du résultat de l'appel
 * @param <ErrorEnum_t> type des codes d'erreur
 * @author B3330
 */
public class ServiceResult<Result_t, ErrorEnum_t> implements Serializable {

    public ServiceResult(Result_t result, ErrorEnum_t error) {
        this.result = result;
        this.error = error;
    }

    @Override
    public int hashCode() {
        return (result != null ? result.hashCode() : 0);
    }

    @Override
    public boolean equals(Object object) {
        if (!(object instanceof ServiceResult))
            return false;

        ServiceResult other = (ServiceResult)object;

        if ((this.result == null && other.result != null) || (this.result != null &&
!this.result.equals(other.result)))
            return false;
        if ((this.error == null && other.error != null) || (this.error != null &&
!this.error.equals(other.error)))
            return false;
        return true;
    }

    @Override
    public String toString() {
        if(result != null)
            return "ServiceResult{ result = '" + result + "', error_code = '" + error + "'}";
        else
            return "ServiceResult{ error_code = '" + error + "'}";
    }

    public Result_t result;
    public ErrorEnum_t error;
}
```

ServiceTechnique (metier.service)

```java
package TP1_SI.metier.service;

import TP1_SI.metier.model.Member;
import TP1_SI.metier.model.Event;

import com.google.maps.DistanceMatrixApi;
import com.google.maps.GeoApiContext;
import com.google.maps.GeocodingApi;
import com.google.maps.model.DistanceMatrix;
import com.google.maps.model.GeocodingResult;
import com.google.maps.model.LatLng;

import java.util.List;

/**
 * Classe regroupant les méthodes des services techniques (services permettant l'envoi de
mail et
 * la géolocalisation via l'API google maps).
 * @author B3330
 */
public class ServiceTechnique {

    /*private static final String SMTP_HOST1 = "";
    private static final String LOGIN_SMTP1 = "paul-emmanuel.sotir@insa-lyon.fr";
    private static final String IMAP_ACCOUNT1 = "";*/
    private static final String GeoAPIKey = "AIzaSyDcVVJjfmxsNdbdUYeg9MjQoJJ6THPuap4";

    /**
     * Envoie un mail de confirmation d'inscription d'un membre
     * @param member membre venant de s'inscrire
     */
    public static void SendSuccessfullInscriptionMail(Member member) {
        /*Properties properties = new Properties();
        properties.setProperty("mail.transport.protocol", "smtp");
        properties.setProperty("mail.smtp.host", SMTP_HOST1);
        properties.setProperty("mail.smtp.user", LOGIN_SMTP1);
        properties.setProperty("mail.from", IMAP_ACCOUNT1);
        Session session = Session.getInstance(properties);*/
        String corps = "Bonjour " + member.getPrenom() + ",\n" +
                "Nous vous confirmons votre adhésion à l'association COLLECT'IF. Votre numéro
d'adhérent est : " + member.getId();

        System.out.println(corps);
    }

    /**
     * Envoie un mail expliquant que l'inscription d'un membre a échouée
     * @param member membre voulant s'inscrire
     */
    public static void SendFailedInscriptionMail(Member member) {
        String corps = "Bonjour " + member.getPrenom() + ",\n" +
                "Votre adhésion à l'association COLLECT'IF a malencontreusement échoué...
Merci de recommencer ultérieurement.";

        System.out.println(corps);
    }

    /**
     * Envoie un mail pour notifier les membres d'un evenement que l'évenement a été assigné
```

```
à un lieu
     * @param event Evenement ayant été assigné à un lieu
     */
    public static void SendEventMail(Event event) {
        final LatLng coord = new LatLng(event.getLocation().getLatitude(),
event.getLocation().getLongitude());
        List<Member> members = event.getMembers();

        for (int i = 0; i < members.size(); i++) {
            try {
                long distance = Distance(coord, new LatLng(members.get(i).getLatitude(),
members.get(i).getLongitude()));

