Mémento
Site Dynamique / PHP

Pour faire fonctionner site statique :
- Navigateur
- Éditeur de texte
- Apache/PHP/MySQL (qu'on peut retrouver dans WampServer (uniquement Windows))

Inclure les portions de page :
Au lieu de recommencer une portion sur code sur chaque page, on le crée sur une page différente et sur la principale, on rajoute **<?php include(**«nomdelaportion**.php»); ?>**

Les variables :
- On utilise un **$** à mettre avant la variable
- Elle peut contenir du décimal, comme du char ou encore du booléen
- On l'affiche sur notre page en mettant **echo** devant la variable
- On peut concaténer en mettant un point entre chaque partie

Les conditions :
- Même principe que le C# : utilisation du if/else/elseif/switch
- On peut mettre du HTML entre du code PHP (en fermant les balises avant), le programme comprendra.

Les boucles :
- Même principe que le C# : utilisation du while/for...

Les fonctions :
- On peut utiliser des fonctions sur les variables
- Par exemple : **strlen** qui indique le nombre de caractère dans une phrase, **strshuffle** qui mélange les caractères d'une chaines, ou encore **date('d OU m OU y')** qui donne soit le jour, le mois ou l'année ou l'on se trouve en ce moment même
- On peut créer des fonctions avec **functions NomFonction($variable)**

Les tableaux :
- On peut créer des variables contenant plusieurs données
- Par exemple : **$variable[0] = 12; $variable[1] = 14** etc...
- Autre exemple : **$variable = array(12,14...);**
- Si on veut afficher une données précise : **echo $variable[1];**
- On peut donner informations à chaque données :
  $variable = array('nom' => 'Faulconnier, 'prenom' => 'Bastien' ;
- Et l'afficher avec

```
foreach ($variable as $autrevariable)
{
    echo '<p>' . $variable . '</p>';
{
```

Transmettre des données :
- Nous pouvons modifier l'url d'une page afin d'en mettre des données que nous pourrons

récupérer sur une futur page par cela il suffit de rajouter les informations à la suite du .php et en mettant un **?** et en les séparant chacun par **&**
Nom de ma page

- Pour se resservir de ses données sur notre nouvelle page, nous l'appelons avec un **$_GET['variable']**
  Si nous voulons l'afficher nous aurons donc à faire **echo $_GET('variable')**
- Bien penser à faire cela entre des balises PHP vu au-dessus
- Nous devons protéger ses données afin que l'utilisateur ne mette pas n'importe quoi sur la barre d'URL, pour cela nous allons mettre nos données dans un if et nous allons utiliser **isset($_GET['variable']**
  if (isset($_GET['variable1'] AND $_GET['variable2']
  {
  les lignes à exécuter si c'est bon
  }
  else message d'erreur...
- Si nous voulons se servir d'une boucle répéter alors ne pas hésiter à mettre un for/while dans ce if issef et de créer une nouvelle variable en la déclarant int s'il s'agit d'un nombre.

```
<p>
    <a href="bonjour.php?nom=Nebra&amp;prenom=Mathieu">Dis-moi bonjour</a>
</p>
```

```
<p>
    Bonjour <?php echo $_GET['nom'] . ' ' . $_GET['prenom']; ?>
</p>
```

```
<p>
    Bonjour <?php echo $_GET['nom']; ?> <?php echo $_GET['prenom'] ?>
</p>
```

```
<p>
    <?php
    if (isset($_GET['nom']) AND isset($_GET['prenom']) AND isset($_GET['repeter']))
    {
        $nbRepetitions = (int) $_GET['repeter'];

        if ($nbRepetitions < 100)
        {
            for ($repetition = 0 ; $repetition < $nbRepetitions ; $repetition++)
            {
                echo '<p>Bonjour ' . $_GET['nom'] . ' ' . $_GET['prenom'] . '</p>';
            }
        }

    }
    else
    {
        echo 'Pas de nom ou de prénom défini !';
    }
    ?>
</p>
```

Les formulaires :

