TP WEB-SERVEUR

BERTRAND Anthony, BANDERA Quentin

Table des matières

1 Introduction

Lors de nos études en L3 Informatique, nous avons dû créer un mini-blog en utilisant les technologies web de notre choix.

2 Technologies utilisées

2.1 Les langages

Nous avons décidé d'utiliser php car il a la possibilité d'être utilisé mélangé à du html, permettant de changer la vue suivant certain paramètres. Mais il peut également servir pour les échanges avec le serveur en tant que langage orienté objet.
Pour la vue, nous avons utilisé html/css. Nous voulions quelque chose de simple que nous connaissions afin de ne pas passer trop de temps sur cet aspect. Le but du projet étant de nous évaluer sur nos compétences en web coté serveur.
Pour les requêtes en base de donnée, nous utilisons SQL.

3 Démarrer le programme

Nous avons travaillé avec le logiciel Wampserver. Pour démarrer notre projet, téléchargez WampServer, déposez le projet dans le dossier "www" de Wampserver, démarrez Wampserver, tapez localhost dans votre navigateur. Vous arrivez sur l'accueil de Wamp. Cliquez sur "créer une virtualhost" et renseigné le chemin absolue du projet. Il ne reste plus qu'à redémarrer wamp (clique gauche sur l'icone puis "rafraichir") et de revenir sur l'accueil de Wamp. Un lien pour le projet sera présent.

4 Problèmes rencontrés

Le problème principal renconntré durant ce projet était le manque d'expériences et de connaissance dans le domaine, ce qui est normal. Ce projet était justement là pour nous donner cette expérience. Il a fallu apprendre sur le tas en se percutant à plusieurs obstacles. Nous avons commis une erreur dès le début de notre projet, mais nous l'avons découverte bien trop tard : Nous avons voulu gérer toutes les actions de l'utilisateur avec notre variable "action" et avec des formulaires. La variables "action" nous a permis d'avoir de la cohérence dans notre code, ainsi qu'une certaine routine dans l'ajout de fonctionnalités. Lorsque l'utilisateur clique sur un bouton, la variable $action change et suivant sa valeur, le bon controleur est appelé pour traiter la demande. La suite est gérée par notre fichier simplemodele.php.

Nous trouvions cette approche très pratique de notre cotés. Mais vers la fin du projet, nous nous sommes rendus compte des problèmes engendrés par cette

méthode. Tout d'abord, l'utilisation abusive de formulaire en html (notamment des boutons submit) fait que lorsque l'utilisateur souhaite revenir en arrière en utilisant la touche "précédent" de son navigateur, ce dernier lui demandait de confirmer l'envoi du formulaire. Cette confirmation est très frustrante en tant qu'utilisateur. L'idée aurait été de garder les formulaires pour la connexion/l'inscription puis de gérer le reste sans. Mais notre variable $action ne peut exister que dans un formulaire (une balise input de type hidden pour changer la valeur de cette variable).
Le deuxième problème engendré par cette méthode est que la variable $action garde sa valeur même après le traitement de la requête. Pour faire simple : Quand un membre envoie un commentaire en cliquant sur "envoyer", la variable $action prend la valeur "ajoutComm" ce qui permet au controleur CtrlMembre d'enregistrer le commentaire en base de donnée avant de rafraichir la page. Si l'utilisateur rafraichit la page manuellement, tout le traitement expliqué ci-dessus est ré-executé, dupliquant son commentaire en base de donnée.

5 Conclusion