                String corps = "Bonjour" + members.get(i).getPrenom() + ",\n" +
                        "Comme vous l'aviez souhaité, COLLECT'IF organise un évènement de " +
event.getActivity().getName() + " le " + event.getDate() + ".\n" +
                        "Vous trouverez ci-dessous les détails de cet évènement.\n" +
                        "Associativement vôtre,\n" +
                        "Le Responsable de l'Association\n" +
                        "Evénement : " + event.getActivity().getName() + "\n" +
                        "Date : " + event.getDate() + "\n" +
                        "Location : " + event.getLocation().getDenomination() + ", " +
event.getLocation().getAddress() + "\n" +
                        "(à " + distance + " km de chez vous)\n" +
                        "Vous jouerez avec :\n";
                if (event.getActivity().isByTeam()) {
                    List<Member> equipe_1 = event.getAdherentsEquipe1();
                    for (int j = 0; j < equipe_1.size(); j++) {
                        if (equipe_1.get(j) != members.get(i))
                            corps += equipe_1.get(j).getPrenom() + " " +
equipe_1.get(j).getNom() + "\n";
                    }
                    corps += "Contre :\n";

                    List<Member> equipe_2 = event.getAdherentsEquipe2();
                    for (int j = 0; j < equipe_2.size(); j++) {
                        if (equipe_2.get(j) != members.get(i))
                            corps += equipe_2.get(j).getPrenom() + " " +
equipe_2.get(j).getNom() + "\n";
                    }

                } else {
                    for (int j = 0; j < members.size(); j++) {
                        if (j != i)
                            corps += members.get(j).getPrenom() + " " +
members.get(j).getNom() + "\n";
                    }
                }

                // TODO: afficher plus d'infos avec une mise en forme
                System.out.println(corps);
            } catch (Exception e) {
                //TODO: gerer ca
            }
        }
    }

    /**
     * Donne la distance entre deux positions
     * @param position1 coordonnées de la première position
     * @param position2 coordonnées de la seconde position
     * @return la distance entre 'position1' et 'position2'
     * @throws Exception pouvant être lancée par l'API google maps
```

```
     */
    public static long Distance(LatLng position1, LatLng position2) throws Exception {
        GeoApiContext context = new GeoApiContext().setApiKey(GeoAPIKey);

        String[] origins = {position1.toUrlValue()};
        String[] dest = {position2.toUrlValue()};
        DistanceMatrix result = DistanceMatrixApi.getDistanceMatrix(context, origins,
dest).await();

        return result.rows[0].elements[0].distance.inMeters;
    }

    /**
     * Retourne les coordonées d'un adresse donnée grâce à l'API google maps.
     * @param address Addresse sous forme de chaine de charactères
     * @return les coordonées de l'adresse
     * @throws Exception pouvant être lancée par l'API google maps
     */
    public static LatLng GetLatLngFromAddress(String address) throws Exception {
        GeoApiContext context = new GeoApiContext().setApiKey(GeoAPIKey);
        GeocodingResult[] results = GeocodingApi.geocode(context, address).await();

        return results[0].geometry.location;
    }
}
```

Services (metier.service)

```java
package TP1_SI.metier.service;

import java.sql.Date;
import java.util.List;

import TP1_SI.DAL.*;
import com.google.maps.model.LatLng;

import TP1_SI.metier.model.Event;
import TP1_SI.metier.model.Member;
import TP1_SI.metier.model.Activity;
import TP1_SI.metier.model.Location;

/**
 * Classe regroupant les services metier de l'application COLLECT'IF.
 * @author B3330
 */
public class Services {

    /**
     * Enumération des codes d'erreur pouvant se produire lors de la connexion/inscription
     * d'un member
     */
    public enum ConnexionError {
        OK, EMPTY_NAME, EMPTY_FIRSTNAME, EMPTY_ADDRESS, EMPTY_EMAIL, WRONG_ADDRESS,
        BAD_EMAIL, DATABASE_ERROR
    }

