2-6 Développement et test de la classe DAO

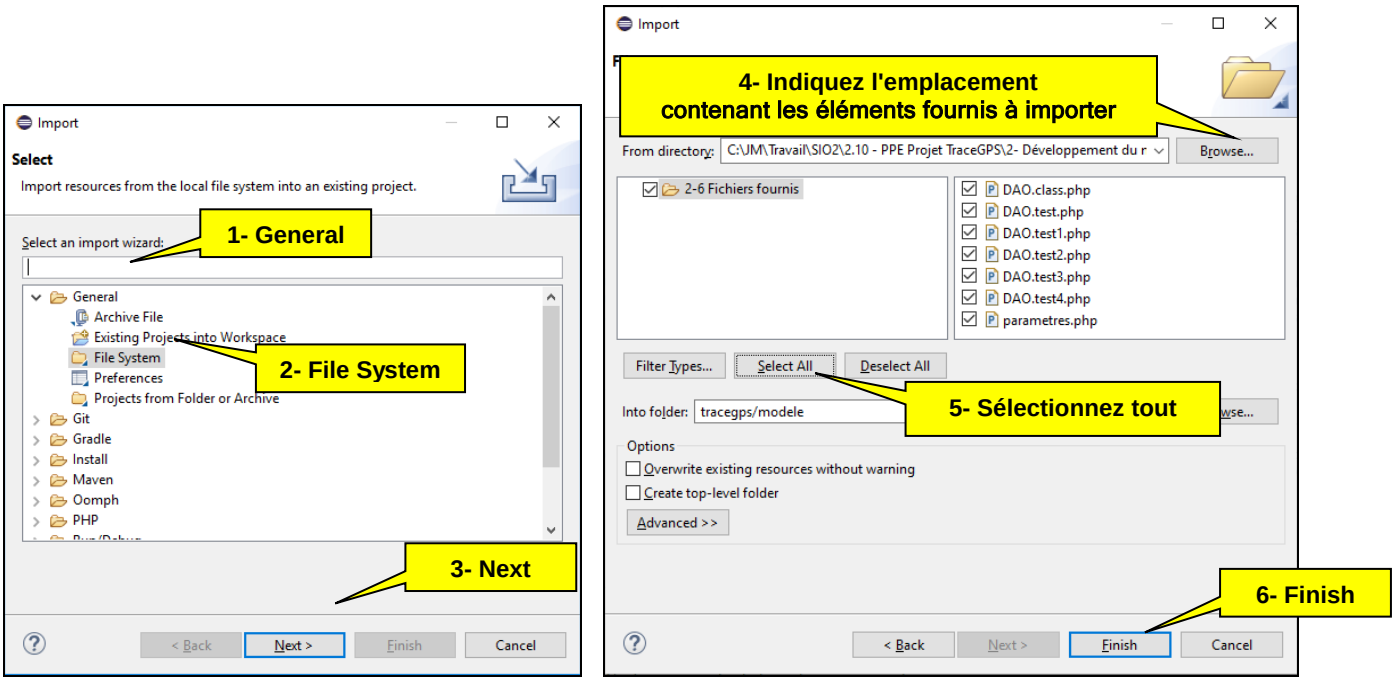
1- Importation des fichiers fournis
2- Rappel du diagramme des cas d'utilisation
3- Diagrammes UML des classes DAO et Outils
4- Liens entre cas d'utilisation et méthodes
5- Présentation du code déjà réalisé et des tests
6- Code restant à développer et organisation de la répartition du travail

1- Importation des fichiers fournis

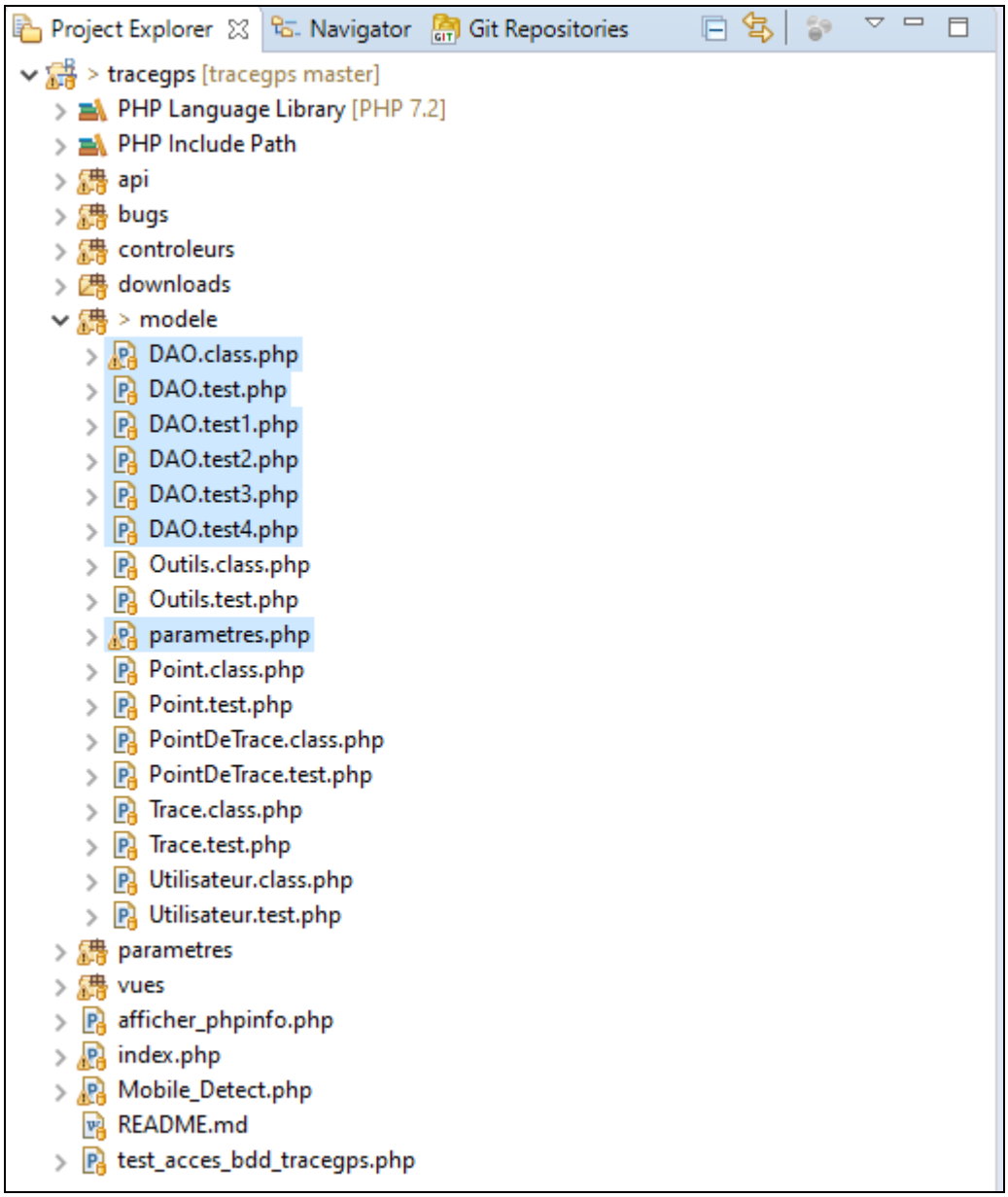
Nous allons importer dans le dossier **modele** les fichiers fournis du dossier "**2-6 Fichiers fournis**" :

- **DAO.class.php** : la classe **DAO** (dont le développement est juste commencé)
- **DAO.test.php** : la page de test de la classe **DAO** (les tests des méthodes fournies)
- **DAO.test1.php** : la page de test de la classe **DAO** (à compléter par le développeur 1)
- **DAO.test2.php** : la page de test de la classe **DAO** (à compléter par le développeur 2)
- **DAO.test3.php** : la page de test de la classe **DAO** (à compléter par le développeur 3)
- **DAO.test4.php** : la page de test de la classe **DAO** (à compléter par le développeur 4)
- **parametres.php** : le fichier contenant les paramètres de connexion à la bdd

Faire un clic droit **sur le dossier modele** (dans le **Project Explorer**) et choisir la commande **Import...** :



Constater dans le **Project Explorer** que les fichiers fournis sont bien importés :