- Pour la base en HTML, se référer un PDF « SI6 – Les formulaire en HTML » disponible dans cours SI6 ou numériquement dossier SI6
- Nous pourrons récupérer les données des visiteurs afin de les afficher sur une autres pages.
- Comme vu en HTML, nous pouvons utiliser GET ou POST dans la méthode, et si nous utilisons **$_POST['variable']**, nous pourrons nous en servir
  echo $_POST par exemple, pour l'afficher
- En cas de checkbox dans le type (case à cocher), on réutilise la méthode **isset** pour savoir quel code exécuter
- Pour éviter le faille XSS, c'est à dire le fait d'un utilisateur incruste du code via nos 'textbox', on met entre parenthèse la méthode **htmlspecialchars()**. Comme ça, l'utilisateur ne pourra pas modifier notre code.

```
<form action="cible.php" method="POST">
    <p><label>Prénom : <input type="text" name="prenom" /></label></p>
    <p><input type="submit" /></p>
</form>
```

```
<?php
if (isset($_POST['vegetarien']))
{
    echo '<p>Vous êtes donc végétarien.</p>';
}
else
{
    echo '<p>Vous n\'êtes pas végétarien, vous mangez de la viande.</p>';
}
?>
```

```
<p>Bonjour <?php echo htmlspecialchars($_POST['prenom']); ?></p>
```

Variables superglobales, sessions et cookies

- Les superglobales commencent toujours pas **$_** (GET, POST, SERVER...)
- **$_SERVER** permet d'afficher des contenus d'informations sur les scripts, les chemins ou encore les headers
- Pour utiliser les variable de sessions, mettre au TOUT DEBUT de CHAQUE page.

  <?php
  sessions_start();
  ($_SESSION['variable'] = 'Donnée'); mettre uniquement sur la première
  ?>

- Les variables de sessions permettent de se souvenir de certaines informations pendant toute la durée de la session de l'utilisateur.
- Le cookies, eux, sont des fichiers enregistrés par notre navigateur et qui sont retenus sur notre ordinateur pendant un certaine durée.
- Sa déclaration se fait aussi au début de notre code mais de manière différente

  <?php

setcookie('variable', 'donnée', time() + DuréeEnSecondes);
?>

- Pour se resservir de cette donnée, on utilise **$_COOKIE['variable'];**
- Nous pouvons utiliser un **isset** dans un if si le cookie a été supprimé et éviter d'afficher un erreur sur notre page web
- On utilise une session juste pour le temps d'une visite alors que le cookie est là pour une durée déterminée qui peut durer très longtemps...ou pas. Une session garde plus d'informations qu'un cookie.

```php
<p><?php print_r($_SERVER); ?></p>
```

```php
<?php
session_start();

$_SESSION['nom'] = 'Dupont';
?>
```

```php
<p><?php echo 'Bonjour M. ' . $_SESSION['nom']; ?></p>

<p>Aller aux <a href="informations.php">informations</a></p>
```

```php
<?php
setcookie('nom', 'Bertrand', time() + 3600*24*365);
?>
```

```php
<?php
if (isset($_COOKIE['nom']))
{
    ?>
        <p><?php echo 'Bonjour M. ' . $_COOKIE['nom']; ?></p>
    <?php
}
?>
```