    /**
     * Enumération des codes d'erreur pouvant se produire lors de l'appel aux autres services
     * (voir type de retour dez méthodes ci-dessous)
     */
    public enum Request_Error {
        OK, WRONG_EVENT_ID, WRONG_LIEU_ID, WRONG_MEMBER_ID, WRONG_ACTIVITY_ID, FULL_EVENT,
        DATABASE_ERROR
    }

    /**
     * Methode permettant la connexion d'un membre à partir de son email
     * @param mail email du membre
     * @return retourne le membre connecté ou un code d'erreur détaillant les raisons de
     * l'echec de la connexion (OK, DATABASE_ERROR ou BAD_EMAIL)
     */
    public static ServiceResult<Member, ConnexionError> Connexion(String mail) {
        try {
            MemberDAL adherent_dao = new MemberDAL();
            Member member = adherent_dao.findByMail(mail);
            if (member == null)
                return new ServiceResult<Member, ConnexionError>(null,
                                                                 ConnexionError.BAD_EMAIL);
            return new ServiceResult<Member, ConnexionError>(member, ConnexionError.OK);
        } catch (Throwable e) {
            return new ServiceResult<Member, ConnexionError>(null,
                                                             ConnexionError.DATABASE_ERROR);
        }
    }

    /**
     * Inscrit un membre si possible.
     * @param nom nom du membre
```

```
     * @param prenom prénom du membre
     * @param mail email du nouveau membre (doit être unique)
     * @param address addresse du nouveau membre. L'API google maps doit pouvoir trouver les
     * coordonées de l'addresse pour que l'inscription ait lieu
     * @return retourne le membre connecté ou un code d'erreur détaillant les raisons de
     * l'echec de l'inscription (OK, EMPTY_FIRSTNAME, EMPTY_ADDRESS,
     * EMPTY_EMAIL, WRONG_ADDRESS, BAD_EMAIL, DATABASE_ERROR ou WRONG_ADDRESS).
     */
    public static ServiceResult<Member, ConnexionError> Inscription(String nom, String
                                                    prenom, String mail, String address) {
        if(prenom == null || prenom.isEmpty())
            return new ServiceResult<Member, ConnexionError>(null,
                                                    ConnexionError.EMPTY_FIRSTNAME);
        if(mail == null || mail.isEmpty())
            return new ServiceResult<Member, ConnexionError>(null,
                                                    ConnexionError.EMPTY_EMAIL);
        if(nom == null || nom.isEmpty())
            return new ServiceResult<Member, ConnexionError>(null,
                                                    ConnexionError.EMPTY_NAME);
        if(address == null || address.isEmpty())
            return new ServiceResult<Member, ConnexionError>(null,
                                                    ConnexionError.EMPTY_ADDRESS);

        try {
            LatLng coords = ServiceTechnique.GetLatLngFromAddress(address);
            Member member = new Member(nom, prenom, address, mail, coords);
            JpaUtil.ouvrirTransaction();
            MemberDAL adherent_dao = new MemberDAL();

            try {
                if (adherent_dao.findByMail(mail) != null) {
                    JpaUtil.annulerTransaction();
                    return new ServiceResult<Member, ConnexionError>(null,
                                                    ConnexionError.BAD_EMAIL);
                }
                adherent_dao.create(member);
                JpaUtil.validerTransaction();
                ServiceTechnique.SendSuccessfullInscriptionMail(member);
                return new ServiceResult<Member, ConnexionError>(member, ConnexionError.OK);
            } catch (Throwable e) {
                ServiceTechnique.SendFailedInscriptionMail(member);
                JpaUtil.annulerTransaction();
                return new ServiceResult<Member, ConnexionError>(null,
                                                    ConnexionError.DATABASE_ERROR);
            }
        } catch (Exception e) {
            JpaUtil.annulerTransaction();
            return new ServiceResult<Member, ConnexionError>(null,
                                                    ConnexionError.WRONG_ADDRESS);
        }
    }