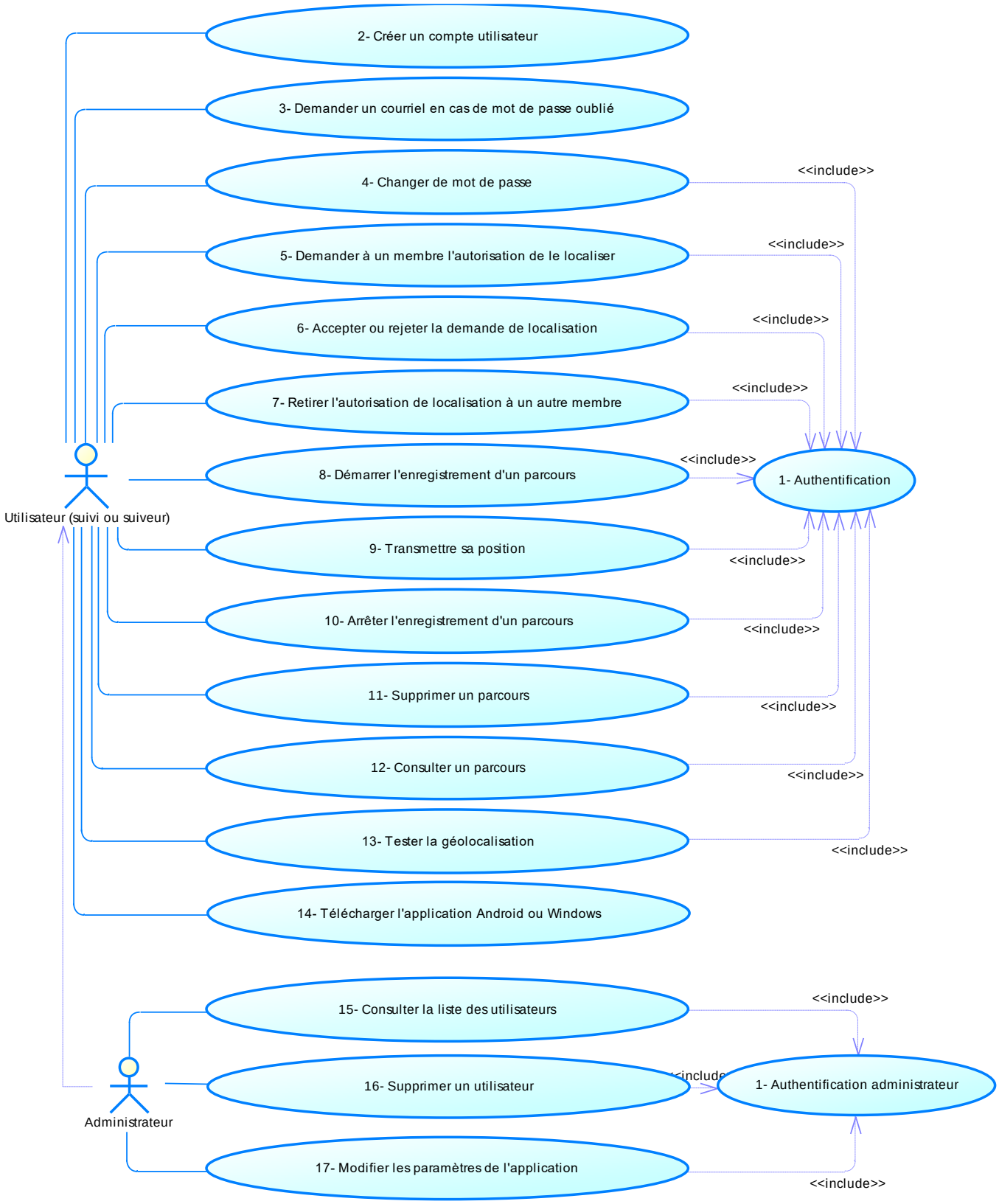


Répartition des tests de la classe DAO

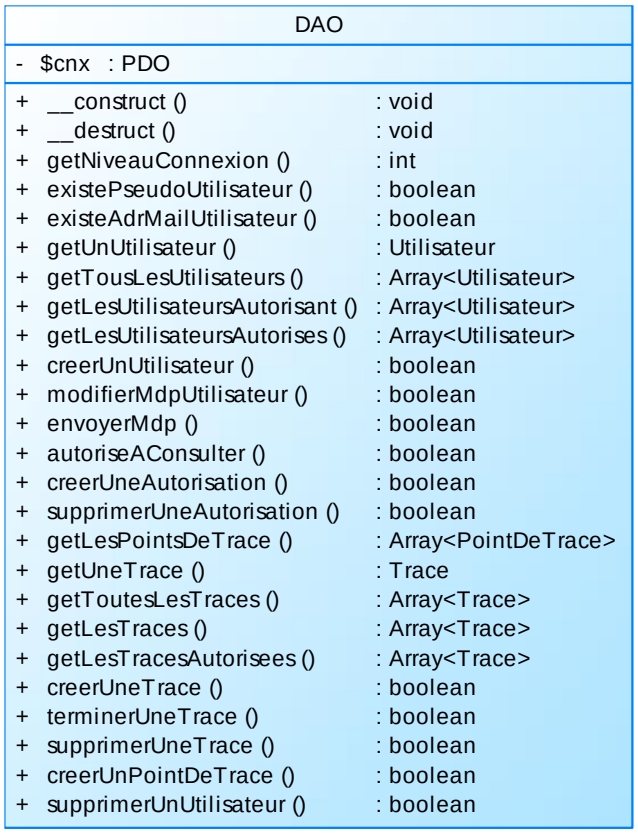
Le fichier **DAO.test.php** contient les tests des méthodes fournies. Vous pouvez l'exécuter pour vérifier le bon fonctionnement des méthodes fournies.

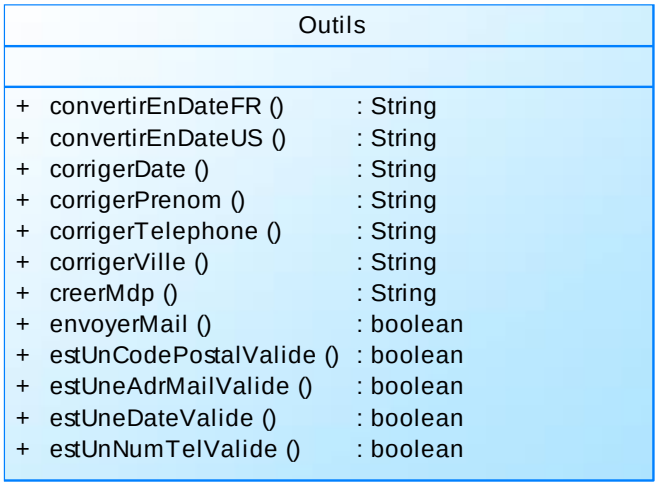
Les autres pages de tests seront réparties entre les développeurs de l'équipe afin d'y placer les tests des méthodes dont ils ont la responsabilité.

2- Rappel du diagramme des cas d'utilisation



3- Diagrammes UML des classes DAO et Outils





4- Liens entre cas d'utilisation et méthodes

Pour chaque cas d'utilisation, trouver les méthodes des classes **DAO** et **Outils** à utiliser et compléter le tableau :

Cas d'utilisation	***Méthodes utiles***
1- Authentification	DAO.getNiveauConnexion
1- Authentification administrateur	DAO.getNiveauConnexion
2- Créer un compte utilisateur	DAO.existePseudoUtilisateur DAO.existeAdrMailUtilisateur Outils.estUneAdrMailValide Outils.estUnNumTelValide Outils.corrigerTelephone Outils.creerMdp DAO.creerUnUtilisateur
3- Demander un courriel en cas de mot de passe oublié	
4- Changer de mot de passe	
5- Demander à un membre l'autorisation de le localiser	
6- Accepter ou rejeter la demande de localisation	
7- Retirer l'autorisation de localisation à un autre membre	
8- Démarrer l'enregistrement d'un parcours	
9- Transmettre sa position	
10- Arrêter l'enregistrement d'un parcours	
11- Supprimer un parcours	
12- Consulter un parcours	
13- Tester la géolocalisation	
14- Télécharger l'application Android ou Windows	
15- Consulter la liste des utilisateurs	
16- Supprimer un utilisateur	
17- Modifier les paramètres de l'application	

5- Présentation du code déjà réalisé et des tests

5-1 Liste des méthodes déjà réalisées :

1. getNiveauConnexion
2. existePseudoUtilisateur
3. getUnUtilisateur
4. getTousLesUtilisateurs
5. creerUnUtilisateur
6. modifierMdpUtilisateur
7. envoyerMdp
8. supprimerUnUtilisateur

Ces méthodes ont été choisies afin de fournir des modèles suffisamment variés pour développer les autres méthodes.

