Notice d'utilisation et de modification du repo

I. Prérequis

Le bon fonctionnement du site nécessite certaines installations couronnées de succès.

Serveur local

Il faut un serveur local fonctionnel.

On peut utiliser :

- Xampp (macOS, Windows) ;
- WAMP (Windows) ;
- MAMP (macOS, Windows).

L'installation doit permettre de disposer d'un serveur Apache fonctionnel ainsi que d'un serveur de base de données avec une interface phpMyAdmin.

Dossier d'exécution PHP

Selon qu'on utilise WAMP ou autre chose le dossier peut changer.

Pour WAMP (Windows 64 bits) : "C:\wamp64\www"

Sinon, il s'agit du dossier « htdocs » qu'on trouve dans le répertoire d'installation de Xampp ou de MAMP.

Authentification à la base de données

Si WAMP est installé, le mot de passe par défaut pour la base de données est « root », sinon il s'agit de la chaîne vide (aucun mot de passe, i.e. « »).

Quelle que soit la solution installée, le login est « root ».

Installation de Git (la ligne de commande)

C'est plus que conseillé.

Sur Windows : https://git-scm.com/downloads

Sur macOS, via le terminal : https://youtu.be/sJ4zr0a4GAs

Après cette étape, il faut se connecter avec son compte GitHub si on souhaite faire communiquer Git et GitHub. Pas de souci : si besoin est, popup de connexion sera !

Il faut aussi connaître les commandes de base : https://youtu.be/USjZcfj8yxE

Pour résumer, être familier des commandes : « add », « commit », « push », « fetch », « pull », « branch », « checkout » et « status » (https://git-scm.com/docs, je comprendrais que ce soit le seul lien que vous n'ouvriez pas).

II. Télécharger le dossier depuis GitHub

On va utiliser Git pour le faire. Une installation propre comme décrite ci-dessus est donc conseillé.

Naviguer en utilisant l'explorateur de fichiers (Finder sur macOS) jusqu'au dossier de PHP (htdocs ou www).

Vérifier l'absence du dossier SoundInShape à cet endroit. Oui ! On souhaite le télécharger, ce qui sous-entend qu'on ne l'a pas déjà sur sa machine… Pour mettre à jour le code du dossier existant, le mieux est de se renseigner sur les commandes « fetch » et « pull » (en spécifiant le nom de la branche de laquelle on souhaite copier les fichier).

Ouvrir un terminal à cet endroit :

- Sur Windows : dans l'espace blanc de la barre d'accès en haut de l'explorateur, écrire « cmd » puis appuyer sur « entrée » ;
- Sur macOS : clic droit sur le dossier htdocs, et sélectionner « ouvrir dans le terminal » (ou analogue).

Exécuter la commande « git clone -b main https://github.com/LIFRANCKY/SoundInShape ».

« -b main » signifie qu'on télécharge la branche « main ». C'est facultatif puisque « main » est la branche par défaut. Cependant on aurait pu faire : « -b Alex » pour récupérer la branche d'Alex…

IMPORTANT : ce n'est pas ainsi qu'on passe d'une branche à une autre branche ! C'est uniquement pour télécharger une branche (une fois). Si on veut passer d'une branche à une autre existante on utilise « git checkout <new-branch> ». Re-regarder la vidéo si ce détail n'est pas clair.

III. Configurer la base de données

Il faut passer par phpMyAdmin pour supprimer la base de données « app » si elle existe déjà.

Ensuite, on la recrée :

- Par l'interface « Nouvelle base de données » ;
- Par le code SQL : « CREATE DATABASE app ».

Il n'y a plus qu'à importer le fichier « app.sql » présent dans le dossier « model ».

Connaissez-vous le mot de passe de la base de données ? (« root » ou « »).

Il faut vérifier dans le fichier « config.php » que la valeur DB_PASS est paramétrée sur la bonne valeur (« root » ou « »).