    /**
     * Liste les événements d'une date données
     * @param date date, au jour près, à la quelle ont cherche les événements
     * @return la liste des événements ou un code d'erreur détaillant les raison pour les
     * quelles la méthode a échouée (OK, DATABASE_ERROR)
     */
    public static ServiceResult<List<Event>, Request_Error> ListEventsOfDate(Date date) {
        EventDAL event_dao = new EventDAL();
        try {
            return new ServiceResult<List<Event>, Request_Error>(event_dao.findByDate(date),
```

```
                                                        Request_Error.OK);
        } catch (Throwable e) {
            return new ServiceResult<List<Event>, Request_Error>(null,
                                                        Request_Error.DATABASE_ERROR);
        }
    }

    /**
     * Liste tout les évenements
     * @return la liste des événements ou un code d'erreur détaillant les raison pour les
     * quelles la méthode a échouée (OK, DATABASE_ERROR)
     */
    public static ServiceResult<List<Event>, Request_Error> ListAllEvents() {
        try {
            EventDAL event_dao = new EventDAL();
            return new ServiceResult<List<Event>, Request_Error>(event_dao.findAll(),
                                                        Request_Error.OK);
        } catch (Throwable e) {
            return new ServiceResult<List<Event>, Request_Error>(null,
                                                        Request_Error.DATABASE_ERROR);
        }
    }

    /**
     * Assigne un lieu (location) à un évenement
     * @param location_id id du lieu
     * @param event_id id de l'évenement au quel assigner le lieu
     * @return Un code d'erreur indiquant si l'assignation s'est déroullée correctement
     * (DATABASE_ERROR, WRONG_LIEU_ID, WRONG_EVENT_ID ou OK)
     */
    public static Request_Error AssignLocationToEvent(long location_id, long event_id) {
        try {
            EventDAL event_dao = new EventDAL();
            Event event = event_dao.findById(event_id);
            if (event == null)
                return Request_Error.WRONG_EVENT_ID;

            LocationDAL lieu_dao = new LocationDAL();
            Location location = lieu_dao.findById(location_id);
            if (location == null)
                return Request_Error.WRONG_LIEU_ID;

            JpaUtil.ouvrirTransaction();
            event.setLocation(location);
            event_dao.update(event);
            if (event.getComplet())
                ServiceTechnique.SendEventMail(event);

            JpaUtil.validerTransaction();
            return Request_Error.OK;
        } catch (Throwable e) {
            JpaUtil.annulerTransaction();
            return Request_Error.DATABASE_ERROR;
        }
    }

    /**
     * Methode permettant à un membre de rejoindre un évenement
     * @param member_id id du membre
     * @param event_id id de l'évenement à rejoindre
     * @return Un code d'erreur indiquant si le membre a pû rejoindre l'évenement
     * (DATABASE_ERROR, FULL_EVENT, WRONG_EVENT_ID, WRONG_MEMBER_ID ou OK)
     */
```

```
public static Request_Error JoinEvent(long member_id, long event_id) {
    try {
        EventDAL event_dao = new EventDAL();
        Event event = event_dao.findById(event_id);
        if (event == null)
            return Request_Error.WRONG_EVENT_ID;
        if (event.getComplet())
            return Request_Error.FULL_EVENT;

        MemberDAL member_dao = new MemberDAL();
        Member member = member_dao.findById(member_id);
        if (member == null)
            return Request_Error.WRONG_MEMBER_ID;

        JpaUtil.ouvrirTransaction();
        event_dao.AddAdherentToEvent(event, member);
        if (event.getComplet())
            ServiceTechnique.SendEventMail(event);

        JpaUtil.validerTransaction();
        return Request_Error.OK;
    } catch (Throwable e) {
        JpaUtil.annulerTransaction();
        return Request_Error.DATABASE_ERROR;
    }
}