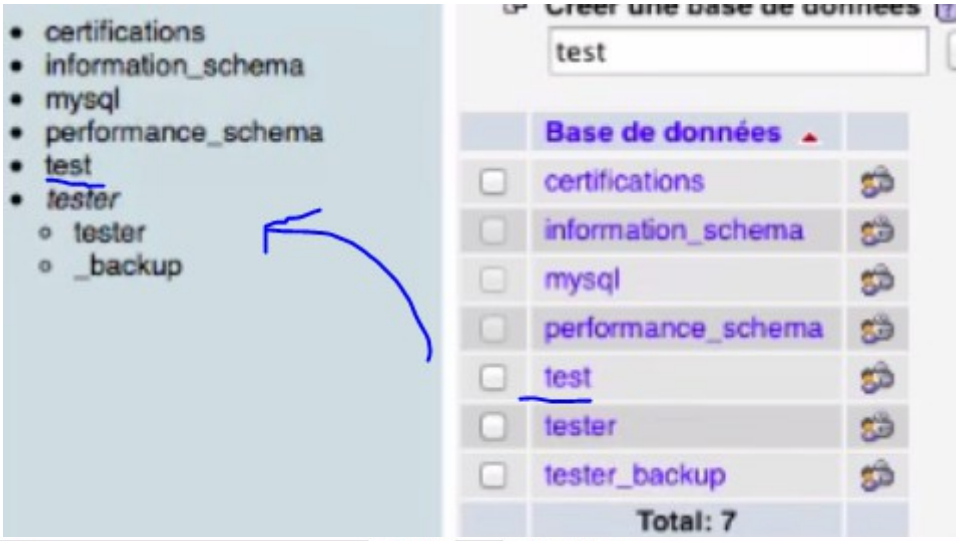
Présentation des bases de données

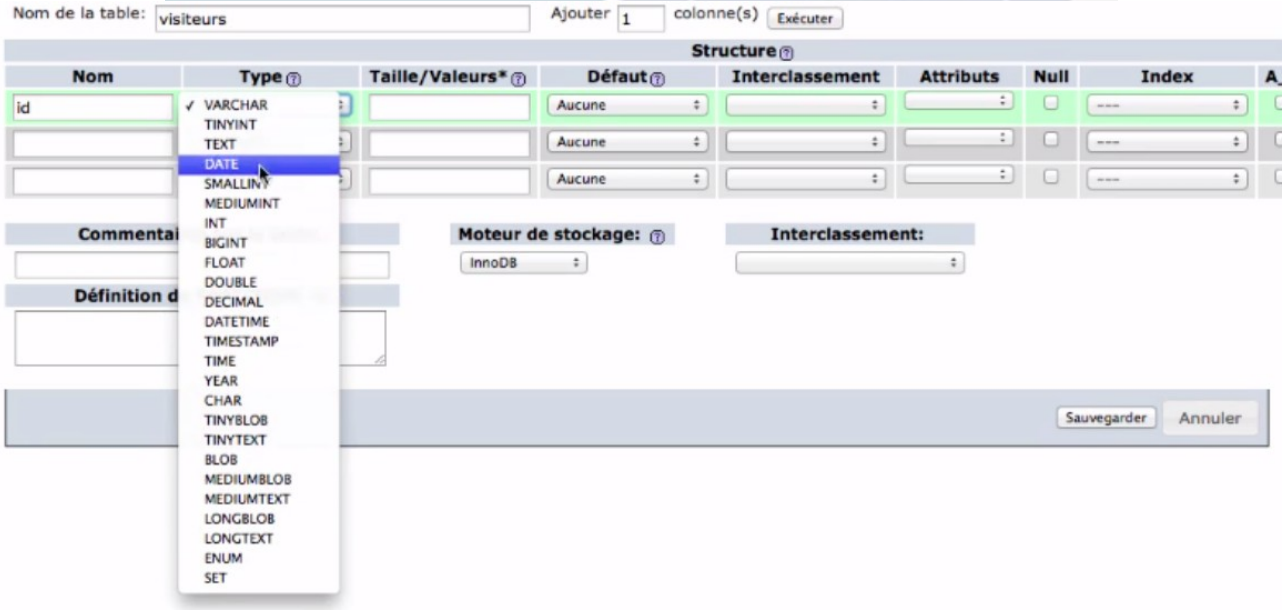
- PHP sert de jonction entre nous et MySQL.
- Explication de comment marche une base de données....

