





AP

4- Développement de l'API Java de l'application Android

1- Rappel de l'architecture applicative globale et objectif de cette étape
2- Le diagramme des classes Java
3- Création d'un nouveau workspace et du projet
4- Exécution des tests unitaires des classes Point, PointDeTrace et Trace
5- Création des tests unitaires de la classe Utilisateur
6- Test des méthodes fournies de la classe PasserelleServicesWebXML
7- Développement et test des éléments manquants de la classe PasserelleServicesWebXML

1- Rappel de l'architecture applicative globale et objectif de cette étape

La page suivante présente l'architecture générale de l'application **TraceGPS**.

Cette étape porte sur le développement et le test de **l'API Java** qui sera utilisée par l'application Android pour accéder aux services web et permettre un accès à la base de données au moyen d'une connexion Internet.

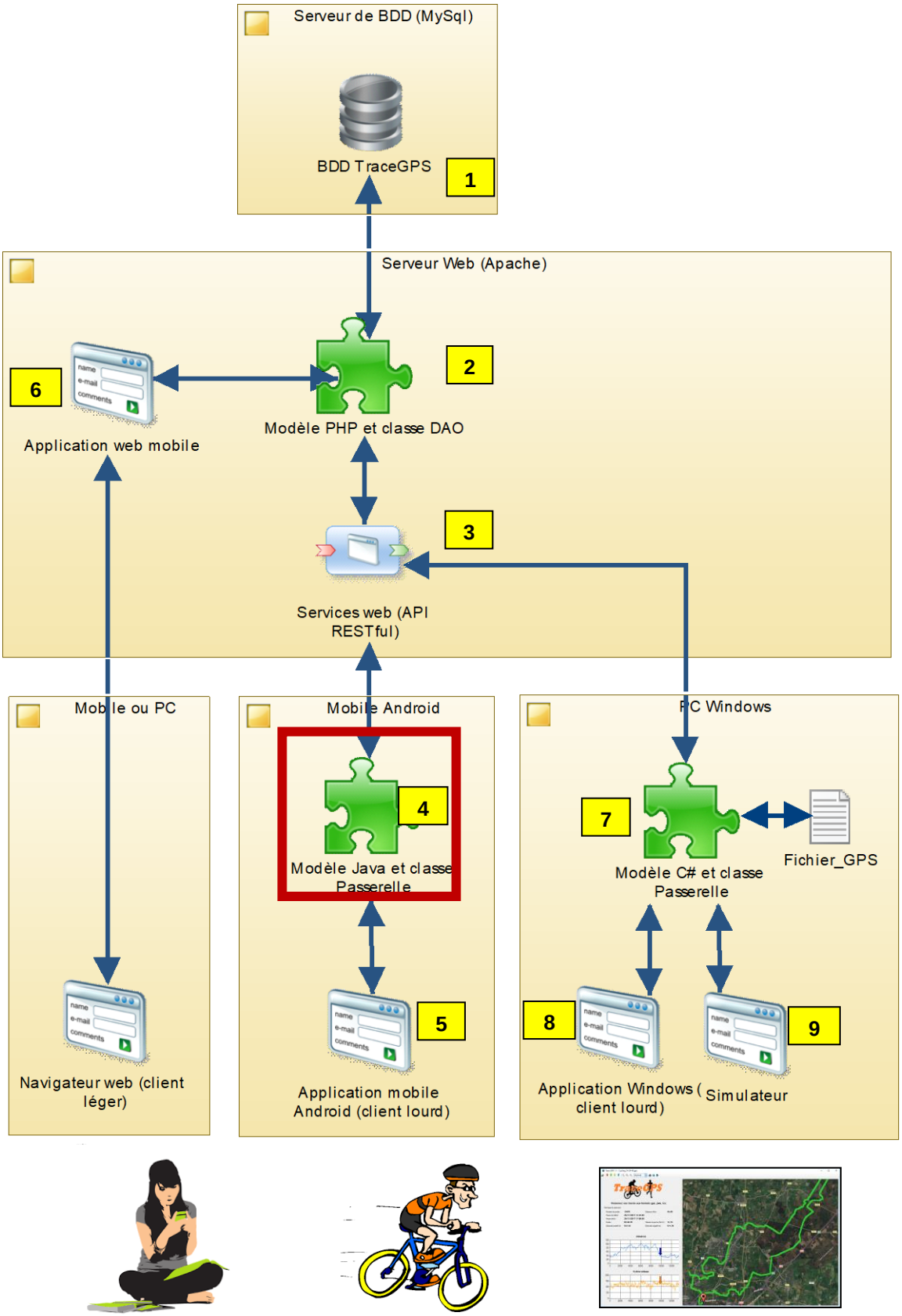
L'API Java est composée de 4 classes métiers et de 6 classes techniques.