/**
 * Crée un évenement au quel participera son créateur
 * @param member_id id du créateur de l'évenement
 * @param activity_id id de l'activité de l'évenement à creer
 * @param date data à laquelle auras lieu l'evenement
 * @return L'évenement crée ou un code d'erreur (OK, WRONG_ACTIVITY_ID, WRONG_MEMBER_ID
 * ou DATABASE_ERROR)
 */
public static ServiceResult<Event, Request_Error> CreateEvent(long member_id, long
                                                              activity_id, Date date) {
    try {
        ActivityDAL activity_dao = new ActivityDAL();
        Activity activity = activity_dao.findById(activity_id);
        if (activity == null)
            return new ServiceResult<Event, Request_Error>(null,
                                                           Request_Error.WRONG_ACTIVITY_ID);

        MemberDAL member_dao = new MemberDAL();
        Member member = member_dao.findById(member_id);
        if (member == null)
            return new ServiceResult<Event, Request_Error>(null,
                                                           Request_Error.WRONG_MEMBER_ID);

        JpaUtil.ouvrirTransaction();
        EventDAL event_dao = new EventDAL();
        Event event = new Event(date, activity, member);
        event_dao.create(event);
        JpaUtil.validerTransaction();

        return new ServiceResult<Event, Request_Error>(event, Request_Error.OK);
    } catch (Throwable e) {
        JpaUtil.annulerTransaction();
        return new ServiceResult<Event, Request_Error>(null,
                                                       Request_Error.DATABASE_ERROR);
    }
}
```

```
    /**
     * Liste les évenements aux quels participe un membre donné en paramètre
     * @param member_id id du membre
     * @return La liste des évnenements ou un code d'erreur (OK ou DATABASE_ERROR) (renvoie
     * DATABASE_ERROR si l'id du membre est mauvais)
     */
    public static ServiceResult<List<Event>, Request_Error> ListEventsOfMember(long
                                                                                member_id) {
        try {
            EventDAL event_dao = new EventDAL();
            List<Event> events = event_dao.findByMember(member_id);
            return new ServiceResult<List<Event>, Request_Error>(events, Request_Error.OK);
        } catch (Throwable e) {
            return new ServiceResult<List<Event>, Request_Error>(null,
                                                    Request_Error.DATABASE_ERROR);
        }
    }

    /**
     * Liste les évenements qui ne sont pas complets (évenements pouvant acceuillir encore
     * des participants)
     * @return La liste des évnenements ou un code d'erreur (OK ou DATABASE_ERROR)
     */
    public static ServiceResult<List<Event>, Request_Error> ListAvailableEvents() {
        try {
            EventDAL event_dao = new EventDAL();
            return new ServiceResult<List<Event>, Request_Error>(event_dao.findDispo(),
                                                        Request_Error.OK);
        } catch (Throwable e) {
            return new ServiceResult<List<Event>, Request_Error>(null,
                                                    Request_Error.DATABASE_ERROR);
        }
    }

    /**
     * Donne la liste des activitées possibles
     * @return La liste de toutes les activitées ou un code d'erreur (OK ou DATABASE_ERROR)
     */
    public static ServiceResult<List<Activity>, Request_Error> ListActivities() {
        try {
            ActivityDAL activity_dao = new ActivityDAL();
            return new ServiceResult<List<Activity>, Request_Error>(activity_dao.findAll(),
                                                        Request_Error.OK);
        } catch (Throwable e) {
            return new ServiceResult<List<Activity>, Request_Error>(null,
                                                    Request_Error.DATABASE_ERROR);
        }
    }

