**Évaluation : projet Microblog**

**1 - Cahier des charges**

Microblog est une application permettant de partager des petits textes (messages, réflexions) de 140 caractères au maximum.

Chaque utilisateur de l'application (qui est enregistré dans la base) est abonné à zéro ou plusieurs autres utilisateurs de la base. Chaque utilisateur est auteur de zéro ou plusieurs messages horodatés.

**1.1 - Fonctionnalités attendues**

Si l'utilisateur n'est pas authentifié, il peut :

* se connecter.

Une fois authentifié avec succès (via un login et un mot de passe, reconnu par l'application), l'utilisateur peut :

* lire les messages dont il est l'auteur,
* lire les messages dont les utilisateurs auxquels il est abonné sont auteur,
* ajouter un nouveau message,
* se déconnecter.

Fonctionnalités **non** demandées :

* gestion des utilisateurs (création, modification, suppression),
* gestion des abonnements (ajouts, retraits),
* gestion des messages (modification, suppression).

**1.2 - Règles supplémentaires**

Sécurité :

* L'utilisateur non authentifié ne doit pas pouvoir voir un message.
* Les messages des utilisateurs auxquels l'utilisateur n'est pas abonné ne doivent pouvoir être vus par l'utilisateur.
* Toute erreur que commet l'utilisateur doit être signalée à l'utilisateur.

Concernant les nouveaux messages :

* Un nouveau message saisi ne doit pas être enregistré s'il fait plus de 140 caractères.
* Un message ne doit pas contenir de HTML, ni de SQL

Charte graphique : laissée à l'appréciation du développeur ; mais doit rester néanmoins utilisable de manière conventionnelle.

La présentation des données :

* La liste des messages devrait être triée et/ou filtrée, mais le tri et le filtre sont laissés à l'appréciation du développeur.
* Le formulaire de saisie d'un nouveau message est également laissé à l'appréciation du développeur.
* Le formulaire d'authentification ne devra pas laisser apparaître le mot de passe en clair.

**2 - Livrables**

Le code du projet devra être fourni complet : PHP, fichiers de configuration, styles CSS, templates HTML, images, etc...

La base de données (et ses données) devra être exportée (dumpée) dans un script SQL de telle sorte qu'on peut reconstruire la base depuis zéro : dans phpmyadmin, il y a une fonction d'export qu'il faut utiliser pour produire un fichier SQL avec structure et données de la base en SQL standard.

Avec le code PHP et la base SQL, seront également livrés (dans des fichiers séparés) :

* Une documentation d'exploitation : un document de quelques lignes détaillant :
  + Les spécificités de votre projet : s'il y a des bibliothèques supplémentaires à installer, ou des opérations non triviales à effectuer avant l'utilisation. Les spécificités sont les opérations autres que : copier le code dans un répertoire, configurer le serveur httpd (apache) pour servir le projet, et lancement du script d'import de la base de données.
  + Les comptes à utiliser pour les tests : login/mdp des utilisateurs que vous avez créés dans la base de votre application.
  + Les éléments de configuration si nécessaire : comment utiliser les fichiers de configuration, le cas échéant.
* Une documentation succincte d'utilisation :
  + L'URL de l'application
  + Comment se connecter à l'application
  + Comment ajouter un nouveau message
  + Autres fonctionnalités si développées

Les documentations seront livrées en format ouvert (Openoffice, Libreoffice, PDF ou fichier texte simple).

Le tout sera livré dans une archive compressée standard (gz, bz2 ou zip) avec votre nom et prénom dans le nom du fichier.

La remise de l'archive se fera via Updago, au plus tard à la date précisée ultérieurement.

**3 - Notation**

Par ordre d'importance décroissante (les coefficients ne sont pas fixés encore) :

**3.1 - Technique d'écriture**

un code lisible permet, entre autre, une lecture facilitée, et permet de déboguer efficacement.

* Nom pertinent pour les variables, fonctions, classes, objets, constantes : ils doivent permettre de comprendre rapidement à quels objets on a affaire.
* Code correctement indenté : on doit pouvoir identifier clairement les blocs de code imbriqués les uns dans les autres.
* Commentaire structuré (doxygen [1], par exemple) pour les classes, les variables importantes, les fonctions/méthodes (ainsi que leurs paramètres et valeur de retour), les constantes : ils permettent de comprendre l'utilisation des objets, et l'étendue de leur utilisation ; ces commentaires sont destinés aux développeurs (y compris vous-même).
* Commentaire classique (// commentaire) pour le code non trivial : il doit permettre de comprendre ce que fait le code lorsqu'il est complexe ou moins complexe.
* Les points précédents doivent être respectés tout au long du code d'un même projet.

**3.2 - Technique de programmation**

* Séparations des 3 couches : stockage de données, présentation des données, traitement métier,  ; MVC, DAO-PO-BO
  + Model, DAO (Data Abstraction Objects) : cette couche gère le stockage des données (lecture et écriture depuis la base de données) ; c'est ici que se trouve le SQL ; ici, il ne doit pas y avoir de HTML ou de traitement métier.
  + View, PO (Presentation Objects) : cette couche s'occupe de l'interface homme-machine, présente les données, récolte les saisies de l'utilisateur, et les met à disposition de la couche métier. C'est ici que réside le HTML ; et il ne doit pas y avoir de SQL ni de traitement métier.
  + Controler, BO (Business Objects) : la couche métier est « l'intelligence » de l'application, traite les données saisies et les données stockées, et les fournit aux couches DAO pour stockage et/ou PO pour l'utilisateur. Ici, il ne doit pas y avoir de SQL ni de HTML.
* Programmation objet :
  + Des classes bien choisies : chaque entité du problème est représentée par une classe.
  + L'héritage utilisé à bon escient : lorsqu'un objet B (admettons B = chien) hérite de A (admettons A = animal), on peut dire B est un A (chien est un animal) et ça ne choque personne.
* Utilisation des fonctions et classes proposées par PHP : ne pas hésiter à utiliser des fonctions, des exemples, des constructions de la documentation PHP [2]

**3.3 - Fonctionnalités et fonctionnement de l'application**

Contrairement à ce que vous rencontrerez dans le monde professionnel, ici, le fonctionnement et les fonctionnalités ont une importance secondaire (mais rapportent des points si c'est bien exécuté)

* respect du cahier des charges (modulo une certaine latitude dans l'interprétation)
* les fonctionnalités doivent être présentes ; les fonctionnalités manquantes doivent être précisées lors de la livraison du projet
* les bogues détectés mais non corrigés doivent être précisés lors de la livraison du projet

**3.4 - Conformité des livrables**

Tout les documents livrés ainsi que décrits dans la section « 2 - Livrables ».

[1] La documentation de Doxygen concernant : http://www.stack.nl/~dimitri/doxygen/manual/docblocks.html (utiliser doxygen n'est pas une obligation, mais recommandée)

[2] La documentation PHP habituelle et officielle : http://www.php.net/manual/fr/ ; évidemment, on peut utiliser d'autres sources de documentation, mais attention à la fiabilité !

* [Microblog\_evaluation-phpDevoir](https://updago.univ-poitiers.fr/mod/assign/view.php?id=596744)

Veuillez déposer l'archive compressée de votre projet Microblog, ici. Ce livrable devra être conforme à ce qui est décrit ci-avant.

Veuillez également noter que les devoirs remis après le [à déterminer] sans raison valable, auront des pénalités de retard à raison de 0,5 par jour de retard.

L'évaluation aura lieu une fois que tous les devoirs seront rendus.

Bon courage, et bonnes fêtes de fin d'année.