5-2 Description et test des méthodes déjà réalisées

5-2-1 Méhode getNiveauConnexion

getNiveauConnexion($pseudo, $mdpSha1)
Rôle : fournit le niveau de connexion d'un utilisateur identifié par **$pseudo** et **$mdpSha1**
Paramètres à fournir : **$pseudo** : le pseudo de l'utilisateur **$mdpSha1** : son mot de passe hashé en SHA1
Valeur de retour : un entier **0** : authentification incorrecte **1** : authentification correcte d'un utilisateur (pratiquant ou personne autorisée) **2** : authentification correcte d'un administrateur

Code PHP du test fonctionnel :

```php
<?php
// Projet TraceGPS
// fichier : modele/DAO.test.php
// Rôle : test de la classe DAO.class.php
// Dernière mise à jour : 12/8/2021 par dP
?>
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Test de la classe DAO</title>
    <style type="text/css">body {font-family: Arial, Helvetica, sans-serif; font-size: small;}</style>
</head>
<body>

<?php
// connexion du serveur web à la base MySQL
include_once ('DAO.class.php');
//include_once ('_DAO.mysql.class.php');
$dao = new DAO();

// test de la méthode getNiveauConnexion ----------------------------------------------------------
// modifié par DP le 12/8/2021
echo "<h3>Test de getNiveauConnexion : </h3>";
$niveau = $dao->getNiveauConnexion("admin", sha1("mdpadmin"));
echo "<p>Niveau de ('admin', 'mdpadmin') : " . $niveau . "</br>";

$niveau = $dao->getNiveauConnexion("europa", sha1("mdputilisateur"));
```

```
echo "<p>Niveau de ('europa', 'mdputilisateur') : " . $niveau . "</br>";

$niveau = $dao->getNiveauConnexion("europa", sha1("123456"));
echo "<p>Niveau de ('europa', '123456') : " . $niveau . "</br>";

$niveau = $dao->getNiveauConnexion("toto", sha1("mdputilisateur"));
echo "<p>Niveau de ('toto', 'mdputilisateur') : " . $niveau . "</br>";
```

Les autres tests seront placés à la suite

```
// ferme la connexion à MySQL :
unset($dao);
?>

</body>
</html>
```

Résultat attendu avec le test :

Test de getNiveauConnexion :

Niveau de ('admin', 'mdpadmin') : 2
Niveau de ('europa', 'mdputilisateur') : 1
Niveau de ('europa', '123456') : 0
Niveau de ('toto', 'mdputilisateur') : 0

Les résultats des tests dépendent bien sûr du contenu de votre base de données

5-2-2 Méhode existePseudoUtilisateur

existePseudoUtilisateur($pseudo)
Rôle : indique si **$pseudo** existe dans la table **tracegps_utilisateurs**
Paramètres à fournir : **$pseudo** : le pseudo de l'utilisateur
Valeur de retour : un booléen **true** si le pseudo **$pseudo** existe dans la table **tracegps_utilisateurs** **false** sinon

Code PHP du test fonctionnel :

```
// test de la méthode existePseudoUtilisateur ----------------------------------------------------
// modifié par DP le 12/8/2021
echo "<h3>Test de existePseudoUtilisateur : </h3>";
if ($dao->existePseudoUtilisateur("admin")) $existe = "oui"; else $existe = "non";
echo "<p>Existence de l'utilisateur 'admin' : <b>" . $existe . "</b><br>";
if ($dao->existePseudoUtilisateur("europa")) $existe = "oui"; else $existe = "non";
echo "Existence de l'utilisateur 'europa' : <b>" . $existe . "</b></br>";
if ($dao->existePseudoUtilisateur("toto")) $existe = "oui"; else $existe = "non";
echo "Existence de l'utilisateur 'toto' : <b>" . $existe . "</b></p>";
```

Résultat attendu avec le test :

Test de existePseudoUtilisateur :

Existence de l'utilisateur 'admin' : oui
Existence de l'utilisateur 'europa' : oui
Existence de l'utilisateur 'toto' : non

Les résultats des tests dépendent bien sûr du contenu de votre base de données

5-2-3 Méhode getUnUtilisateur

getUnUtilisateur($pseudo)
Rôle : fournit un objet **Utilisateur** à partir de son pseudo **$pseudo**
Paramètres à fournir : **$pseudo** : le pseudo de l'utilisateur
Valeur de retour : un objet un objet de la classe **Utilisateur** si **$pseudo** existe l'objet null si **$pseudo** n'existe pas

Code PHP du test fonctionnel :

```php
// test de la méthode getUnUtilisateur ------------------------------------------------------------
// modifié par dP le 12/8/2021
echo "<h3>Test de getUnUtilisateur : </h3>";
$unUtilisateur = $dao->getUnUtilisateur("admin");
if ($unUtilisateur) {
    echo "<p>L'utilisateur admin existe : <br>" . $unUtilisateur->toString() . "</p>";
}
else {
    echo "<p>L'utilisateur admin n'existe pas !</p>";
}
$unUtilisateur = $dao->getUnUtilisateur("europa");
if ($unUtilisateur) {
    echo "<p>L'utilisateur europa existe : <br>" . $unUtilisateur->toString() . "</p>";
}
else {
    echo "<p>L'utilisateur europa n'existe pas !</p>";
}
$unUtilisateur = $dao->getUnUtilisateur("admon");
if ($unUtilisateur) {
    echo "<p>L'utilisateur admon existe : <br>" . $unUtilisateur->toString() . "</p>";
}
else {
    echo "<p>L'utilisateur admon n'existe pas !</p>";
}
```

Résultat attendu avec le test :

Test de getUnUtilisateur :

L'utilisateur admin existe :
id : 1
pseudo : admin
mdpSha1 : ff9fff929a1292db1c00e3142139b22ee4925177
adrMail : delasalle.sio.eleves@gmail.com
numTel : 11.22.33.44.55
niveau : 2
dateCreation : 2018-08-12 19:45:23
nbTraces : 0
dateDerniereTrace :

Les résultats des tests dépendent bien sûr du contenu de votre base de données

L'utilisateur europa existe :
id : 3
pseudo : europa
mdpSha1 : 13e3668bbee30b004380052b086457b014504b3e
adrMail : delasalle.sio.eleves@gmail.com
numTel : 22.33.44.55.66

```
niveau : 1
dateCreation : 2018-08-12 19:45:23
nbTraces : 2
dateDerniereTrace : 2018-01-19 13:08:48

L'utilisateur admon n'existe pas !
```

5-2-4 Méhode getTousLesUtilisateurs

getTousLesUtilisateurs()
Rôle : fournit la collection de tous les utilisateurs (de niveau 1)
Paramètres à fournir : aucun
Valeur de retour : une collection d'objets **Utilisateur**

Code PHP du test fonctionnel :

```php
// test de la méthode getTousLesUtilisateurs ----------------------------------------------------
// modifié par dP le 12/8/2021
echo "<h3>Test de getTousLesUtilisateurs : </h3>";
$lesUtilisateurs = $dao->getTousLesUtilisateurs();
$nbReponses = sizeof($lesUtilisateurs);
echo "<p>Nombre d'utilisateurs : " . $nbReponses . "</p>";
// affichage des utilisateurs
foreach ($lesUtilisateurs as $unUtilisateur)
{       echo ($unUtilisateur->toString());
    echo ('<br>');
}
```

Résultat attendu avec le test :

Test de getTousLesUtilisateurs :