    /**
     * Donne la liste de tout les lieux possibles
     */
    public static ServiceResult<List<Location>, Request_Error> ListLocations() {
        try {
            LocationDAL Location_dao = new LocationDAL();
            return new ServiceResult<List<Location>, Request_Error>(Location_dao.findAll(),
                                                        Request_Error.OK);
        } catch (Throwable e) {
            return new ServiceResult<List<Location>, Request_Error>(null,
                                                    Request_Error.DATABASE_ERROR);
        }
    }
}
```

ConsoleConnexionView (View)

```java
package TP1_SI.View;

import TP1_SI.Utils.Callable;
import TP1_SI.Utils.ConsoleMenu;
import TP1_SI.Utils.ConsoleUtils;
import TP1_SI.metier.model.Member;
import TP1_SI.metier.service.Services;
import TP1_SI.metier.service.ServiceResult;
import TP1_SI.metier.service.Services.ConnexionError;

import java.util.Scanner;

/**
 * Classe gèrant la vue console de connexion.
 * L'utilisateur peut y creer un compte ou se connecter avec une compte existant.
 * @author B3330
 */
public class ConsoleConnexionView {
    static final String admin_email = "admin.collectif@insa-lyon.fr";

    /**
     * Lance la vue
     */
    public void run() {
        ConsoleMenu connexion_menu = new ConsoleMenu("CONNEXION MENU", new Quit());
        connexion_menu.addChoice(new Login(), "Log in");
        connexion_menu.addChoice(new Signin(), "Sign in");

        connexion_menu.runMenu();
    }

    /**
     * Callback gèrant la connexion avezc un coimpte existant
     */
    private static class Login implements Callable
    {
        @Override
        public void call() {
            System.out.println("### Connexion ###");
            String email = ConsoleUtils.lireChaine("Entrez votre mail : ");
            if(!admin_email.equals(email))
            {
                ServiceResult<Member, ConnexionError> newConnexion =
                                                        Services.Connexion(email);
                if (newConnexion.error == ConnexionError.OK)
                    goto_espace_adherents(newConnexion.result);
                else
                    // TODO: gèrer l'erreur en fonction de 'newConnexion.error'
                    System.out.println("Erreur de connexion : " + newConnexion.error);
            }
            else
                goto_dashboard();
        }
    }

    /**
     * Callback gèrant la création de comptes
     */
    private static class Signin implements Callable
```

```
    {
        @Override
        public void call() {
            System.out.println("### Inscription ###");
            String nom = ConsoleUtils.lireChaine("Entrez votre Nom : ");
            String prenom = ConsoleUtils.lireChaine("Entrez votre Prenom : ");
            String email = ConsoleUtils.lireChaine("Entrez votre Email : ");
            String address = ConsoleUtils.lireChaine("Entrez votre Adresse : ");
            if(!admin_email.equals(email))
            {
                ServiceResult<Member, ConnexionError> newInscription =
                                        Services.Inscription(nom, prenom, email, address);
                if (newInscription.error == ConnexionError.OK)
                    goto_espace_adherents(newInscription.result);
                else
                    // TODO: gèrer l'erreur en fonction de 'newInscription.error'
                    System.out.println("Erreur d'inscription : " + newInscription.error);
            }
            else
                System.out.println("Erreur d'inscription : " + ConnexionError.BAD_EMAIL);
        }
    }

    /**
     * Callback appelée si l'utilisateur veux quitter l'application
     */
    private static class Quit implements Callable {
        @Override
        public void call() {
            System.out.println("\n\nExiting application...");
        }
    }

    /**
     * Navigue vers la vue 'espace adherent' (MemberView)
     * @param logged_member Membre s'étant coinnecté avec succès
     */
    private static void goto_espace_adherents(Member logged_member) {
        System.out.println("########################################\n");
        ConsoleMemberView new_view = new ConsoleMemberView(logged_member);
        new_view.run();
    }

    /**
     * Navigue vers la vue dashboard de l'administrateur
     */
    private static void goto_dashboard() {
        System.out.println("########################################\n");
        ConsoleDashboardView new_view = new ConsoleDashboardView();
        new_view.run();
    }
}
```

ConsoleMemberView (View)