Ce fichier est utilisé pour se connecter à la base de données dans le fichier « model/db-connection.php ».

$PDO = new *PDO*('mysql:host='.DB_HOST.';dbname='.DB_NAME, DB_USER, DB_PASS);

Les valeurs sont :

- DB_HOST : “localhost:3306” ;
- DB_NAME : “app” ;
- DB_USER : “root” ;
- DB_PASS : c’est vous qui avez écrit cette valeur. Par défaut : « root ».

IV. Déploiement du site Internet

Il suffit de se rendre à l’adresse : http://localhost/soundinshape. Le serveur mail ne sera pas bien configuré. Si vous voulez le configurer correctement et que vous avez un mac, bonne chance. Sinon, regarder les ressources sur Moodle pour configurer sendmail avec WampServer (à noter que la configuration d’une adresse Gmail est rendue difficile depuis la mise à jour de 2022 des services tiers de la messagerie Gmail).

V. Changement de l’URL

Comme Franck l’a remarqué, on peut facilement changer l’URL du site en modifiant trois valeurs.

Imaginons qu’on place le site dans un dossier newsoundinshape (qui se situe dans www ou htdocs, cela va sans dire !).

On disposera de la nouvelle URL : http://localhost/newsoundinshape

Pour que cette nouvelle URL fonctionne, il est nécessaire de modifier :

- Dans config.php

DOMAIN_NAME = “http://localhost/newsoundinshape/”

BASE_NAME = “/newsoundinshape/”

- .htaccess

RewriteBase /newsoundinshape/

VI. Explication du code

Le code suit une architecture MVC sophistiquée qu'il faut intégrer (c'est normal de prendre plus d'une heure). ***Une bonne connaissance en MVC vous aidera énormément au second semestre quand on utilisera Java.***

« index.php » : ce fichier récupère l'URL saisie par l'utilisateur ainsi que la méthode d'accès HTTP (POST ou GET).

https://developer.mozilla.org/fr/docs/Web/HTTP/Methods

On récupère les routes configurées dans les controllers grâce au fichier « Router.php » et notamment de la méthode « generateRoutes() ».

Le reste du fichier index.php appelle le controller à exécuter.

Architecture MVC

Le principe est simple, et terriblement puissant. L'architecture MVC du projet est extrêmement puissante (extraordinairement plus que celle des TPs MVC de notre cher tuteur !). Mais la courbe d'apprentissage est un peu plus relevée.

Controller

Il s'agit d'une classe présente dans le dossier « controller ». Une classe dispose de plusieurs méthodes (ou fonctions) chacune permettant de réaliser une tâche.

https://fr.wikipedia.org/wiki/Programmation_orient%C3%A9e_objet

Les méthodes des controllers ont deux objectifs :

- Inclure la bonne vue en transmettant les variables de la méthode au fichier de vue ;
- Ou envoyer une réponse JSON (surtout quand la méthode HTTP est POST, après l'envoi d'un formulaire).

L'exemple le plus simple de controller est présent dans le fichier « StaticController.php » du dossier « controller ». Observez de près la méthode « showCGU() » et envoyez « Aïe <3 CGU » sur le groupe What's App.

Allons un peu plus dans le détail : le « UserController », avec la méthode « display_register_user_form() » :

```php
#[Route('register-user')]
public function display_register_user_form()
{
    if (!is_the_session_started()) {
        require_once 'view/user/register-user.php';
    } else {
        header('Location: ' . DOMAIN_NAME);
        return;
    }
}
```

Ce code s'exécute lorsqu'on appelle l'URL « [nom du site]/register-user ». Par exemple, ce code vérifie que l'utilisateur n'est pas connecté avec la fonction is_the_session_started(). Si l'utilisateur est connecté, on préfère le rediriger vers la racine du site Internet.