Les 4 **classes métiers** :

- **Point** : représente un point géographique
- **PointDeTrace** : (hérite de **Point**) ; représente un point de passage lors d'un parcours
- **Trace** : représente une trace (ou un parcours) réalisé par un utilisateur
- **Utilisateur** : représente un utilisateur

Les 6 **classes techniques** :

- **PasserelleXML** : classe abstraite pour parser un document XML
- **PasserelleServiceWebXML** : hérite de PasserelleXML ; pour parser un document XML obtenu par l'appel d'un service web
- **PasserelleFichierXML** : hérite de **PasserelleXML** ; pour parser un document XML obtenu par lecture d'un fichier
- **PasserelleGPX** : hérite de **PasserelleFichierXML** ; pour parser un document XML obtenu par l'appel d'un fichier au format **GPX**
- **PasserellePWX** : hérite de **PasserelleFichierXML** ; pour parser un document XML obtenu par l'appel d'un fichier au format **PWX**
- **PasserelleTCX** : hérite de **PasserelleFichierXML** ; pour parser un document XML obtenu par l'appel d'un fichier au format **TCX**


Serveur de BDD (MySql)
BDD TraceGPS
1
Serveur Web (Apache)
2
Modèle PHP et classe DAO
6
Application web mobile
3
Services web (API RESTful)
Mobile ou PC
Navigateur web (client léger)
Mobile Android
4
Modèle Java et classe Passerelle
5
Application mobile Android (client lourd)
PC Windows
7
Modèle C# et classe Passerelle
Fichier_GPS
8
Application Windows (client lourd)
9
Simulateur

2- Le diagramme des classes Java

3- Création d'un nouveau workspace et du projet

3-1 Création d'un nouveau workspace

Créez un nouvel espace de travail nommé ***D:\ws-java-utf8-xxx*** (où xxx est votre nom).

Ce workspace sera encodé en UTF8 pour être compatible avec le workspace d'Android Studio.

Ouvrez Eclipse, et choisissez ce nouveau workspace avec la commande **File/Switch Workspace**.

3-2 Réglage des préférences

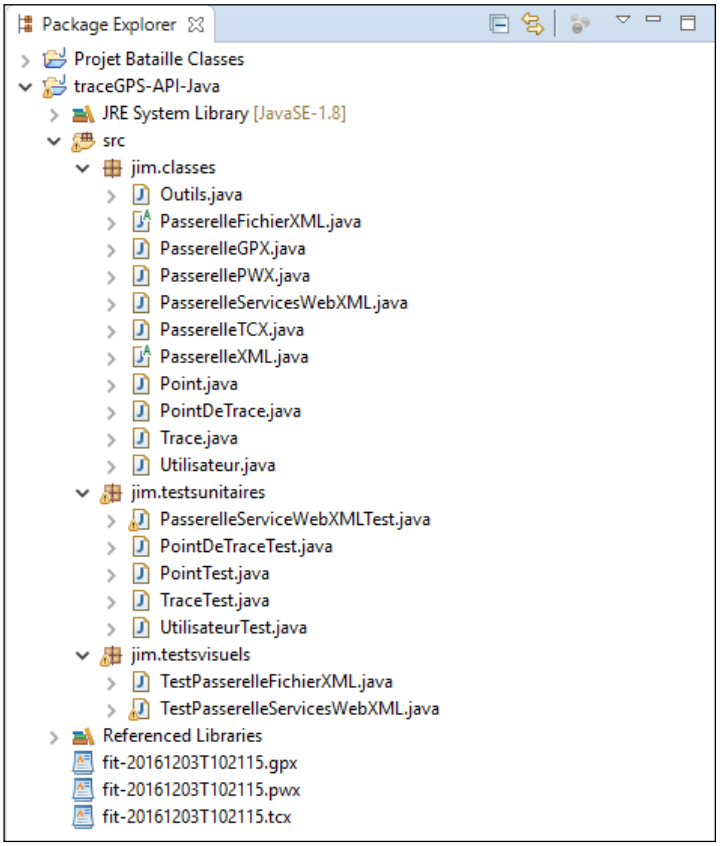
Avec la commande **Window/Preferences**, onglet **General/Workspace**, choisissez l'encodage **UTF8**.

Pour développer en Java, sélectionnez la **perspective Java**.

3-3 Création du projet Java

1. créer un nouveau projet avec la commande **File / New / Java Project** et préciser le nom du projet (**traceGPS-API-Java**).
2. dans le dossier **src**, créer un premier package avec la commande **File / New / Package** et lui donner un nom (**xxx.classes**) ; ce package contiendra les classes métiers et les classes techniques ; importez les fichiers fournis avec la commande **File / Import...**
3. dans le dossier **src**, créer un deuxième package avec la commande **File / New / Package** et lui donner un nom (**xxx.testsunitaires**) ; ce package contiendra les tests unitaires des classes métiers ; importez les fichiers fournis avec la commande **File / Import...**
4. dans le dossier **src**, créer un troisième package avec la commande **File / New / Package** et lui donner un nom (**xxx.testsvisuels**) ; ce package contiendra les tests visuels des classes passerelles ; importez les fichiers fournis avec la commande **File / Import...**
5. à la racine du projet, importez les 3 fichiers XML fournis avec la commande **File / Import...**

La structure du projet devrait devenir :



3-4 Préparation de l'environnement de tests

Pour pouvoir utiliser **JUnit**, il faut indiquer l'emplacement du fichier **junit.jar** dans le classpath du projet (il se trouve normalement dans le dossier **C:\photon\plugins\org.junit_4.12.0.v201504281640** de l'environnement Eclipse) ainsi que l'emplacement du fichier **hamcrest-all-1.3.jar** (il se trouve normalement dans le dossier **C:\photoon\plugins \Pour faire du junit 4**) :

- Sélectionnez le projet à tester et lancez la commande **File / Properties** pour accéder aux propriétés du projet.
- Allez au nœud **Java Build Path** puis à l'onglet **Libraries** et ajoutez les librairies **junit.jar** et **hamcrest-all-1.3.jar** (avec le bouton **Add External JARs...**)

4- Exécution des tests unitaires des classes Point, PointDeTrace et Trace

Exécutez les tests unitaires suivants pour vérifier le bon fonctionnement des classes métiers **Point**, **PointDeTrace** et **Trace** :

- **PointTest.java**
- **PointDeTraceTest.java**
- **TraceTest.java**

En cas d'échec d'un test, trouvez la cause du problème et relancez les tests jusqu'à leur réussite.