Nombre d'utilisateurs : 12

Les résultats des tests dépendent bien sûr du contenu de votre base de données

```
id : 2
pseudo : callisto
mdpSha1 : 13e3668bbee30b004380052b086457b014504b3e
adrMail : delasalle.sio.eleves@gmail.com
numTel : 22.33.44.55.66
niveau : 1
dateCreation : 2018-08-12 19:45:23
nbTraces : 2
dateDerniereTrace : 2018-01-19 13:08:48

id : 3
pseudo : europa
mdpSha1 : 13e3668bbee30b004380052b086457b014504b3e
adrMail : delasalle.sio.eleves@gmail.com
numTel : 22.33.44.55.66
niveau : 1
dateCreation : 2018-08-12 19:45:23
nbTraces : 2
dateDerniereTrace : 2018-01-19 13:08:48

id : 4
pseudo : galileo
mdpSha1 : 13e3668bbee30b004380052b086457b014504b3e
adrMail : delasalle.sio.eleves@gmail.com
numTel : 22.33.44.55.66
```

```
niveau : 1
dateCreation : 2018-08-12 19:45:23
nbTraces : 2
dateDerniereTrace : 2018-01-19 13:08:48

id : 5
pseudo : helios
mdpSha1 : 13e3668bbee30b004380052b086457b014504b3e
adrMail : delasalle.sio.eleves@gmail.com
numTel : 33.44.55.66.77
niveau : 1
dateCreation : 2018-08-12 19:45:23
nbTraces : 2
dateDerniereTrace : 2018-01-19 13:08:48

id : 6
pseudo : indigo
mdpSha1 : 13e3668bbee30b004380052b086457b014504b3e
adrMail : delasalle.sio.eleves@gmail.com
numTel : 44.55.66.77.88
niveau : 1
dateCreation : 2018-08-12 19:45:23
nbTraces : 2
dateDerniereTrace : 2018-01-19 13:08:48

id : 7
pseudo : juno
mdpSha1 : 13e3668bbee30b004380052b086457b014504b3e
adrMail : delasalle.sio.eleves@gmail.com
numTel : 44.55.66.77.88
niveau : 1
dateCreation : 2018-08-12 19:45:23
nbTraces : 2
dateDerniereTrace : 2018-01-19 13:08:48

id : 8
pseudo : kepler
mdpSha1 : 13e3668bbee30b004380052b086457b014504b3e
adrMail : delasalle.sio.eleves@gmail.com
numTel : 44.55.66.77.88
niveau : 1
dateCreation : 2018-08-12 19:45:23
nbTraces : 2
dateDerniereTrace : 2018-01-19 13:08:48

id : 9
pseudo : luna
mdpSha1 : 13e3668bbee30b004380052b086457b014504b3e
adrMail : delasalle.sio.eleves@gmail.com
numTel : 44.55.66.77.88
niveau : 1
dateCreation : 2018-08-12 19:45:23
nbTraces : 2
dateDerniereTrace : 2018-01-19 13:08:48

id : 10
pseudo : mars
mdpSha1 : 13e3668bbee30b004380052b086457b014504b3e
adrMail : delasalle.sio.eleves@gmail.com
numTel : 44.55.66.77.88
niveau : 1
```

```
dateCreation : 2018-08-12 19:45:23
nbTraces : 2
dateDerniereTrace : 2018-01-19 13:08:48

id : 11
pseudo : neon
mdpSha1 : 13e3668bbee30b004380052b086457b014504b3e
adrMail : delasalle.sio.eleves@gmail.com
numTel : 44.55.66.77.88
niveau : 1
dateCreation : 2018-08-12 19:45:23
nbTraces : 2
dateDerniereTrace : 2018-01-19 13:08:48

id : 12
pseudo : oxygen
mdpSha1 : 13e3668bbee30b004380052b086457b014504b3e
adrMail : delasalle.sio.eleves@gmail.com
numTel : 44.55.66.77.88
niveau : 1
dateCreation : 2018-08-12 19:45:23
nbTraces : 2
dateDerniereTrace : 2018-01-19 13:08:48

id : 13
pseudo : photon
mdpSha1 : 13e3668bbee30b004380052b086457b014504b3e
adrMail : delasalle.sio.eleves@gmail.com
numTel : 44.55.66.77.88
niveau : 1
dateCreation : 2018-08-12 19:45:23
nbTraces : 0
dateDerniereTrace :
```

5-2-5 Méhode creerUnUtilisateur

creerUnUtilisateur($unUtilisateur)
Rôle : enregistre l'utilisateur **$unUtilisateur** dans la table **tracegps_utilisateurs**
Paramètres à fournir : **$unUtilisateur** : l'utilisateur à enregistrer (un objet de la classe **Utilisateur**)
Valeur de retour : un booléen **true** si l'enregistrement s'est bien passé **false** sinon

Code PHP du test fonctionnel :

```
// test de la méthode creerUnUtilisateur ----------------------------------------------------------
// modifié par dP le 12/8/2021
echo "<h3>Test de creerUnUtilisateur : </h3>";
$unUtilisateur = new Utilisateur(0, "toto", "mdputilisateur", "toto@gmail.com", "5566778899", 1, date('Y-m-d H:i:s', time()), 0, null);
$ok = $dao->creerUnUtilisateur($unUtilisateur);
if ($ok)
{   echo "<p>Utilisateur bien enregistré !</p>";
    echo $unUtilisateur->toString();
}
else {
    echo "<p>Echec lors de l'enregistrement de l'utilisateur !</p>";
}
```

Résultat attendu avec le test :

Test de creerUnUtilisateur :

Utilisateur bien enregistré !
id : 24
pseudo : toto
mdpSha1 : mdputilisateur
adrMail : toto@gmail.com
numTel : 55.66.77.88.99
niveau : 1
dateCreation : 2018-08-12 20:29:16
nbTraces : 0
dateDerniereTrace :

Les résultats des tests dépendent bien sûr du contenu de votre base de données

5-2-6 Méhode modifierMdpUtilisateur

modifierMdpUtilisateur($pseudo, $nouveauMdp)
Rôle : enregistre le nouveau mot de passe **$nouveauMdp** de l'utilisateur **$pseudo** daprès l'avoir hashé en SHA1
Paramètres à fournir : **$pseudo** : le pseudo de l'utilisateur à modifiier **$nouveauMdp** : le nouveau mot de passe (en clair)
Valeur de retour : un booléen **true** si la modification s'est bien passée **false** sinon

Code PHP du test fonctionnel :

```php
// test de la méthode modifierMdpUtilisateur ----------------------------------------------------
// modifié par dP le 12/8/2021
echo "<h3>Test de modifierMdpUtilisateur : </h3>";
$unUtilisateur = $dao->getUnUtilisateur("toto");
if ($unUtilisateur) {
    echo "<p>Ancien mot de passe de l'utilisateur toto : <b>" . $unUtilisateur->getMdpSha1() . "</b><br>";
    $dao->modifierMdpUtilisateur("toto", "mdpadmin");
    $unUtilisateur = $dao->getUnUtilisateur("toto");
    echo "Nouveau mot de passe de l'utilisateur toto : <b>" . $unUtilisateur->getMdpSha1() . "</b><br>";

    $niveauDeConnexion = $dao->getNiveauConnexion('toto', sha1('mdputilisateur'));
    echo "Niveau de connexion de ('toto', 'mdputilisateur') : <b>" . $niveauDeConnexion . "</b><br>";

    $niveauDeConnexion = $dao->getNiveauConnexion('toto', sha1('mdpadmin'));
    echo "Niveau de connexion de ('toto', 'mdpadmin') : <b>" . $niveauDeConnexion . "</b></p>";
}
else {
    echo "<p>L'utilisateur toto n'existe pas !</p>";
}
```

Résultat attendu avec le test :

Test de modifierMdpUtilisateur :

Ancien mot de passe de l'utilisateur toto : **13e3668bbee30b004380052b086457b014504b3e**
Nouveau mot de passe de l'utilisateur toto : **ff9fff929a1292db1c00e3142139b22ee4925177**
Niveau de connexion de ('toto', 'mdputilisateur') : **0**
Niveau de connexion de ('toto', 'mdpadmin') : **1**

Les résultats des tests dépendent bien sûr du contenu de votre base de données

5-2-7 Méhode supprimerUnUtilisateur

supprimerUnUtilisateur($pseudo)
Rôle : supprime l'utilisateur **$pseudo** dans la bdd, ainsi que ses traces et ses autorisations
Paramètres à fournir : **$pseudo** : le pseudo de l'utilisateur à supprimer
Valeur de retour : un booléen **true** si l'effacement s'est bien passé **false** sinon

Code PHP du test fonctionnel :

```php
// test de la méthode supprimerUnUtilisateur ----------------------------------------------------
// modifié par dP le 12/8/2021
echo "<h3>Test de supprimerUnUtilisateur : </h3>";
$ok = $dao->supprimerUnUtilisateur("toto");
if ($ok) {
    echo "<p>Utilisateur toto bien supprimé !</p>";
}
else {
    echo "<p>Echec lors de la suppression de l'utilisateur toto !</p>";
}
$ok = $dao->supprimerUnUtilisateur("toto");
if ($ok) {
    echo "<p>Utilisateur toto bien supprimé !</p>";
}
else {
    echo "<p>Echec lors de la suppression de l'utilisateur toto !</p>";
}
```

Résultat attendu avec le test :

Test de supprimerUnUtilisateur :

Utilisateur toto bien supprimé !
Echec lors de la suppression de l'utilisateur toto !