Autre exemple de méthode : celle qui est appelée par POST lorsqu'on envoie un formulaire (l'attribut « action » de la balise « form » correspond à la route qui décore la méthode dans le controller.

```php
#[Route('post-register-user')]
#[Method('POST')]
public function post_register_user()
{
    header('Content-Type: application/json');

    require_once 'controller/functions/validateFormData.php';
    list($form_data, $error_message) = validateFormData($_POST, [···
    ]);

    if ($error_message !== '') {
        SendJSONResponse('error', $error_message);
        return false;
    }

    if ($form_data['password1'] !== $form_data['password2']) {
        SendJSONResponse('error', 'Passwords do not match.');
        return false;
    }

    require_once 'model/db-user.php';
    if (get_id_by_mail($form_data['mail'])) {
        SendJSONResponse('error', 'Mail already used.');
        return false;
    }
```

Voilà le début de la méthode « post_register_user() ». Il s'agit d'un code logique qui vérifie l'intégralité des informations contenues dans la superglobale « $_POST » avec la fonction « validateFormData() ». Si le message d'erreur est vide, aucune erreur n'a été relevée et on continue l'exécution du code.

On vérifie ensuite que les mots de passe sont identiques (il a été correctement réécrit par l'utilisateur lors de la création de son compte) et que l'adresse mail n'est pas déjà enregistrée dans la base de données.

Si on arrive à la fin de la fonction, cela veut dire qu'aucune erreur n'a été relevée précédemment. On écrit le code JSON de succès ainsi que la valeur « true » (au lieu de « false »).

View

Il s'agit de fichiers PHP qui contiennent majoritairement du code HTML. Regarder le fichier « CGU.php » et envoyez le message « I Want My Morning Chocolate So Badly » sur le groupe What's App.

Un fichier view est très simple. En voici un avec un formulaire (que j'ai replié pour gagner de l'espace).

```php
<?php
$PAGE_TITLE = 'Connectez-vous';
addCSS('login.css');
require_once 'view/header.php';
?>

<!-- Submission via AJAX (POST) -->
<div class="form-container">
    <h2>Se connecter</h2>

    <form action="<?= DOMAIN_NAME ?>post-login">···
    </form>
</div>

<div id="feedback"></div>

<script src="<?= DOMAIN_NAME ?>view/js/send-forms-ajax.js"></script>

<?php
require_once 'view/footer.php';
?>
```

La balise <title> se trouve dans le fichier header.php et contient la chaîne de caractère « $PAGE_TITLE ». Le fichier CSS « login.css » sera inclus dans une balise <link> comme d'ordinaire. A noter que les tous les fichiers CSS se trouvent dans « view/css/ ».

Remarquez les « require_once » qui permettent d'inclure le header et le footer. Pas besoin de les créer deux fois !

Lorsqu'on envoie le formulaire (<input type=«submit» />), c'est la route « post-login » qui sera appelée (le bon controller sera automatiquement détecté). Les données du formulaire seront envoyées à la méthode décorée par #[Route('post-login')] et #[Method('POST')]. C'est bien entendu dans le UserController que se trouve cette méthode (fonctionnalité de base d'un utilisateur).

« <?= DOMAIN_NAME ?> » sera remplacé par le contenu de la constante (https://www.php.net/manual/en/language.constants.php) DOMAIN_NAME, comme par exemple : http://localhost/soundinshape/. L'expression sera donc évaluée à http://localhost/soundinshape/post-login. Fantastique !

Même syntaxe pour ajouter le fichier Javascript à la fin. Ce fichier permet d'envoyer n'importe quel formulaire présent dans le code HTML par Ajax (présentation du tuteur à venir). Tous les fichiers JS se trouvent dans « view/js/ ».