5- Création des tests unitaires de la classe Utilisateur

Les tests unitaires de la classe **Utilisateur** ont été commencés dans le fichier **UtilisateurTest.java**.

La fonction **setUp** est appelée avant chaque test ; elle crée 2 objets de la classe **Utilisateur** :

- **utilisateur1** : avec les données par défaut
- **utilisateur2** : avec des données indiquées au constructeur

Le test de la méthode **toString** est déjà écrit. Vous êtes chargés d'écrire les tests unitaires manquants en utilisant les 2 objets **utilisateur1** et **utilisateur2** créés par la fonction **setUp**. Vous pouvez bien sûr vous inspirer des tests fournis pour les autres classes.

Quand ces tests seront écrits, exécutez-les.
En cas d'échec, trouvez la cause du problème et relancez les tests jusqu'à leur réussite.

6- Test des méthodes fournies de la classe PasserelleServicesWebXML

Les 5 méthodes fournies concernent **la gestion des utilisateurs** et utilisent les services web qui vous étaient fournis à l'étape du développement des services web :

- connecter
- getTousLesUtilisateurs
- creerUnUtilisateur
- supprimerUnUtilisateur
- changerDeMdp

Remarque générale concernant toutes les méthodes écrites ou à écrire :

Les méthodes retournent toutes une chaine contenant le message de réponse du service web (ou un message d'erreur lorsqu'une exception se produit lors de l'exécution).

Remarques générales concernant les tests :

Les méthodes dont le nom commence par **get** fournissent des listes de données ; elles seront testées par des tests visuels regroupés dans le fichier **TestPasserelleServiceWebXML.java** du package **jim.testsunitaires**.

Les autres méthodes seront testées par des tests unitaires regroupés dans le fichier **PasserelleServiceWebXMLTest.java** du package **jim.testsvisuels.**
Ces tests doivent bien sûr être adaptés en fonction du contenu de votre base de données.

Quand un test est réussi, désactiver son code en le mettant en commentaire.

6-1 la méthode connecter

public static String connecter(String pseudo, String mdpSha1)
Rôle : permet à l'utilisateur de se connecter pour accéder au menu de l'application (appelle le service **Connecter**).
Paramètres à fournir : **pseudo** : le pseudo de l'utilisateur qui fait appel au service web **mdpSha1** : son mot de passe hashé en SHA1
Valeur de retour : une chaine contenant le message de réponse du service web (ou un message d'erreur lorsqu'une exception se produit lors de l'exécution).

Ajoutez le code du **test unitaire** suivant dans le fichier **PasserelleServiceWebXMLTest.java** :

```
@Test
public void testConnecter() {
    String msg = PasserelleServicesWebXML.connecter("admin", "adminnnnnnnn");
    assertEquals("Erreur : authentification incorrecte.", msg);

    msg = PasserelleServicesWebXML.connecter("admin", Outils.sha1("mdpadmin"));
    assertEquals("Administrateur authentifié.", msg);

    msg = PasserelleServicesWebXML.connecter("europa", Outils.sha1("mdputilisateur"));
    assertEquals("Utilisateur authentifié.", msg);
}
```

Exécutez le test.

En cas d'échec, trouvez la cause du problème et relancez le test jusqu'à sa réussite.

6-2 la méthode getTousLesUtilisateurs

public static String getTousLesUtilisateurs(String pseudo, String mdpSha1, ArrayList<Utilisateur> lesUtilisateurs)
Rôle : permet d'obtenir la liste de tous les utilisateurs de niveau 1 (appelle le service **GetTousLesUtilisateurs**).
Paramètres à fournir : **pseudo** : le pseudo de l'utilisateur qui fait appel au service web **mdpSha1** : son mot de passe hashé en SHA1 **lesUtilisateurs** : collection (vide) à remplir à partir des données fournies par le service web
Valeur de retour : une chaine contenant le message de réponse du service web (ou un message d'erreur lorsqu'une exception se produit lors de l'exécution).

Ajoutez le code du **test visuel** suivant dans le fichier **TestPasserelleServiceWebXML.java** :

```
String msg;

// test visuel de la méthode getTousLesUtilisateurs
ArrayList<Utilisateur> lesUtilisateurs = new ArrayList<Utilisateur>();
msg = PasserelleServicesWebXML.getTousLesUtilisateurs("europa", Outils.sha1("mdputilisateur"), lesUtilisateurs);
// affichage de la réponse
System.out.println(msg);
// affichage du nombre d'utilisateurs
System.out.println("Nombre d'utilisateurs : " + lesUtilisateurs.size());
// affichage de tous les utilisateurs
for (Utilisateur unUtilisateur : lesUtilisateurs)
{    System.out.println(unUtilisateur.toString());
}
```

Exécutez le test.