```
package TP1_SI.View;

import TP1_SI.Utils.Callable;
import TP1_SI.Utils.ConsoleMenu;
import TP1_SI.Utils.ConsoleUtils;
import TP1_SI.metier.model.Activity;
import TP1_SI.metier.model.Event;
import TP1_SI.metier.model.Member;
import TP1_SI.metier.service.Services;
import TP1_SI.metier.service.ServiceResult;
import TP1_SI.metier.service.Services.Request_Error;

import java.sql.Date;
import java.util.List;

/**
 * @author B3330
 */
public class ConsoleMemberView {
    public ConsoleMemberView(Member logged_member) {
        this.logged_member = logged_member;
    }

    /**
     * Lance la vue
     */
    public void run() {
        ConsoleMenu connexion_menu = new ConsoleMenu("ESPACE ADHERENT", new Logout());

        connexion_menu.addChoice(new SeeEventsList(), "Voir la liste des événements");
        connexion menu.addChoice(new CreateEvent(), "Créer un événement");
        connexion menu.addChoice(new JoinEvent(), "Rejoindre un événement");

        connexion_menu.runMenu();
    }

    /**
     * Callback permettant d'afficher la liste des événements qui ne sont pas complets et la liste des
     * événements aux quels participe le membre courant
     */
    private class SeeEventsList implements Callable {
        @Override
        public void call() {
            ServiceResult<List<Event>, Request Error> available events rslt =
                                                Services.ListAvailableEvents();
            ServiceResult<List<Event>, Request_Error> member_events_rslt =
                                                Services.ListEventsOfMember(logged_member.getId());

            if(available_events_rslt.error == Request_Error.OK && member_events_rslt.error ==
                                                Request Error.OK) {
                System.out.println("### Evenements que vous avez rejoint ###");
                List<Event> member_events = member_events_rslt.result;
                for(int i = 0; i < member_events.size(); ++i)
                    System.out.println("\t- " + member_events.get(i));

                System.out.println("### Evenements disponnibles (pas encore complets) ###");
                List<Event> available_events = available_events_rslt.result;
                for(int i = 0; i < available_events.size(); ++i)
                    System.out.println("\t- " + available_events.get(i));
            }
            else
                // TODO: gèrer l'erreur en fonction de 'available_events_rslt.error' ou
                // 'member_events_rslt.error'
                System.out.println("Erreur lors de l'execution du service : " +
          (member_events_rslt.error == Request_Error.OK ? available_events_rslt.error :
member_events_rslt.error));
```

```
        }
    }

    /**
     * Callback permetant au membre de crer un événement
     */
    private class CreateEvent implements Callable {
        @Override
        public void call() {
            System.out.println("### Création d'événement ###");

            // Affiche la liste des activités disponnibles
            ServiceResult<List<Activity>, Request_Error> activities_list_rslt =
                                                                    Services.ListActivities();
            if(activities_list_rslt.error == Request_Error.OK)
            {
                List<Activity> activities = activities_list_rslt.result;
                for(int i = 0; i < activities.size(); ++i)
                    System.out.print((i == 0 ? "\tActivités disponnibles : " : ", \n\t\t") +
                                                                    activities.get(i));
            }
            else
                // TODO: gèrer l'erreur en fonction de 'activities_list_rslt.error'
                System.out.println("Erreur lors de l'execution du service : " +
                                   activities list rslt.error);

            // Demmande l'activité de l'événement et crée l'évenement
            long activity_id = ConsoleUtils.lireInteger("\nEntrez l'id de l'activité : ");
            Date today = new java.sql.Date((new java.util.Date()).getTime()); // La date de l'évenement
                                                                    // est la date actuelle
            ServiceResult<Event, Request_Error> event_creation_rslt =
                                   Services.CreateEvent(logged_member.getId(), activity_id, today);

            if(event_creation_rslt.error == Request_Error.OK)
                System.out.println("Evenement créé avec succès : " + event_creation_rslt.result);
            else
                // TODO: gèrer l'erreur en fonction de 'event_creation_rslt.error'
                System.out.println("Erreur lors de l'execution du service:"+ event_creation_rslt.error);
        }
    }