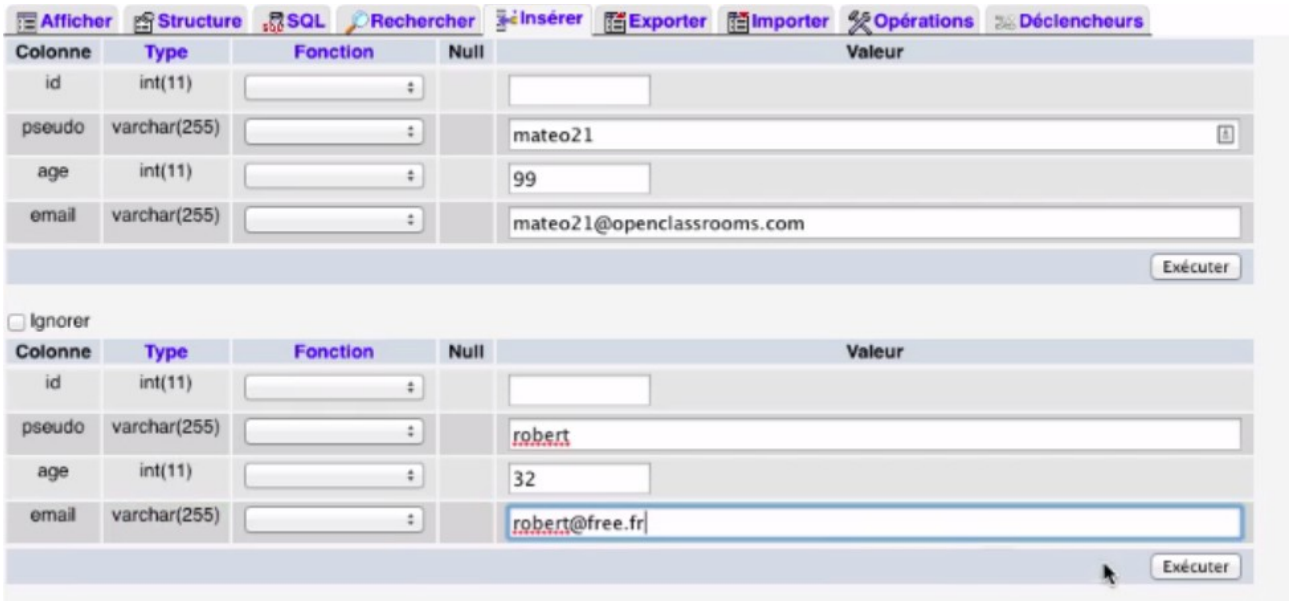
phpMyAdmin

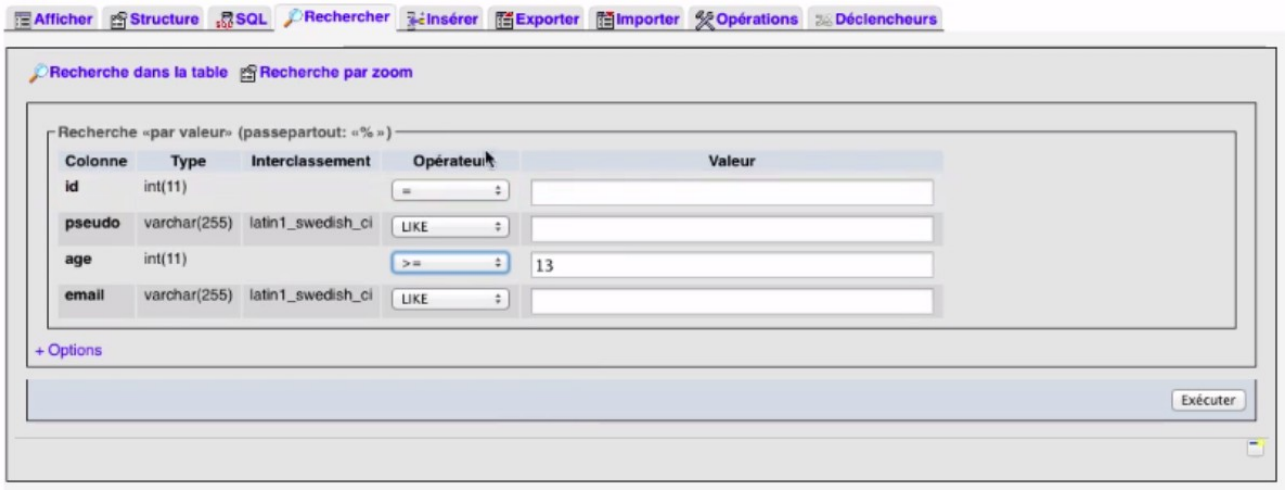
- C'est un logiciel qui permet de communiquer avec MySQL via une interface graphique (installé avec WampServer), On retrouve dedans donc, nos bases de données
- On peut assez simplement créer des BDD ainsi que ces tables
- Les types des tables sont déjà prédéfinis via une liste
- On indique la clé primaire de la classe via Index (liste)
- La case A_I se trouvant à côté, permet de créer un chiffre (+1) à chaque occurrence
- Si on utilise un type VARCHAR, on doit indiquer sa taille car c'est un type qui peut rentrer qu'une certaine taille d'informations (255 max.)