Comparez l'affichage obtenu avec le contenu de votre base de données et en cas d'échec, trouvez la cause du problème et relancez le test jusqu'à sa réussite.

6-3 la méthode creerUnUtilisateur

public static String creerUnUtilisateur(String pseudo, String adrMail, String numTel)
Rôle : permet de créer un compte utilisateur (appelle le service **creerUnUtilisateur**).
Paramètres à fournir : **pseudo** : le pseudo de l'utilisateur qui fait appel au service web **adrMail** : son adresse mail **numTel** : son numéro de téléphone
Valeur de retour : une chaine contenant le message de réponse du service web (ou un message d'erreur lorsqu'une exception se produit lors de l'exécution).

Ajoutez le code du **test unitaire** suivant dans le fichier **PasserelleServiceWebXMLTest.java** :

```
    @Test
    public void testCreerUnUtilisateur() {
        String msg = PasserelleServicesWebXML.creerUnUtilisateur("jim", "delasalle.sio.eleves@gmail.com", "1122334455");
        assertEquals("Erreur : pseudo trop court (8 car minimum) ou déjà existant.", msg);

        msg = PasserelleServicesWebXML.creerUnUtilisateur("turlututu", "delasalle.sio.elevesgmail.com", "1122334455");
        assertEquals("Erreur : adresse mail incorrecte ou déjà existante.", msg);

        msg = PasserelleServicesWebXML.creerUnUtilisateur("turlututu", "delasalle.sio.eleves@gmailcom", "1122334455");
        assertEquals("Erreur : adresse mail incorrecte ou déjà existante.", msg);

        msg = PasserelleServicesWebXML.creerUnUtilisateur("turlututu", "delasalle.sio.eleves@gmail.com", "1122334455");
        assertEquals("Erreur : adresse mail incorrecte ou déjà existante.", msg);

        msg = PasserelleServicesWebXML.creerUnUtilisateur("turlututu", "delasallesioeleves@gmail.com", "1122334455");
        assertEquals("Enregistrement effectué ; vous allez recevoir un courriel avec votre mot de passe.", msg);

        msg = PasserelleServicesWebXML.creerUnUtilisateur("turlututu", "de.la.salle.sio.eleves@gmail.com", "1122334455");
        assertEquals("Erreur : pseudo trop court (8 car minimum) ou déjà existant.", msg);
    }
```

Exécutez le test.

En cas d'échec, trouvez la cause du problème et relancez le test jusqu'à sa réussite.

6-4 la méthode supprimerUnUtilisateur

public static String supprimerUnUtilisateur(String pseudo, String mdpSha1, String pseudoAsupprimer)
Rôle : permet à un administrateur de supprimer un utilisateur (appelle le service **SupprimerUnUtilisateur**).
Paramètres à fournir : **pseudo** : le pseudo de l'administrateur qui fait appel au service web **mdpSha1** : le mot de passe hashé en sha1 **pseudoAsupprimer** : le pseudo de l'utilisateur à supprimer
Valeur de retour : une chaine contenant le message de réponse du service web (ou un message d'erreur lorsqu'une exception se produit lors de l'exécution).

Ajoutez le code du **test unitaire** suivant dans le fichier **PasserelleServiceWebXMLTest.java** :

```
    @Test
    public void testSupprimerUnUtilisateur() {
        String msg;
        msg = PasserelleServicesWebXML.supprimerUnUtilisateur("europa", Outils.sha1("mdputilisateurrrrrr"), "toto");
        assertEquals("Erreur : authentification incorrecte.", msg);

        msg = PasserelleServicesWebXML.supprimerUnUtilisateur("europa", Outils.sha1("mdputilisateur"), "toto");
        assertEquals("Erreur : authentification incorrecte.", msg);

        msg = PasserelleServicesWebXML.supprimerUnUtilisateur("admin", Outils.sha1("mdpadminnnnn"), "toto");
        assertEquals("Erreur : authentification incorrecte.", msg);

        msg = PasserelleServicesWebXML.supprimerUnUtilisateur("admin", Outils.sha1("mdpadmin"), "toto");
        assertEquals("Erreur : pseudo utilisateur inexistant.", msg);

        msg = PasserelleServicesWebXML.supprimerUnUtilisateur("admin", Outils.sha1("mdpadmin"), "neon");
        assertEquals("Erreur : suppression impossible ; cet utilisateur possède encore des traces.", msg);

        msg = PasserelleServicesWebXML.supprimerUnUtilisateur("admin", Outils.sha1("mdpadmin"), "turlututu");
        assertEquals("Suppression effectuée ; un courriel va être envoyé à l'utilisateur.", msg);
    }
```

Exécutez le test.

En cas d'échec, trouvez la cause du problème et relancez le test jusqu'à sa réussite.

6-5 la méthode changerDeMdp

public static String changerDeMdp(String pseudo, String mdpSha1, String nouveauMdp, String confirmationMdp)
Rôle : permet de modifier son mot de passe (appelle le service **ChangerDeMdp**).
Paramètres à fournir : **pseudo** : le pseudo de l'utilisateur qui fait appel au service web **mdpSha1** : le mot de passe hashé en sha1 **nouveauMdp** : le nouveau mot de passe **confirmationMdp** : la confirmation du nouveau mot de passe
Valeur de retour : une chaine contenant le message de réponse du service web (ou un message d'erreur lorsqu'une exception se produit lors de l'exécution).