Les résultats des tests dépendent bien sûr du contenu de votre base de données

5-2-8 Méhode envoyerMdp

envoyerMdp($pseudo, $nouveauMdp)
Rôle : envoie un mail à l'utilisateur **$pseudo** avec son nouveau mot de passe **$nouveauMdp** en clair
Paramètres à fournir : **$pseudo** : le pseudo de l'utilisateur à prévenir **$nouveauMdp** : le nouveau mot de passe (en clair)
Valeur de retour : un booléen **true** si l'envoi s'est bien passé **false** sinon

Code PHP du test fonctionnel :

```php
// test de la méthode envoyerMdp ----------------------------------------------------------------
// modifié par dP le 12/8/2021
echo "<h3>Test de envoyerMdp : </h3>";
// pour ce test, une adresse mail que vous pouvez consulter
$unUtilisateur = new Utilisateur(0, "toto", "mdputilisateur", "delasalle.sio.xxxxx@gmail.com", "5566778899", 2,
date('Y-m-d H:i:s', time()), 0, null);
$ok = $dao->creerUnUtilisateur($unUtilisateur);
$dao->modifierMdpUtilisateur("toto", "mdpadmin");
$ok = $dao->envoyerMdp("toto", "mdpadmin");
if ($ok) {
    echo "<p>Mail bien envoyé !</p>";
}
else {
    echo "<p>Echec lors de l'envoi du mail !</p>";
}
// supprimer le compte créé
$ok = $dao->supprimerUnUtilisateur("toto");
if ($ok) {
    echo "<p>Utilisateur toto bien supprimé !</p>";
}
else {
    echo "<p>Echec lors de la suppression de l'utilisateur toto !</p>";
}
```

Pour vos tests, utilisez une de vos adresses mail que vous pourrez consulter

Résultat attendu avec le test :

Test de envoyerMdp :
Mail bien envoyé !
Utilisateur toto bien supprimé !

Les résultats des tests dépendent bien sûr du contenu de votre base de données

6- Code restant à développer et organisation de la répartition du travail

6-1 Liste des méthodes à réaliser :

Gestion des utilisateurs

1. existeAdrMailUtilisateur

Gestion des autorisations

2. getLesUtilisateursAutorisant
3. getLesUtilisateursAutorises
4. autoriseAConsulter
5. creerUneAutorisation
6. supprimerUneAutorisation

Gestion des points de traces

7. getLesPointsDeTrace
8. creerUnPointDeTrace

Gestion des traces

9. getUneTrace
10. getToutesLesTraces
11. getLesTraces
12. getLesTracesAutorisees
13. creerUneTrace
14. supprimerUneTrace
15. terminerUneTrace

6-2 Organisation du travail collaboratif

Le code restant à développer va être réparti entre les membres de l'équipe de développement. Afin de limiter les conflits avec GitHub, il est décidé un fichier de test à chaque développeur.

- **DAO.test1.php** : la page de test de la classe **DAO** (à compléter par le développeur 1)
- **DAO.test2.php** : la page de test de la classe **DAO** (à compléter par le développeur 2)
- **DAO.test3.php** : la page de test de la classe **DAO** (à compléter par le développeur 3)
- **DAO.test4.php** : la page de test de la classe **DAO** (à compléter par le développeur 4)

Quelques conseils pour le travail collaboratif :

- Avant d'attaquer un cycle de développement (début de séance, nouvelle méthode, ...), faites un **Pull** pour récupérer la dernière version du fichier.
- Après avoir testé et validé une méthode, faites un **Commit** et un **Push** pour transmettre cette version aux autres développeurs.

6-3 Le suivi de l'avancement du projet

Le fichier Excel fourni dans votre équipe Teams placé dans le canal AP-Equipe…. sera utilisé pour le suivi de l'avancement du projet.

Il sera renseigné en permanence par les membres :

- au démarrage d'une tâche
- à la fin d'une tâche

Chaque membre de l'équipe pourra utiliser une couleur personnelle.

6-3 Description des méthodes à réaliser

6-3-1 Méthode existeAdrMailUtilisateur

existeAdrMailUtilisateur($adrMail)
Rôle : indique si **$adrMail** existe dans la table **tracegps_utilisateurs**
Paramètres à fournir : **$adrMail** : une adresse mail
Valeur de retour : un booléen **true** si l'adresse **$adrMail** existe dans la table **tracegps_utilisateurs** **false** sinon

Code PHP du test fonctionnel :

```php
// test de la méthode existeAdrMailUtilisateur ----------------------------------------------------
// modifié par dP le 12/8/2021
echo "<h3>Test de existeAdrMailUtilisateur : </h3>";
if ($dao->existeAdrMailUtilisateur("admin@gmail.com")) $existe = "oui"; else $existe = "non";
echo "<p>Existence de l'utilisateur 'admin@gmail.com' : <b>" . $existe . "</b><br>";
if ($dao->existeAdrMailUtilisateur("delasalle.sio.eleves@gmail.com")) $existe = "oui"; else $existe = "non";
echo "Existence de l'utilisateur 'delasalle.sio.eleves@gmail.com' : <b>" . $existe . "</b></br>";
```

Résultat attendu avec le test :

Test de existeAdrMailUtilisateur :

Existence de l'utilisateur 'admin@gmail.com' : **non**
Existence de l'utilisateur 'delasalle.sio.eleves@gmail.com' : **oui**

Les résultats des tests dépendent bien sûr du contenu de votre base de données

6-3-2 Méthode getLesUtilisateursAutorisant

getLesUtilisateursAutorisant($idUtilisateur)
Rôle : fournit la collection des utilisateurs (de niveau 1) autorisant l'utilisateur **$idUtilisateur** à voir leurs parcours
Paramètres à fournir : **$idUtilisateur** : identifiant de l'utilisateur autorisé à consulter des parcours
Valeur de retour : collection d'objets **Utilisateur** la collection des utilisateurs qui ont donné l'autorisation à **$idUtilisateur**

Code PHP du test fonctionnel :

```php
// test de la méthode getLesUtilisateursAutorisant ------------------------------------------------
// modifié par dP le 13/8/2021
echo "<h3>Test de getLesUtilisateursAutorisant(idUtilisateur) : </h3>";
$lesUtilisateurs = $dao->getLesUtilisateursAutorisant(4);
$nbReponses = sizeof($lesUtilisateurs);
echo "<p>Nombre d'utilisateurs autorisant l'utilisateur 4 à voir leurs parcours : " . $nbReponses . "</p>";
// affichage des utilisateurs
foreach ($lesUtilisateurs as $unUtilisateur)
{   echo ($unUtilisateur->toString());
    echo ('<br>');
}
```

Résultat attendu avec le test :

Test de getLesUtilisateursAutorisant(idUtilisateur) :

Nombre d'utilisateurs autorisant l'utilisateur 4 à voir leurs parcours : 2

id : 2
pseudo : callisto
mdpSha1 : 13e3668bbee30b004380052b086457b014504b3e
adrMail : delasalle.sio.eleves@gmail.com
numTel : 22.33.44.55.66
niveau : 1
dateCreation : 2018-08-12 19:45:23
nbTraces : 2
dateDerniereTrace : 2018-01-19 13:08:48

Les résultats des tests dépendent bien sûr du contenu de votre base de données

id : 3
pseudo : europa
mdpSha1 : 13e3668bbee30b004380052b086457b014504b3e
adrMail : delasalle.sio.eleves@gmail.com
numTel : 22.33.44.55.66
niveau : 1
dateCreation : 2018-08-12 19:45:23
nbTraces : 2
dateDerniereTrace : 2018-01-19 13:08:48

6-3-3 Méthode getLesUtilisateursAutorises

getLesUtilisateursAutorises($idUtilisateur)
Rôle : fournit la collection des utilisateurs (de niveau 1) autorisés à voir les parcours de l'utilisateur **$idUtilisateur**
Paramètres à fournir : **$idUtilisateur** : identifiant de l'utilisateur autorisant à consulter ses parcours
Valeur de retour : collection d'objets **Utilisateur** la collection des utilisateurs qui sont autorisés à voir les parcours de l'utilisateur **$idUtilisateur**

Code PHP du test fonctionnel :

```
// test de la méthode getLesUtilisateursAutorises ------------------------------------------------
// modifié par dP le 13/8/2021
echo "<h3>Test de getLesUtilisateursAutorises(idUtilisateur) : </h3>";
$lesUtilisateurs = $dao->getLesUtilisateursAutorises(2);
$nbReponses = sizeof($lesUtilisateurs);
echo "<p>Nombre d'utilisateurs autorisés par l'utilisateur 2 : " . $nbReponses . "</p>";
// affichage des utilisateurs
foreach ($lesUtilisateurs as $unUtilisateur)
{       echo ($unUtilisateur->toString());
    echo ('<br>');
}
```

Résultat attendu avec le test :

Test de getLesUtilisateursAutorises(idUtilisateur) :

Nombre d'utilisateurs autorisés par l'utilisateur 2 : 2

id : 3
pseudo : europa
mdpSha1 : 13e3668bbee30b004380052b086457b014504b3e
adrMail : delasalle.sio.eleves@gmail.com
numTel : 22.33.44.55.66
niveau : 1
dateCreation : 2018-08-12 19:45:23
nbTraces : 2
dateDerniereTrace : 2018-01-19 13:08:48