https://www.w3schools.com/js/js_ajax_intro.asp

Model

C'est très simple ! Toutes les fonctions se trouvent dans le dossier « model », regroupées par catégories (« user », « forum », « statistics », « connection », etc.). Chaque fichier présent dans « model » se présente sous la forme d'une liste de fonctions qui contiennent une requête SQL. Il suffit juste de lire le titre de la fonction pour savoir ce quelle fait. Exemple :

```php
<?php

global $PDO;
require_once 'model/db-connection.php';

function get_all_topics()
{
    global $PDO;
    return $PDO->query('SELECT * FROM forum_topic')->fetchAll();
}

function get_topic($topic_id)
{
    global $PDO;
    $query = $PDO->prepare('SELECT * FROM forum_topic WHERE forum_topic_id = :topic_id');
    $query->execute(['topic_id' => $topic_id]);
    return $query->fetch();
}

function topic_exists($topic_id)
{...
}
```

A part que le fait de rédiger ces requêtes nécessite de bien connaître la structure de la base de données, c'est très simple.

Il faut cependant noter l'usage du mot clé « global » présent au début de fichier et au début de chaque fonction. La variable $PDO contient l'interface permettant de communiquer avec la base de données. Elle est définie dans l'espace de nom global.

https://www.php.net/manual/en/language.variables.scope.php

Pour pouvoir utiliser $PDO dans les fonctions, il faut utiliser global à ces deux endroits !

VII. Entrainement

Créer un controller « HibouController » qui dispose d'une méthode « AfficherFormulaire() » décorée avec la route « hibou-formulaire ».

Créer la view associé « HibouForm.php ». Cette view comprendra un $PAGE_TITLE, **le** header (header.php), un formulaire et un **le** footer (footer.php). Attention à ne pas oublier la ligne qui inclut le fichier JS juste avant le footer. `<script src="view/js/send-forms-ajax.js"></script>`

Dans le form, on trouvera la balise :

```
<input type="text" name="nom-hibou" placeholder="nom du hibou"/>
```

Après le form, rajouter une balise vide : `<div id="feedback"></div>`. Cette balise contiendra les informations renvoyées par le serveur à l'issu de la requête POST.

Créer une seconde méthode dans « HibouController » qui répondra pour la route « post-hibou-formulaire ». Ne pas oublier le décorateur #[Method('POST')].

Voici le code à insérer dans cette seconde méthode :

```
$postDataString = '';
foreach ($_POST as $key => $value) {
    $postDataString .= $key . ': ' . $value . "\n";
}

echo json_encode([
    'status' => 'error',
    'message' => $postDataString,
    'redirect_to' => ''
]);
return true;
```

Que fait ce code ? Etapes :

- Comprendre que « $_POST » est un tableau associatif clé-valeur ;
- Comprendre que le point (.) est l'opérateur de concaténation des chaînes de caractères ;
- Savoir ce qu'est le type de fichier « JSON » ;
- Comprendre ce que permet de faire « json_encode() » ;
- Comprendre pourquoi on affiche (« echo ») le résultat de « json_encode ».

Envoyer le message « Trop simple ! » sur le groupe What's App quand vous aurez compris.

VIII. BONUS

- Créer une table hibou dans la base de données (id, nom) ;
- Créer un fichier « db-hibou.php » dans le dossier « model » ;
- Ajouter (« require_once ») le code de connexion à la base de données ;
- Ajouter une fonction « ajouter_hibou($nom) » dans ce fichier.
- Modifier la méthode « POST » du controller pour ajouter le nom du hibou dans la base de données ;
- Envoyer « I killed it so easily » sur le groupe What's App.

IX. Astuces

Savez-vous que vous pouvez activer le statut « PRO » sur votre compte Google en transmettant à GitHub que vous bénéficiez d'une adresse mail ISEP ?

https://education.github.com/students

Cet enregistrement va vous permette d'installer le plugin GitHub Copilot dans VS Code. Hein ? Il s'agit de l'outil de suggestion de code le plus avancé au monde qui utilise GPT-4 et entrainé sur **l'ensemble des codes sources présents sur GitHub**. Attention, il faut savoir appuyer sur « Tab » quand la suggestion apparaît sur l'écran…

https://marketplace.visualstudio.com/items?itemName=GitHub.copilot

Fin du document 😊