Ajoutez le code du **test unitaire** suivant dans le fichier **PasserelleServiceWebXMLTest.java** :

```
    @Test
    public void testChangerDeMdp() {
        String msg = PasserelleServicesWebXML.changerDeMdp("europa", Outils.sha1("mdputilisateur"),
"passepasse", "passepassepasse");
        assertEquals("Erreur : le nouveau mot de passe et sa confirmation sont différents.", msg);

        msg = PasserelleServicesWebXML.changerDeMdp("europa", Outils.sha1("mdputilisateurrrr"), "passepasse",
"passepasse");
        assertEquals("Erreur : authentification incorrecte.", msg);

        msg = PasserelleServicesWebXML.changerDeMdp("europa", Outils.sha1("mdputilisateur"),
"mdputilisateurrrr", "mdputilisateurrrr");
        assertEquals("Enregistrement effectué ; vous allez recevoir un courriel de confirmation.", msg);

        msg = PasserelleServicesWebXML.changerDeMdp("europa", Outils.sha1("mdputilisateurrrr"),
"mdputilisateur", "mdputilisateur");
        assertEquals("Enregistrement effectué ; vous allez recevoir un courriel de confirmation.", msg);
    }
```

Exécutez le test.

En cas d'échec, trouvez la cause du problème et relancez le test jusqu'à sa réussite.

7- Développement et test des éléments manquants de la classe PasserelleServicesWebXML

Les 11 méthodes restant à développer utilisent les services web que vous avez développés à l'étape du développement des services web :

Services web du domaine **Gestion des utilisateurs** :

- demanderMdp

Services web du domaine **Gestion des autorisations** :

- getLesUtilisateursQueJautorise
- getLesUtilisateursQuiMautorisent
- demanderUneAutorisation
- retirerUneAutorisation

Services web du domaine **Gestion des points de traces** :

- envoyerPosition

Services web du domaine **Gestion des traces** :

- getUnParcoursEtSesPoints
- getLesParcoursDunUtilisateur
- supprimerUnParcours
- demarrerEnregistrementParcours
- arreterEnregistrementParcours

7-1 la méthode demanderMdp

public static String demanderMdp(String pseudo)
Rôle : permet de demander un nouveau mot de passe (appelle le service **DemanderMdp**).
Paramètres à fournir : **pseudo** : le pseudo de l'utilisateur qui fait appel au service web
Valeur de retour : une chaine contenant le message de réponse du service web (ou un message d'erreur lorsqu'une exception se produit lors de l'exécution).

Ajoutez le code du **test unitaire** suivant dans le fichier **PasserelleServiceWebXMLTest.java** :

```
@Test
public void testDemanderMdp() {
    String msg = PasserelleServicesWebXML.demanderMdp("jim");
    assertEquals("Erreur : pseudo inexistant.", msg);

    msg = PasserelleServicesWebXML.demanderMdp("europa");
    assertEquals("Vous allez recevoir un courriel avec votre nouveau mot de passe.", msg);
}
```

Exécutez le test.

En cas d'échec, trouvez la cause du problème et relancez le test jusqu'à sa réussite.

7-2 la méthode getLesUtilisateursQueJautorise

public static String getLesUtilisateursQueJautorise(String pseudo, String mdpSha1, ArrayList<Utilisateur> lesUtilisateurs)
Rôle : permet d'obtenir la liste des utilisateurs que j'autorise (appelle le service **GetLesUtilisateursQueJautorise**).
Paramètres à fournir : **pseudo** : le pseudo de l'utilisateur qui fait appel au service web **mdpSha1** : son mot de passe hashé en SHA1 **lesUtilisateurs** : collection (vide) à remplir à partir des données fournies par le service web
Valeur de retour : une chaine contenant le message de réponse du service web (ou un message d'erreur lorsqu'une exception se produit lors de l'exécution).

Ajoutez le code du **test visuel** suivant dans le fichier **TestPasserelleServiceWebXML.java** :

```
String msg;

// test visuel de la méthode getLesUtilisateursQueJautorise
ArrayList<Utilisateur> lesUtilisateurs = new ArrayList<Utilisateur>();
msg = PasserelleServicesWebXML.getLesUtilisateursQueJautorise("europa", Outils.sha1("mdputilisateur"), lesUtilisateurs);
// affichage de la réponse
System.out.println(msg);
// affichage du nombre d'utilisateurs
System.out.println("Nombre d'utilisateurs : " + lesUtilisateurs.size());
// affichage de tous les utilisateurs
for (Utilisateur unUtilisateur : lesUtilisateurs)
{    System.out.println(unUtilisateur.toString());
}
```

Exécutez le test.

Comparez l'affichage obtenu avec le contenu de votre base de données et en cas d'échec, trouvez la cause du problème et relancez le test jusqu'à sa réussite.

7-3 la méthode getLesUtilisateursQuiMautorisent

public static String getLesUtilisateursQuiMautorisent(String pseudo, String mdpSha1, ArrayList<Utilisateur> lesUtilisateurs)
Rôle : permet d'obtenir la liste des utilisateurs qui m'autorisent (appelle le service **GetLesUtilisateursQuiMautorisent**).
Paramètres à fournir : **pseudo** : le pseudo de l'utilisateur qui fait appel au service web **mdpSha1** : son mot de passe hashé en SHA1 **lesUtilisateurs** : collection (vide) à remplir à partir des données fournies par le service web
Valeur de retour : une chaine contenant le message de réponse du service web (ou un message d'erreur lorsqu'une exception se produit lors de l'exécution).