- Nous pouvons exécuter nos actions simplement avec l'interface ou utiliser le langage SQL
- Nous pouvons exporter/importer nos tables dans un fichier .sql avec des données précises









Lire des données

- Pour se servir de MySQL dans une page PHP, on doit la déclarer sur le page avec
  <?php
  $bdd = PDO('mysql:host=NomDuHost;dbname=NomDeLaBDD','Login','mdp');
  ?>
- Pour faire une requête SQL sur PHP nous faisons
  $variable1 = $bdd->query('REQUETE SQL');
- On peut bien sur utiliser les données d'une BBD pour cela on utilise la fonction **fetch** dans une boucle **while**.
  while ($varible2 = $variable1->fetch())
  {
  echo...
  }
- En cas de soucis de nos requête, nous pouvons générer une erreur. Pour cela on rajoute **array(PDO:ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION)** dans la déclaration de notre BDD juste après le mdp.
- Bien faire attention aux erreurs assez récurrente en SQL
- Pour faire des requêtes 'préparées', on utilise pas query mais prépare et on utilise un **?** dans notre requête (par exemple dans le WHERE) et entre guillemet. Ensuite on écrit cette ligne **$variable2->execute(array($_GET/POST['donnée']));**
- Bien utiliser un isset pour vérifier si notre variable existe bien

```php
<?php
$bdd = new PDO('mysql:host=localhost;dbname=test', 'root', '');
?>
```

```php
<?php
$bdd = new PDO('mysql:host=localhost;dbname=test', 'root', 'root');
$reponse = $bdd->query('SELECT * FROM jeux_video');
while ($donnees = $reponse->fetch())
{
    echo '<p>' . $donnees['console'] . ' - ' . $donnees['nom'] . '</p>';
}
?>
```

```
array(PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION));
$requete = $bdd->prepare('SELECT * FROM jeux_video WHERE console = ?');
$requete->execute(array($_GET['console']));
while ($donnees = $requete->fetch())
```

Ecrire des données

- phpMyAdmin propose déjà des requêtes SQL pré remplies
- On peut, bien sur, insérer des informations dans notre BDD via une page PHP avec **INSERT INTO.** Nous pouvons, comme vu avant, le demander à l'utilisateur. Pour cela, on utilise le même principe que vu avant avec dans **VALUES()** autant de **?** Que d'informations demandées.
- Possibilité d'utiliser UPDATE et DELETE aussi

```
$requete = $bdd->prepare('INSERT INTO jeux_video(nom, possesseur) VALUES(?, ?)');
$requete->execute(array($_GET['nom'], $_GET['possesseur']));
```

Les fonctions SQL/dates en SQL

- On peut très bien se servir de tout ce qu'on a vu dans le SI3
- Pour utiliser l'heure actuelle, on met **NOW()**

Les expressions régulières

- **preg_match** permet de faire une recherche avec une expression régulière
  preg_match('#mot#', 'Phrase avec le mot.')
- Les 2 # permettent de délimiter le début et la fin du mot.
- Après le deuxième #, nous pouvons mettre des options (i, par exemple, permet de ne pas faire la différence entre les majuscules et minuscules.
- Nous pouvons mettre plusieurs mots entre les dièses. Ils seront séparés par un | (ou)
- Mettre un dollard avant le dernier # pour dire qu'on veut que ce mot soit uniquement à la fin (dernier caractère)
- Même principe au tout début avec un ^
- En mettant des crochets dans un mot, avec plusieurs lettres dedans, cela nous permet de dire que l'une de ses lettres peut être autorisés. En mettant un tiret entre 2 lettres et dans le crochet, cela veut dire que les lettres existant entre les 2 seront autorisées.
  #phr[a-z]se#
  Emploi de la négation avec le ^ : tous sauf (ne pas confondre avec : au tout début)
  On peut aussi faire cela entre des balises HTML
- En précisant un bout de mot entre parenthése peut être présent un certains nombre de fois :
  ? 0 ou 1 une fois
  + 1 ou plusieurs fois
  * 0 ou plusieurs fois
- On peut préciser le nombre de fois que notre partie entre parenthèse soit présente avec un nombre précis entre crochet. Il faut bien rajouter ^ au début et $ à la fin. Dans le crochet, si on rajoute une virgule suivi d'un autre chiffre, on accepte qu'il soit accepté entre les 2 chiffres
  {1-7} ← il pourra être afficher une ou 7 fois.

```
#(((https?|ftp)://(w{3}\.)?)(?<!www)(\w+-?)*\.([a-z]{2,4}))#
```

```
if (preg_match('#banjo#', 'J\'aime la guitare.'))
{
    echo 'VRAI';
}
else
{
    echo 'FAUX';
}
```

```
g_match('#gr[iao]s#', 'La nuit tous les chats sont gras'))
```

```
(preg_match('#Ay(ay)+#', 'Ay'))
```

```
(preg_match('#Ay(ay){3}#', 'Ayayayayay'))
```

```
(preg_match('#^[0-9]{3,6}$#', '5888000'))
```