Les résultats des tests dépendent bien sûr du contenu de votre base de données

id : 4
pseudo : galileo
mdpSha1 : 13e3668bbee30b004380052b086457b014504b3e
adrMail : delasalle.sio.eleves@gmail.com
numTel : 22.33.44.55.66
niveau : 1
dateCreation : 2018-08-12 19:45:23
nbTraces : 2
dateDerniereTrace : 2018-01-19 13:08:48

6-3-4 Méthode autoriseAConsulter

autoriseAConsulter($idAutorisant, $idAutorise)
Rôle : indique si l'utilisateur **$idAutorisant** autorise l'utilisateur **$idAutorise** à consulter ses traces
Paramètres à fournir : **$idAutorisant** : l'id de l'utilisateur qui autorise **$idAutorise** : l'id de l'utilisateur qui est autorisé
Valeur de retour : un booléen **true** si l'autorisation est donnée **false** sinon

Code PHP du test fonctionnel :

```
// test de la méthode autoriseAConsulter ------------------------------------------------------
// modifié par dP le 13/8/2021
echo "<h3>Test de autoriseAConsulter : </h3>";
if ($dao->autoriseAConsulter(2, 3)) $autorise = "oui"; else $autorise = "non";
echo "<p>L'utilisateur 2 autorise l'utilisateur 3 : <b>" . $autorise . "</b><br>";

if ($dao->autoriseAConsulter(3, 2)) $autorise = "oui"; else $autorise = "non";
echo "<p>L'utilisateur 3 autorise l'utilisateur 2 : <b>" . $autorise . "</b><br>";
```

Résultat attendu avec le test :

Test de autoriseAConsulter :

L'utilisateur 2 autorise l'utilisateur 3 : **oui**
L'utilisateur 3 autorise l'utilisateur 2 : **non**

Les résultats des tests dépendent bien sûr du contenu de votre base de données

6-3-5 Méthode creerUneAutorisation

creerUneAutorisation($idAutorisant, $idAutorise)
Rôle : enregistre l'autorisation (**$idAutorisant, $idAutorise**) dans la table **tracegps_autorisations**
Paramètres à fournir : **$idAutorisant** : l'id de l'utilisateur qui autorise **$idAutorise** : l'id de l'utilisateur qui est autorisé
Valeur de retour : un booléen **true** si l'enregistrement s'est bien passé **false** sinon

Code PHP du test fonctionnel :

```
// test de la méthode creerUneAutorisation ----------------------------------------------------------
// modifié par dP le 13/8/2021
echo "<h3>Test de creerUneAutorisation : </h3>";
if ($dao->creerUneAutorisation(2, 1)) $ok = "oui"; else $ok = "non";
echo "<p>La création de l'autorisation de l'utilisateur 2 vers l'utilisateur 1 a réussi : <b>" . $ok . "</b><br>";
// la même autorisation ne peut pas être enregistrée 2 fois
if ($dao->creerUneAutorisation(2, 1)) $ok = "oui"; else $ok = "non";
echo "<p>La création de l'autorisation de l'utilisateur 2 vers l'utilisateur 1 a réussi : <b>" . $ok . "</b><br>";
```

Résultat attendu avec le test :

Les résultats des tests dépendent bien sûr du contenu de votre base de données

Test de creerUneAutorisation :

La création de l'autorisation de l'utilisateur 2 vers l'utilisateur 1 a réussi : **oui**
La création de l'autorisation de l'utilisateur 2 vers l'utilisateur 1 a réussi : **non**

6-3-6 Méthode supprimerUneAutorisation

supprimerUneAutorisation($idAutorisant, $idAutorise)
Rôle : supprime l'autorisation (**$idAutorisant, $idAutorise**) dans la table **tracegps_autorisations**
Paramètres à fournir : **$idAutorisant** : l'id de l'utilisateur qui autorise **$idAutorise** : l'id de l'utilisateur qui est autorisé
Valeur de retour : un booléen **true** si la suppression s'est bien passée **false** sinon

Code PHP du test fonctionnel :

```
// test de la méthode supprimerUneAutorisation ----------------------------------------------------
// modifié par dP le 13/8/2021
echo "<h3>Test de supprimerUneAutorisation : </h3>";
// on crée une autorisation
if ($dao->creerUneAutorisation(2, 1)) $ok = "oui"; else $ok = "non";
echo "<p>La création de l'autorisation de l'utilisateur 2 vers l'utilisateur 1 a réussi : <b>" . $ok . "</b><br>";
// puis on la supprime
if ($dao->supprimerUneAutorisation(2, 1)) $ok = "oui"; else $ok = "non";
echo "<p>La suppression de l'autorisation de l'utilisateur 2 vers l'utilisateur 1 a réussi : <b>" . $ok . "</b><br>";
```

Résultat attendu avec le test :

Test de supprimerUneAutorisation :

La création de l'autorisation de l'utilisateur 2 vers l'utilisateur 1 a réussi : **non**
La suppression de l'autorisation de l'utilisateur 2 vers l'utilisateur 1 a réussi : **oui**

6-3-7 Méthode getLesPointsDeTrace

getLesPointsDeTrace($idTrace)
Rôle : fournit la collection des points de la trace **$idTrace**
Paramètres à fournir : **$idTrace** : identifiant de la trace
Valeur de retour : collection d'objets **PointDeTrace** la collection des points de la trace **$idTrace**

Code PHP du test fonctionnel :

```php
// test de la méthode getLesPointsDeTrace ------------------------------------------------------
// modifié par dP le 13/8/2021
echo "<h3>Test de getLesPointsDeTrace : </h3>";
$lesPoints = $dao->getLesPointsDeTrace(1);
$nbPoints = sizeof($lesPoints);
echo "<p>Nombre de points de la trace 1 : " . $nbPoints . "</p>";
// affichage des points
foreach ($lesPoints as $unPoint)
{   echo ($unPoint->toString());
    echo ('<br>');
}
```

Résultat attendu avec le test :

Test de getLesPointsDeTrace :

Nombre de points de la trace 1 : 5

IdTrace : 1
Id : 1
latitude : 48.2109
longitude : -1.5535
altitude : 60
Heure de passage : 2018-01-19 13:08:48
Rythme cardiaque : 81
Temps cumule (s) : 0
Temps cumule (hh:mm:ss) : 00:00:00
Distance cumulée (Km) : 0
Vitesse (Km/h) : 0

IdTrace : 1
Id : 2
latitude : 48.2119
longitude : -1.5525
altitude : 70
Heure de passage : 2018-01-19 13:09:08
Rythme cardiaque : 82
Temps cumule (s) : 0
Temps cumule (hh:mm:ss) : 00:00:00
Distance cumulée (Km) : 0
Vitesse (Km/h) : 0

IdTrace : 1
Id : 3
latitude : 48.2129
longitude : -1.5515
altitude : 80

```
Heure de passage : 2018-01-19 13:09:28
Rythme cardiaque : 83
Temps cumule (s) : 0
Temps cumule (hh:mm:ss) : 00:00:00
Distance cumulée (Km) : 0
Vitesse (Km/h) : 0

IdTrace : 1
Id : 4
latitude : 48.2139
longitude : -1.5505
altitude : 90
Heure de passage : 2018-01-19 13:09:48
Rythme cardiaque : 84
Temps cumule (s) : 0
Temps cumule (hh:mm:ss) : 00:00:00
Distance cumulée (Km) : 0
Vitesse (Km/h) : 0

IdTrace : 1
Id : 5
latitude : 48.2149
longitude : -1.5495
altitude : 100
Heure de passage : 2018-01-19 13:10:08
Rythme cardiaque : 85
Temps cumule (s) : 0
Temps cumule (hh:mm:ss) : 00:00:00
Distance cumulée (Km) : 0
Vitesse (Km/h) : 0
```

6-3-8 Méthode creerUnPointDeTrace

creerUnPointDeTrace($unPointDeTrace)
Rôle : enregistre le point **$unPointDeTrace** dans la bdd
Paramètres à fournir : **$unPointDeTrace** : le point de trace à enregistrer
Valeur de retour : un booléen **true** si l'enregistrement s'est bien passé **false** sinon
Particularité : Si le point est le premier d'une trace ($id = 1), il faut modifier la date de début de la trace en lui affectant la date du point