Ajoutez le code du **test visuel** suivant dans le fichier **TestPasserelleServiceWebXML.java** :

```
String msg;

// test visuel de la méthode getLesUtilisateursQuiMautorisent
ArrayList<Utilisateur> lesUtilisateurs = new ArrayList<Utilisateur>();
msg = PasserelleServicesWebXML.getLesUtilisateursQuiMautorisent("europa", Outils.sha1("mdputilisateur"), lesUtilisateurs);
// affichage de la réponse
System.out.println(msg);
// affichage du nombre d'utilisateurs
System.out.println("Nombre d'utilisateurs : " + lesUtilisateurs.size());
// affichage de tous les utilisateurs
for (Utilisateur unUtilisateur : lesUtilisateurs)
{    System.out.println(unUtilisateur.toString());
}
```

Exécutez le test.

Comparez l'affichage obtenu avec le contenu de votre base de données et en cas d'échec, trouvez la cause du problème et relancez le test jusqu'à sa réussite.

7-4 la méthode demanderUneAutorisation

public static String demanderUneAutorisation(String pseudo, String mdpSha1, String pseudoDestinataire, String texteMessage, String nomPrenom)
Rôle : permet de demander une autorisation (appelle le service **DemanderUneAutorisation**).
Paramètres à fournir : **pseudo** : le pseudo de l'utilisateur qui fait appel au service web **mdpSha1** : le mot de passe hashé en sha1 **pseudoDestinataire** : le pseudo de l'utilisateur à qui on demande l'autorisation **texteMessage** : le texte d'un message accompagnant la demande **nomPrenom** : le nom et le prénom du demandeur
Valeur de retour : une chaine contenant le message de réponse du service web (ou un message d'erreur lorsqu'une exception se produit lors de l'exécution).

Ajoutez le code du **test unitaire** suivant dans le fichier **PasserelleServiceWebXMLTest.java** :

```
    @Test
    public void testDemanderUneAutorisation() {
        String msg = PasserelleServicesWebXML.demanderUneAutorisation("europa",
Outils.sha1("mdputilisateurrrrrr"), "toto", "", "");
        assertEquals("Erreur : données incomplètes.", msg);

        msg = PasserelleServicesWebXML.demanderUneAutorisation("europa", Outils.sha1("mdputilisateurrrrrr"),
"toto", "coucou", "charles-edouard");
        assertEquals("Erreur : authentification incorrecte.", msg);

        msg = PasserelleServicesWebXML.demanderUneAutorisation("europa", Outils.sha1("mdputilisateur"), "toto",
"coucou", "charles-edouard");
        assertEquals("Erreur : pseudo utilisateur inexistant.", msg);

        msg = PasserelleServicesWebXML.demanderUneAutorisation("europa", Outils.sha1("mdputilisateur"),
"galileo", "coucou", "charles-edouard");
        assertEquals("galileo va recevoir un courriel avec votre demande.", msg);
    }
```

Exécutez le test.

En cas d'échec, trouvez la cause du problème et relancez le test jusqu'à sa réussite.

7-5 la méthode retirerUneAutorisation

public static String retirerUneAutorisation(String pseudo, String mdpSha1, String pseudoARetirer, String texteMessage)
Rôle : permet de retirer une autorisation (appelle le service **RetirerUneAutorisation**).
Paramètres à fournir : **pseudo** : le pseudo de l'utilisateur qui fait appel au service web **mdpSha1** : le mot de passe hashé en sha1 **pseudoARetirer** : le pseudo de l'utilisateur à qui on veut retirer l'autorisation **texteMessage** : le texte d'un message pour un éventuel envoi de courriel
Valeur de retour : une chaine contenant le message de réponse du service web (ou un message d'erreur lorsqu'une exception se produit lors de l'exécution).

Ajoutez le code du **test unitaire** suivant dans le fichier **PasserelleServiceWebXMLTest.java** :

```
    @Test
    public void testRetirerUneAutorisation() {
        String msg = PasserelleServicesWebXML.retirerUneAutorisation("europa", Outils.sha1("mdputilisateurrrrrr"),
"toto", "coucou");
        assertEquals("Erreur : authentification incorrecte.", msg);

        msg = PasserelleServicesWebXML.retirerUneAutorisation("europa", Outils.sha1("mdputilisateur"), "toto",
"coucou");
        assertEquals("Erreur : pseudo utilisateur inexistant.", msg);

        msg = PasserelleServicesWebXML.retirerUneAutorisation("europa", Outils.sha1("mdputilisateur"), "juno",
"coucou");
        assertEquals("Erreur : l'autorisation n'était pas accordée.", msg);

        msg = PasserelleServicesWebXML.retirerUneAutorisation("neon", Outils.sha1("mdputilisateur"), "oxygen",
"coucou");
        assertEquals("Autorisation supprimée ; oxygen va recevoir un courriel de notification.", msg);

        msg = PasserelleServicesWebXML.retirerUneAutorisation("neon", Outils.sha1("mdputilisateur"), "photon", "");
        assertEquals("Autorisation supprimée.", msg);
    }
```

Exécutez le test.