- Le point permet de savoir si il y'a bien la présente d'un caractère dans notre chaîne (On peut utiliser les crochet, bien entendu)
- Bien utiliser un antislash si on veut remarquer la présence d'un caractère dans une chaîne (?, ^, $, etc...)
- Pour vérifier une date **#^[0-9]{2}/[0-9]{2}/[0-9]{4}$#**
  ou **([0-9]{2}/){2}...**

- On peut récupérer une portion de notre recherche avec **preg_replace**. Dans notre recherche, nous mettons entre parenthèse ce qui nous intéresse. Nous devons rajouter un second paramètre au milieu. Il contiendra un $ suivi du numéro où est présente la parenthèse dans la phrase On le met dans une variable pour s'en resservir ailleurs.

```
(preg_match('#^[0-9]{2}/[0-9]{2}/[0-9]{4}$#', '05/09/1994'
```

```
$annee = preg_replace('#^([0-9]{2}/){2}([0-9]{4})$#', '$2', '05/09/1994');
echo "An : " . $annee;
```

```
$date = preg_replace('#^([0-9]{2})/([0-9]{2})/([0-9]{4})$#', '$3-$2-$1', '20/01/2005');
echo $date;
```

La programmation orientée objet (différence et nouveauté par rapport à ce que l'on connaît déjà)

- Instanciatio : **$NomObjet = new NomClasse();**
- Pour pouvoir se servir de notre classe sur une page PHP : **include('NomClasse.php');**
  On peut utiliser **include_once** à la place pour que notre classe soit déclarée dans tout notre programme
- Déclaration des méthodes : mettre $ avant le nom. Penser à la visibilité.

- Accesseur :
  public function getNomMethode()
  {
  return $this->NomMethode;
  }
- Mutateur :
  public function setNomMethode($NomMethode)
  {
  $this->NomMethode = $NomMethode;
  }
- Constructeur :
  public function __construct($NomMethode, $NomMethode2) Les 2 tirets sont importants
  {
  $this->NomMethode = $NomMethode;
  $this->NomMethode2 = $NomMethode2;
  ...
  }

```php
class Membre
{
    private $pseudo;
    private $email;
    private $actif;

    public function __construct($pseudo)
    {
        $this->pseudo = $pseudo;
        $this->actif = true;
    }

    public function getPseudo()
    {
        return $this->pseudo;
    }

    public function setPseudo($pseudo)
    {
        $this->pseudo = $pseudo;
    }

    public function bannir()
    {
        $this->actif = false;
    }

    public function debannir()
    {
        $this->actif = true;
    }
}
```

```php
include_once('Membre.class.php');

$robert = new Membre('Robert');

echo $robert->getPseudo();

$robert->setPseudo('Gerard');

echo $robert->getPseudo();

$robert->bannir();


$robert->debannir();
```

Résultat quizz

1)

Test

Votre travail a bien été enregistré. Votre score : **9 / 10**

2)

Test

Votre travail a bien été enregistré. Votre score : **9 / 10**

3)

Test

Votre travail a bien été enregistré. Votre score : **10 / 10**

4)

Test

Votre travail a bien été enregistré. Votre score : **8 / 10**

OPENCLASSROOMS

BRIGHT TEACHERS, GREAT CLASSMATES.

Vous avez réussi le cours "Concevez votre site web avec PHP et MySQL" !

Bonjour bastienfaulconnier,

Bonne nouvelle ! Vous venez de réussir avec succès le cours "Concevez votre site web avec PHP et MySQL" avec un score de **80%**.

Saviez-vous que vous pouvez obtenir votre certification de réussite du cours avec un compte Premium ?

Vous pouvez devenir Premium dès à présent pour obtenir votre certification, mais aussi accéder à tout le catalogue d'eBooks d'OpenClassrooms et à tous les autres cours certifiants du site !

Pour obtenir votre certification : une fois votre compte Premium activé, envoyez-nous votre identifiant OpenClassrooms à certifications@openclassrooms.com. Nous vous répondrons en vous transmettant votre certification !

Merci et à bientôt pour de nouveaux cours !

L'équipe d'OpenClassrooms.