Code PHP du test fonctionnel :

```
// test de la méthode creerUnPointDeTrace ------------------------------------------------------
// modifié par dP le 13/8/2021
echo "<h3>Test de creerUnPointDeTrace : </h3>";
// on affiche d'abord le nombre de points (5) de la trace 1
$lesPoints = $dao->getLesPointsDeTrace(1);
$nbPoints = sizeof($lesPoints);
echo "<p>Nombre de points de la trace 1 : " . $nbPoints . "</p>";
// on crée un sixième point et on l'ajoute à la trace 1
$unIdTrace = 1;
$unID = 6;
$uneLatitude = 48.20;
$uneLongitude = -1.55;
$uneAltitude = 50;
$uneDateHeure = date('Y-m-d H:i:s', time());
$unRythmeCardio = 80;
$unTempsCumule = 0;
$uneDistanceCumulee = 0;
$uneVitesse = 15;
$unPoint = new PointDeTrace($unIdTrace, $unID, $uneLatitude, $uneLongitude, $uneAltitude, $uneDateHeure,
$unRythmeCardio, $unTempsCumule, $uneDistanceCumulee, $uneVitesse);
$ok = $dao->creerUnPointDeTrace($unPoint);
// on affiche à nouveau le nombre de points (6) de la trace 1
$lesPoints = $dao->getLesPointsDeTrace(1);
$nbPoints = sizeof($lesPoints);
echo "<p>Nombre de points de la trace 1 : " . $nbPoints . "</p>";
echo ('<br>');
```

Résultat attendu avec le test :

Test de creerUnPointDeTrace :

Nombre de points de la trace 1 : 5
Nombre de points de la trace 1 : 6

6-3-9 Méhode getUneTrace

getUneTrace($idTrace)
Rôle : fournit un objet **Trace** à partir de son identifiant **$idTrace**
Paramètres à fournir : **$idTrace** : l'identifiant de la trace
Valeur de retour : un objet un objet de la classe **Trace** si **$idTrace** existe l'objet **null** si **$idTrace** n'existe pas
Particularité : utiliser la méthode **getLesPointsDeTrace($idTrace)** pour obtenir les points de la trace et les ajouter à l'objet **Trace** qui sera retourné

Code PHP du test fonctionnel :

```
// test de la méthode getUneTrace ----------------------------------------------------------------
// modifié par dP le 14/8/2021
echo "<h3>Test de getUneTrace : </h3>";
$uneTrace = $dao->getUneTrace(2);
if ($uneTrace) {
    echo "<p>La trace 2 existe : <br>" . $uneTrace->toString() . "</p>";
}
else {
    echo "<p>La trace 2 n'existe pas !</p>";
}
$uneTrace = $dao->getUneTrace(100);
if ($uneTrace) {
    echo "<p>La trace 100 existe : <br>" . $uneTrace->toString() . "</p>";
}
else {
    echo "<p>La trace 100 n'existe pas !</p>";
}
```

Résultat attendu avec le test :

Test de getUneTrace :

La trace 2 existe :
Id : 2
Utilisateur : 2
Heure de début : 2018-01-19 13:08:48
Terminée : Oui
Nombre de points : 10
Heure de fin : 2018-01-19 13:11:48
Durée en secondes : 180
Durée totale : 00:03:00
Distance totale en Km : 1.2017846309768
Dénivelé en m : 90
Dénivelé positif en m : 90
Dénivelé négatif en m : 0
Vitesse moyenne en Km/h : 24.035692619536
Centre du parcours :
- Latitude : 48.2154
- Longitude : -1.549
- Altitude : 0

La trace 100 n'existe pas !

6-3-10 Méhode getToutesLesTraces

getToutesLesTraces()
Rôle : fournit la collection de toutes les traces
Paramètres à fournir : aucun
Valeur de retour : une collection d'objets **Trace**
Particularité : utiliser la méthode **getLesPointsDeTrace($idTrace)** pour obtenir les points de chaque trace et les ajouter à chaque objet **Trace** qui sera ajouté à la collection

Code PHP du test fonctionnel :

```
// test de la méthode getToutesLesTraces ----------------------------------------------------------
// modifié par dP le 14/8/2021
echo "<h3>Test de getToutesLesTraces : </h3>";
$lesTraces = $dao->getToutesLesTraces();
$nbReponses = sizeof($lesTraces);
echo "<p>Nombre de traces : " . $nbReponses . "</p>";
// affichage des traces
foreach ($lesTraces as $uneTrace)
{   echo ($uneTrace->toString());
    echo ('<br>');
}
```

Résultat attendu avec le test :

Test de getToutesLesTraces :

Nombre de traces : 22

Id : 22
Utilisateur : 12
Heure de début : 2018-01-19 13:08:48
Terminée : Oui
Nombre de points : 10
Heure de fin : 2018-01-19 13:11:48
Durée en secondes : 180
Durée totale : 00:03:00
Distance totale en Km : 1.2017846309768
Dénivelé en m : 90
Dénivelé positif en m : 90
Dénivelé négatif en m : 0
Vitesse moyenne en Km/h : 24.035692619536
Centre du parcours :
- Latitude : 48.2154
- Longitude : -1.549
- Altitude : 0

Id : 21
Utilisateur : 12
Heure de début : 2018-01-19 13:08:48
Terminée : Non
Nombre de points : 5
Durée en secondes : 80
Durée totale : 00:01:20
Distance totale en Km : 0.53413448659631
Dénivelé en m : 40
Dénivelé positif en m : 40
Dénivelé négatif en m : 0
Vitesse moyenne en Km/h : 24.036051896834
Centre du parcours :

```
- Latitude : 48.2129
- Longitude : -1.5515
- Altitude : 0

Id : 20
Utilisateur : 11
Heure de début : 2018-01-19 13:08:48
Terminée : Oui
Nombre de points : 10
Heure de fin : 2018-01-19 13:11:48
Durée en secondes : 180
Durée totale : 00:03:00
Distance totale en Km : 1.2017846309768
Dénivelé en m : 90
Dénivelé positif en m : 90
Dénivelé négatif en m : 0
Vitesse moyenne en Km/h : 24.035692619536
Centre du parcours :
- Latitude : 48.2154
- Longitude : -1.549
- Altitude : 0

.....

Id : 1
Utilisateur : 2
Heure de début : 2018-01-19 13:08:48
Terminée : Non
Nombre de points : 6
Durée en secondes : 17825269
Durée totale : 4951:27:49
Distance totale en Km : 2.1902692187635
Dénivelé en m : 50
Dénivelé positif en m : 40
Dénivelé négatif en m : 50
Vitesse moyenne en Km/h : 0.00044234783708164
Centre du parcours :
- Latitude : 48.20745
- Longitude : -1.5515
- Altitude : 0
```

6-3-11 Méhode getLesTraces

getLesTraces($idUtilisateur)
Rôle : fournit la collection des traces de l'utilisateur **$idUtilisateur**
Paramètres à fournir : **$idUtilisateur** : identifiant de l'utilisateur dont on veut obtenir les traces
Valeur de retour : une collection d'objets **Trace**
Particularité : utiliser la méthode **getLesPointsDeTrace($idTrace)** pour obtenir les points de chaque trace et les ajouter à chaque objet **Trace** qui sera ajouté à la collection

Code PHP du test fonctionnel :

```
// test de la méthode getLesTraces($idUtilisateur) ----------------------------------------------
// modifié par dP le 14/8/2021
echo "<h3>Test de getLesTraces(idUtilisateur) : </h3>";
$lesTraces = $dao->getLesTraces(2);
$nbReponses = sizeof($lesTraces);
echo "<p>Nombre de traces de l'utilisateur 2 : " . $nbReponses . "</p>";
// affichage des traces
foreach ($lesTraces as $uneTrace)
{   echo ($uneTrace->toString());
    echo ('<br>');
}
```

Résultat attendu avec le test :

Test de getLesTraces(idUtilisateur) :

Nombre de traces de l'utilisateur 2 : 2

Id : 2
Utilisateur : 2
Heure de début : 2018-01-19 13:08:48
Terminée : Oui
Nombre de points : 10
Heure de fin : 2018-01-19 13:11:48
Durée en secondes : 180
Durée totale : 00:03:00
Distance totale en Km : 1.2017846309768
Dénivelé en m : 90
Dénivelé positif en m : 90
Dénivelé négatif en m : 0
Vitesse moyenne en Km/h : 24.035692619536
Centre du parcours :
- Latitude : 48.2154
- Longitude : -1.549
- Altitude : 0

Id : 1
Utilisateur : 2
Heure de début : 2018-01-19 13:08:48
Terminée : Non
Nombre de points : 6
Durée en secondes : 17825269
Durée totale : 4951:27:49
Distance totale en Km : 2.1902692187635
Dénivelé en m : 50
Dénivelé positif en m : 40

```
Dénivelé négatif en m : 50
Vitesse moyenne en Km/h : 0.00044234783708164
Centre du parcours :
- Latitude : 48.20745
- Longitude : -1.5515
- Altitude : 0
```