En cas d'échec, trouvez la cause du problème et relancez le test jusqu'à sa réussite.

7-6 la méthode envoyerPosition

public static String envoyerPosition(String pseudo, String mdpSha1, PointDeTrace lePoint)
Rôle : permet d'envoyer la position de l'utilisateur (appelle le service **envoyerPosition**).
Paramètres à fournir : **pseudo** : le pseudo de l'utilisateur qui fait appel au service web **mdpSha1** : le mot de passe hashé en sha1 **lePoint** : un objet PointDeTrace (vide) qui permettra de récupérer le numéro attribué à partir des données fournies par le service web
Valeur de retour : une chaine contenant le message de réponse du service web (ou un message d'erreur lorsqu'une exception se produit lors de l'exécution).

Ajoutez le code du **test unitaire** suivant dans le fichier **PasserelleServiceWebXMLTest.java** :

```
    @Test
    public void testEnvoyerPosition() throws ParseException {
        Date laDate = Outils.convertirEnDateHeure("24/01/2018 13:42:21");

        PointDeTrace lePoint = new PointDeTrace(23, 0, 48.15, -1.68, 50, laDate, 80);
        String msg = PasserelleServicesWebXML.envoyerPosition("europa", Outils.sha1("mdputilisateurrrrrr"), lePoint);
        assertEquals("Erreur : authentification incorrecte.", msg);

        lePoint = new PointDeTrace(2333, 0, 48.15, -1.68, 50, laDate, 80);
        msg = PasserelleServicesWebXML.envoyerPosition("europa", Outils.sha1("mdputilisateur"), lePoint);
        assertEquals("Erreur : le numéro de trace n'existe pas.", msg);

        lePoint = new PointDeTrace(22, 0, 48.15, -1.68, 50, laDate, 80);
        msg = PasserelleServicesWebXML.envoyerPosition("europa", Outils.sha1("mdputilisateur"), lePoint);
        assertEquals("Erreur : le numéro de trace ne correspond pas à cet utilisateur.", msg);

        lePoint = new PointDeTrace(4, 0, 48.15, -1.68, 50, laDate, 80);
        msg = PasserelleServicesWebXML.envoyerPosition("europa", Outils.sha1("mdputilisateur"), lePoint);
        assertEquals("Point créé.", msg);
    }
```

Exécutez le test.

En cas d'échec, trouvez la cause du problème et relancez le test jusqu'à sa réussite.

7-7 la méthode getUnParcoursEtSesPoints

public static String getUnParcoursEtSesPoints(String pseudo, String mdpSha1, int idTrace, Trace laTrace)
Rôle : permet d'obtenir un parcours et la liste de ses points (appelle le service **GetUnParcoursEtSesPoints**).
Paramètres à fournir : **pseudo** : le pseudo de l'utilisateur qui fait appel au service web **mdpSha1** : son mot de passe hashé en SHA1 **idTrace** : l'id de la trace à consulter **laTrace** : objet Trace (vide) à remplir à partir des données fournies par le service web
Valeur de retour : une chaine contenant le message de réponse du service web (ou un message d'erreur lorsqu'une exception se produit lors de l'exécution).

Ajoutez le code du **test visuel** suivant dans le fichier **TestPasserelleServiceWebXML.java** :

```
        String msg;

        // test visuel de la méthode getUnParcoursEtSesPoints
        Trace laTrace = new Trace();
        msg = PasserelleServicesWebXML.getUnParcoursEtSesPoints("europa", Outils.sha1("mdputilisateur"), 2, laTrace);
        // affichage de la réponse
        System.out.println(msg);
        // affichage de la trace
        System.out.println(laTrace.toString());
```

Exécutez le test.

Comparez l'affichage obtenu avec le contenu de votre base de données et en cas d'échec, trouvez la cause du problème et relancez le test jusqu'à sa réussite.

7-8 la méthode getLesParcoursDunUtilisateur

public static String getLesParcoursDunUtilisateur(String pseudo, String mdpSha1, String pseudoConsulte, ArrayList<Trace> lesTraces)
Rôle : permet d'obtenir la liste des parcours d'un utilisateur (appelle le service **GetLesParcoursDunUtilisateur**).
Paramètres à fournir : **pseudo** : le pseudo de l'utilisateur qui fait appel au service web **mdpSha1** : son mot de passe hashé en SHA1 **idUtilisateur** : l'id de l'utilisateur dont on veut la liste des parcours **lesTraces** : collection (vide) à remplir à partir des données fournies par le service web
Valeur de retour : une chaine contenant le message de réponse du service web (ou un message d'erreur lorsqu'une exception se produit lors de l'exécution).

Ajoutez le code du **test visuel** suivant dans le fichier **TestPasserelleServiceWebXML.java** :

```
        String msg;

        // test visuel de la méthode getLesParcoursDunUtilisateur
        ArrayList<Trace> lesTraces = new ArrayList<Trace>();
        msg = PasserelleServicesWebXML.getLesParcoursDunUtilisateur("europa", Outils.sha1("mdputilisateur"),
"callisto", lesTraces);
        // affichage de la réponse
        System.out.println(msg);
        // affichage du nombre de traces
        System.out.println("Nombre de traces : " + lesTraces.size());
        // affichage de toutes les traces
        for (Trace uneTrace : lesTraces)
        {   System.out.println(uneTrace.toString());
        }
```

Exécutez le test.