    /**
     * Callback permettant au membre de rejoindre un événement
     */
    private class JoinEvent implements Callable {
        @Override
        public void call() {
            long id = ConsoleUtils.lireInteger("Entrez l'id de l'évenement que vous voulez rejoindre:");

            Request_Error join_event_rslt = Services.JoinEvent(logged_member.getId(), id);

            if(join event rslt == Request Error.OK)
                System.out.println("Evenement rejoint avec succès.");
            else
                // TODO: gèrer l'erreur en fonction de 'join_event_rslt'
                System.out.println("Erreur lors de l'execution du service : " + join_event_rslt);
        }
    }

    /**
     * Callback appelée si l'utilisateur veux se déconnecter (retourner à la vue de connexion)
     */
    private class Logout implements Callable {
        @Override
        public void call() {

        }
    }

    private Member logged member;
}
```

ConsoleDashboardView (View)

```java
package TP1 SI.View;

import TP1 SI.Utils.Callable;
import TP1_SI.Utils.ConsoleMenu;
import TP1_SI.Utils.ConsoleUtils;
import TP1_SI.metier.model.Event;
import TP1 SI.metier.model.Location;
import TP1_SI.metier.service.ServiceResult;
import TP1 SI.metier.service.Services;

import java.util.List;

/**
 * Vue de l'administrateur assignat les lieux aux evenements
 * @author B3330
 */
public class ConsoleDashboardView {

    /**
     * Lance la vue
     */
    public void run() {
        ConsoleMenu connexion_menu = new ConsoleMenu("DASHBOARD", new Logout());
        connexion menu.addChoice(new AssignLocationToEvent(), "Assign location to event");

        connexion menu.runMenu();
    }

    /**
     * Callback gèrant l'assingation d'un lieu à un evenement
     */
    private static class AssignLocationToEvent implements Callable {
        @Override
        public void call() {
            ServiceResult<List<Event>, Services.Request_Error> events_rslt = Services.ListAllEvents();
            ServiceResult<List<Location>, Services.Request_Error> locations_rslt =
                                                                    Services.ListLocations();

            if(events_rslt.error == Services.Request_Error.OK && locations_rslt.error ==
                                                                    Services.Request_Error.OK ) {
                System.out.println("### Liste de tout les évenements ###");
                List<Event> events = events_rslt.result;
                for(int i = 0; i < events.size(); ++i)
                    System.out.println("\t- " + events.get(i));

                System.out.println("\n### Liste de tout les lieux ###");
                List<Location> locations = locations rslt.result;
                for(int i = 0; i < locations.size(); ++i)
                    System.out.println("\t- " + locations.get(i));

                long event_id = ConsoleUtils.lireInteger("\nEntrez l'id de l'évenement : ");
                long lieu_id = ConsoleUtils.lireInteger("Entrez l'id du lieu : ");

                Services.Request_Error error_code = Services.AssignLocationToEvent(lieu_id, event_id);
                if(error code == Services.Request Error.OK)
                    System.out.println("Assignation réussie.");
                else
                    System.out.println("Assignation échouée.");
            }
            else
                // TODO: gèrer l'erreur en fonction de 'events_rslt.error' ou 'locations_rslt.error'
                System.out.println("Erreur lors de l'execution du service : " + (events_rslt.error ==
                            Services.Request_Error.OK ? locations_rslt.error : events_rslt.error) );
        }
    }
```

```
    /**
     * Callback appelé si l'utilisateur veux retourner à la vue de connexion
     */
    private static class Logout implements Callable {
        @Override
        public void call() {

        }
    }
}
```

FIN