6-3-12 Méhode getLesTracesAutorisees

getLesTracesAutorisees($idUtilisateur)
Rôle : fournit la collection des traces que l'utilisateur **$idUtilisateur** a le droit de consulter
Paramètres à fournir : **$idUtilisateur** : identifiant de l'utilisateur dont on veut obtenir les traces qu'il peut consulter
Valeur de retour : une collection d'objets Trace
Particularité : utiliser la méthode **getLesPointsDeTrace($idTrace)** pour obtenir les points de chaque trace et les ajouter à chaque objet **Trace** qui sera ajouté à la collection

Code PHP du test fonctionnel :

```php
// test de la méthode getLesTracesAutorisees($idUtilisateur) ----------------------------------
// modifié par dP le 14/8/2021
echo "<h3>Test de getLesTracesAutorisees(idUtilisateur) : </h3>";
$lesTraces = $dao->getLesTracesAutorisees(2);
$nbReponses = sizeof($lesTraces);
echo "<p>Nombre de traces autorisées à l'utilisateur 2 : " . $nbReponses . "</p>";
// affichage des traces
foreach ($lesTraces as $uneTrace)
{   echo ($uneTrace->toString());
    echo ('<br>');
}
$lesTraces = $dao->getLesTracesAutorisees(3);
$nbReponses = sizeof($lesTraces);
echo "<p>Nombre de traces autorisées à l'utilisateur 3 : " . $nbReponses . "</p>";
// affichage des traces
foreach ($lesTraces as $uneTrace)
{   echo ($uneTrace->toString());
    echo ('<br>');
}
```

Résultat attendu avec le test : A MODIFIER

```
Test de getLesTracesAutorisees(idUtilisateur) :
Nombre de traces autorisées à l'utilisateur 2 : 2

Nombre de traces autorisées à l'utilisateur 9 : 0

Id : 2
Utilisateur : 2
Heure de début : 2018-01-19 13:08:48
Terminée : Oui
Nombre de points : 10
Heure de fin : 2018-01-19 13:11:48
Durée en secondes : 180
Durée totale : 00:03:00
Distance totale en Km : 1.2017846309768
Dénivelé en m : 90
```

```
Dénivelé positif en m : 90
Dénivelé négatif en m : 0
Vitesse moyenne en Km/h : 24.035692619536
Centre du parcours :
- Latitude : 48.2154
- Longitude : -1.549
- Altitude : 0

Id : 1
Utilisateur : 2
Heure de début : 2018-01-19 12:08:48
Terminée : Non
Nombre de points : 6
Durée en secondes : 122002735
Durée totale : 33889:38:55
Distance totale en Km : 2.1902693899948
Dénivelé en m : 50
Dénivelé positif en m : 40
Dénivelé négatif en m : 50
Vitesse moyenne en Km/h : 6.4629451167477E-5
Centre du parcours :
- Latitude : 48.20745
- Longitude : -1.5515
- Altitude : 0
```

6-3-13 Méthode creerUneTrace

creerUneTrace($uneTrace)
Rôle : enregistre la trace **$uneTrace** dans la table **tracegps_traces** et met à jour l'objet **$uneTrace** avec l'identifiant (auto_increment) attribué par le SGBD
Paramètres à fournir : **$uneTrace** : la trace à enregistrer
Valeur de retour : un booléen **true** si l'enregistrement s'est bien passé **false** sinon
Particularités : - Si la date de fin est nulle (cas d'une trace non terminée), le champ **dateFin** prendra une valeur nulle (**PDO::PARAM_NULL**) ; sinon il prendra une valeur chaine (**PDO::PARAM_STR**). - On n'enregistre pas les points de la trace, même si l'objet **$uneTrace** en contient.

Code PHP du test fonctionnel :

```php
// test de la méthode creerUneTrace ----------------------------------------------------
// modifié par dP le 14/8/2021
echo "<h3>Test de creerUneTrace : </h3>";
$trace1 = new Trace(0, "2017-12-18 14:00:00", "2017-12-18 14:10:00", true, 3);
$ok = $dao->creerUneTrace($trace1);
if ($ok) {
    echo "<p>Trace bien enregistrée !</p>";
    echo $trace1->toString();
}
else {
    echo "<p>Echec lors de l'enregistrement de la trace !</p>";
}
$trace2 = new Trace(0, date('Y-m-d H:i:s', time()), null, false, 3);
$ok = $dao->creerUneTrace($trace2);
if ($ok) {
    echo "<p>Trace bien enregistrée !</p>";
```

```
    echo $trace2->toString();
}
else {
    echo "<p>Echec lors de l'enregistrement de la trace !</p>";
}
```

Résultat attendu avec le test : (cela crée les traces dans la base mais ne peut pas remonter l'id)

Test de creerUneTrace :

Trace bien enregistrée !

Id : 0
Utilisateur : 3
Heure de début : 2017-12-18 14:00:00
Terminée : Oui
Nombre de points : 0

Trace bien enregistrée !

Id : 0
Utilisateur : 3
Heure de début : 2018-08-14 23:33:07
Terminée : Non
Nombre de points : 0

6-3-14 Méthode supprimerUneTrace

supprimerUneTrace($idTrace)
Rôle : supprime la trace d'identifiant **$idTrace** dans la table **tracegps_traces**, ainsi que tous ses points dans la table **tracegps_points**
Paramètres à fournir : **$idTrace** : l'identifiant de la trace à supprimer
Valeur de retour : un booléen **true** si la suppression s'est bien passée **false** sinon

Code PHP du test fonctionnel :

```
// test de la méthode supprimerUneTrace ----------------------------------------------------------
// modifié par dP le 15/8/2021
echo "<h3>Test de supprimerUneTrace : </h3>";
$ok = $dao->supprimerUneTrace(22);
if ($ok) {
    echo "<p>Trace bien supprimée !</p>";
}
else {
    echo "<p>Echec lors de la suppression de la trace !</p>";
}
```

Résultat attendu avec le test :

Test de supprimerUneTrace :

Trace bien supprimée !

6-3-15 Méthode terminerUneTrace

terminerUneTrace($idTrace)
Rôle : enregistre la fin de la trace d'identifiant **$idTrace** dans la table **tracegps_traces** ainsi que la date de fin
Paramètres à fournir : **$idTrace** : l'identifiant de la trace à terminer
Valeur de retour : un booléen **true** si la modification s'est bien passée **false** sinon
Particularités : - Le champ **terminee** doit être mis à 1 - Le champ **dateFin** doit prendre comme valeur la date du dernier point de la trace (si la trace contient des points) ou la date système (si la trace ne contient aucun point)

Code PHP du test fonctionnel :

```php
// test des méthodes creerUnPointDeTrace et terminerUneTrace ------------------------------------
// modifié par dP le 15/8/2021
echo "<h3>Test de terminerUneTrace : </h3>";
// on choisit une trace non terminée
$unIdTrace = 3;
// on l'affiche
$laTrace = $dao->getUneTrace($unIdTrace);
echo "<h4>l'objet laTrace avant l'appel de la méthode terminerUneTrace : </h4>";
echo ($laTrace->toString());
echo ('<br>');
// on la termine
$dao->terminerUneTrace($unIdTrace);
// et on l'affiche à nouveau
$laTrace = $dao->getUneTrace($unIdTrace);
echo "<h4>l'objet laTrace après l'appel de la méthode terminerUneTrace : </h4>";
echo ($laTrace->toString());
echo ('<br>');
```

Résultat attendu avec le test :

Test de terminerUneTrace :

l'objet laTrace avant l'appel de la méthode terminerUneTrace :

Id : 3
Utilisateur : 3
Heure de début : 2018-01-19 13:08:48
Terminée : Non
Nombre de points : 5
Durée en secondes : 80
Durée totale : 00:01:20
Distance totale en Km : 0.53413448659631
Dénivelé en m : 40
Dénivelé positif en m : 40
Dénivelé négatif en m : 0
Vitesse moyenne en Km/h : 24.036051896834
Centre du parcours :
- Latitude : 48.2129
- Longitude : -1.5515
- Altitude : 0

l'objet laTrace après l'appel de la méthode terminerUneTrace :

Id : 3
Utilisateur : 3
Heure de début : 2018-01-19 13:08:48
Terminée : Oui
Nombre de points : 5
Heure de fin : 2018-01-19 13:10:08
Durée en secondes : 80
Durée totale : 00:01:20
Distance totale en Km : 0.53413448659631
Dénivelé en m : 40
Dénivelé positif en m : 40
Dénivelé négatif en m : 0
Vitesse moyenne en Km/h : 24.036051896834
Centre du parcours :
- Latitude : 48.2129
- Longitude : -1.5515
- Altitude : 0