Comparez l'affichage obtenu avec le contenu de votre base de données et en cas d'échec, trouvez la cause du problème et relancez le test jusqu'à sa réussite.

7-9 la méthode supprimerUnParcours

public static String supprimerUnParcours(String pseudo, String mdpSha1, int idTrace)
Rôle : permet de supprimer un parcours (appelle le service **SupprimerUnParcours**).
Paramètres à fournir : **pseudo** : le pseudo de l'utilisateur qui fait appel au service web **mdpSha1** : le mot de passe hashé en sha1 **idTrace** : l'id de la trace à supprimer
Valeur de retour : une chaine contenant le message de réponse du service web (ou un message d'erreur lorsqu'une exception se produit lors de l'exécution).

Ajoutez le code du **test unitaire** suivant dans le fichier **PasserelleServiceWebXMLTest.java** :

```
    @Test
    public void testSupprimerUnUnParcours() {
        String msg = PasserelleServicesWebXML.supprimerUnParcours("europa", Outils.sha1("mdputilisateurrrrrr"), 10);
        assertEquals("Erreur : authentification incorrecte.", msg);

        msg = PasserelleServicesWebXML.supprimerUnParcours("europa", Outils.sha1("mdputilisateur"), 100);
        assertEquals("Erreur : parcours inexistant.", msg);

        msg = PasserelleServicesWebXML.supprimerUnParcours("europa", Outils.sha1("mdputilisateur"), 22);
        assertEquals("Erreur : vous n'êtes pas le propriétaire de ce parcours.", msg);

        msg = PasserelleServicesWebXML.supprimerUnParcours("europa", Outils.sha1("mdputilisateur"), 30);
        assertEquals("Parcours supprimé.", msg);
    }
```

Exécutez le test.

En cas d'échec, trouvez la cause du problème et relancez le test jusqu'à sa réussite.

7-10 la méthode demarrerEnregistrementParcours

public static String demarrerEnregistrementParcours(String pseudo, String mdpSha1, Trace laTrace)
Rôle : permet de démarrer l'enregistrement d'un parcours (appelle le service **DemarrerEnregistrementParcours**).
Paramètres à fournir : **pseudo** : le pseudo de l'utilisateur qui fait appel au service web **mdpSha1** : le mot de passe hashé en sha1 **laTrace** : un objet Trace (vide) à remplir à partir des données fournies par le service web
Valeur de retour : une chaine contenant le message de réponse du service web (ou un message d'erreur lorsqu'une exception se produit lors de l'exécution).

Ajoutez le code du **test unitaire** suivant dans le fichier **PasserelleServiceWebXMLTest.java** :

```
    @Test
    public void testDemarrerEnregistrementParcours() {
        Trace laTrace = new Trace();
        String msg = PasserelleServicesWebXML.demarrerEnregistrementParcours("europa",
Outils.sha1("mdputilisateurrrrrr"), laTrace);
        assertEquals("Erreur : authentification incorrecte.", msg);

        laTrace = new Trace();
        msg = PasserelleServicesWebXML.demarrerEnregistrementParcours("europa",
Outils.sha1("mdputilisateur"), laTrace);
        assertEquals("Trace créée.", msg);
    }
```

Exécutez le test.

En cas d'échec, trouvez la cause du problème et relancez le test jusqu'à sa réussite.

7-11 la méthode arreterEnregistrementParcours

public static String arreterEnregistrementParcours(String pseudo, String mdpSha1, int idTrace)
Rôle : permet de terminer l'enregistrement d'un parcours (appelle le service **ArreterEnregistrementParcours**).
Paramètres à fournir : **pseudo** : le pseudo de l'utilisateur qui fait appel au service web **mdpSha1** : le mot de passe hashé en sha1 **idTrace** : l'id de la trace à terminer
Valeur de retour : une chaine contenant le message de réponse du service web (ou un message d'erreur lorsqu'une exception se produit lors de l'exécution).

Ajoutez le code du **test unitaire** suivant dans le fichier **PasserelleServiceWebXMLTest.java** :

```
    @Test
    public void testArreterEnregistrementParcours() {
        String msg;

        msg = PasserelleServicesWebXML.arreterEnregistrementParcours("europa", Outils.sha1("mdputilisateurrrrrr"), 23);
        assertEquals("Erreur : authentification incorrecte.", msg);

        msg = PasserelleServicesWebXML.arreterEnregistrementParcours("europa", Outils.sha1("mdputilisateur"), 230);
        assertEquals("Erreur : parcours inexistant.", msg);

        msg = PasserelleServicesWebXML.arreterEnregistrementParcours("europa", Outils.sha1("mdputilisateur"), 5);
        assertEquals("Erreur : le numéro de trace ne correspond pas à cet utilisateur.", msg);

        msg = PasserelleServicesWebXML.arreterEnregistrementParcours("europa", Outils.sha1("mdputilisateur"), 4);
        assertEquals("Erreur : cette trace est déjà terminée.", msg);

        msg = PasserelleServicesWebXML.arreterEnregistrementParcours("europa", Outils.sha1("mdputilisateur"), 23);
        assertEquals("Enregistrement terminé.", msg);
    }
```

Exécutez le test.

En cas d'échec, trouvez la cause du problème et relancez le test jusqu'à sa réussite.