

Projet "Bob-Project"

LAFON Sylvain, MAINGRET François et LEVASSEUR Thomas

25 novembre 2011

Table des matières

I	Introduction	2
II	Bob-Project	3
1	Objectifs	3
2	Logique et Organisation	5
2.1	Fichiers	5
2.1.1	header.php	5
2.1.2	index.php	5
2.1.3	les fichiers de /classes et de /Smarty	5
2.1.4	les fichiers de /templates	5
2.1.5	les fichiers de /css	5
2.1.6	En somme	6
2.2	Catégories	7
2.3	Pages	8
3	Architecture des pages	9
3.1	Un modèle	9
3.2	Le résultat	9
4	Technologies utilisées	9
4.1	Support du code	9
4.2	Les langages de programmation	10
4.3	Les bibliothèques utilisées	10
5	Base de données	10
6	Les sessions	12

7	Portabilité	12
7.1	Différents navigateurs	12
7.2	Javascript	12
8	Flexibilité	13
8.1	Changement du design du site	13
8.2	Type de matériel	13
8.3	Changement du contenu	13
III	Manuel	14
8.4	Installation des fichiers	14
8.5	Installation du serveur	14
8.6	Installation de la base de données	14
8.6.1	La base de données	14
8.6.2	PhpMyAdmin	14
8.6.3	Configurer le site	15
8.7	Deplacement des fichiers	15
IV	Conclusion	16

Première partie

Introduction

L'objectif de ce projet était de réaliser un site web pour un magasin de bricolage, Brico-Bob. Ce site devra gérer plusieurs aspects : une partie privée et une partie publique :

- La partie publique sera accessible à n'importe qui, et comprendra, entre autres, la possibilité de visionner des produits, d'en rechercher et d'en commenter.
- La partie privée sera réservée aux administrateurs du site et comprendra un panneau permettant de gérer tout le contenu du site, des produits, images, catégories...

Deuxième partie

Bob-Project

1 Objectifs

L'objectif de notre site est de proposer à la vente, comme à la location, des articles de bricolage pour le grand public. Le site devra donc être assez ergonomique et attrayant visuellement afin de ne pas repousser les visiteurs.

Voilà les points importants sur lesquels nous avons fait attention :

- Un temps de chargement des pages optimisé. En effet on estime le seuil d'acceptation de chargement d'un site à deux secondes (selon plus de 47% des internautes sondés). Au delà, vous commencez à perdre des visiteurs. (source : Marketing Management par P. Kotler).
- Proposer une description des produits assez détaillée : si le client ne trouve pas assez d'informations sur un produit de notre site, il ira chercher ces informations chez un concurrent, et à moins que notre prix soit significativement plus bas, il y a de grandes chances qu'il achète chez ce concurrent.
- Il faudra utiliser le plus de termes simples à comprendre par l'utilisateur plutôt que des termes techniques barbares. Notre site permet donc d'écrire de longues descriptions, mais cette responsabilité incombe plutôt à la société Brico-Bob.



- Donner des informations sur la société et sur le service client, ainsi que sur les administrateurs du site. Le client veut savoir à qui il a affaire et sera plus enclin à acheter si il sait qu'il peut contacter quelqu'un ensuite. Le client sera mis en confiance, ce qui est vraiment important lorsque son argent est en jeu.

Ces informations seront donc accessibles depuis n'importe quelle page du site grâce au menu :



- Ne pas trop s'introduire dans la vie privée du client : nous n'avons demandé au client que des informations non personnelles lors de son inscription : son pseudo et un mot de passe. Il n'a pas besoin de donner son nom ni son adresse tant qu'il ne valide pas une commande. Cela lui permettra de ne pas se sentir "tracké" et de visiter librement le site.
- Un moteur de recherche bien réalisé. Si le client sait exactement ce qu'il veut lorsqu'il va sur le site, il ira chercher le nom du produit qu'il a en tête dans la barre de recherche. Si le moteur de recherche n'est pas bien conçu, il risque de penser que nous n'avons pas le produit qu'il cherche et il ira donc voir ailleurs. Si il n'a qu'une idée très vague, il y a alors des chances qu'il se tourne vers le moteur de recherche avancée. Cela nous permettra de cerner ses critères de prix par exemple.
- Mettre en évidence le produit en mettant une image assez grande : trop de site proposent encore des images ridiculement petites, ce qui peut décourager le client d'acheter, car il ne peut pas bien voir le produit et peut même penser qu'il y a des risques de "tromperie" sur la marchandise.
- Une navigation aisée, c'est-à-dire des catégories bien organisées. Bien que notre site permette d'afficher à la fois des produits et des catégories sur une même page, cela sera à éviter dans le cas général. Enfin, il faudra veiller à ne pas créer de catégories vides, ce qui prendrait de la place pour rien et serait une source de frustration pour le client (quoi de plus désagréable que de trouver une catégorie qui correspond parfaitement à ce que l'on recherche et se rendre compte qu'elle est vide)
- Mettre l'accent sur les produits : le but d'un site de e-commerce est de vendre, et nous avons donc fait attention de mettre l'accent sur les produits. Le design est assez sobre, et les autres éléments du site n'empiètent pas sur les produits. Il y a également sur la page d'accueil des "coups-de-cœur" permettant de stimuler un achat imprévu chez le client.

2 Logique et Organisation

2.1 Fichiers

2.1.1 header.php

Ce fichier va insérer la bibliothèque Smarty et les déclarations de classes, puis il va déclarer des constantes et crée quelques fonctions.

2.1.2 index.php

Ce fichier va tout d'abord appeler header.php puis va créer l'objet Bob et smarty.

Il va ensuite analyser la requete pour savoir quel template il va appeler pour fabriquer la page et va effectuer les actions qu'on lui demande de faire.

Il va ensuite passer à Smarty plusieurs variables puis générer la page.

2.1.3 les fichiers de /classes et de /Smarty

Ces fichiers contiennent les classes qui contiennent les infos de la base de données et qui la manipule.

2.1.4 les fichiers de /templates

Ils permettent d'afficher une page en utilisant les variables qu'on lui prete, il y a deux style de templates :

- les fichiers modèle : ils se situent dans /templates/modèle
 - main.tpl : Contient ce que toute page va contenir, il va appeler les autres fichiers templates de modèle
 - menu.tpl : Contient le menu du site
 - entete.tpl : Contient les informations sur la page (< head >)
 - espace_membre.tpl : Contient l'espace membre
- le fichier appelé pour la page : ils se situent dans /templates.
Il y en a un par page

2.1.5 les fichiers de /css

Ils permettent de coder le design du site, ils se situent dans /css. Ces fichiers sont séparés en plusieurs fichiers. On préférera en mettre un général puis un par page.

2.1.6 En somme

Seul index.php permet de générer une page, il va commencer par appeler toutes les ressources à l'aide de header.php (qui inclut les classes de /classes et /Smarty), il va ensuite regarder le type de la requete :

- Est-ce pour le panneau d'admin, une image, une information ?
- Est-ce une page ou une action que l'on nous demande ?

En fonction de cela nous allons faire telle ou telle action (méthodes de la classe Bob) puis choisir tel ou tel template à appeller.

Une fois l'action faite et le template choisi, Smarty (l'objet) va prendre certaines variables (contenues dans Bob, ses attributs ainsi qu'un eventuel message) puis va utiliser le template pour générer une page.

Le template appelé est une extension du template modèle templates/-modele/main.tpl (divisé en plusieurs templates modèles), cela permet de garder une meme forme pour chaque page : seul le contenu, certains scripts sera modifié d'une page à l'autre !

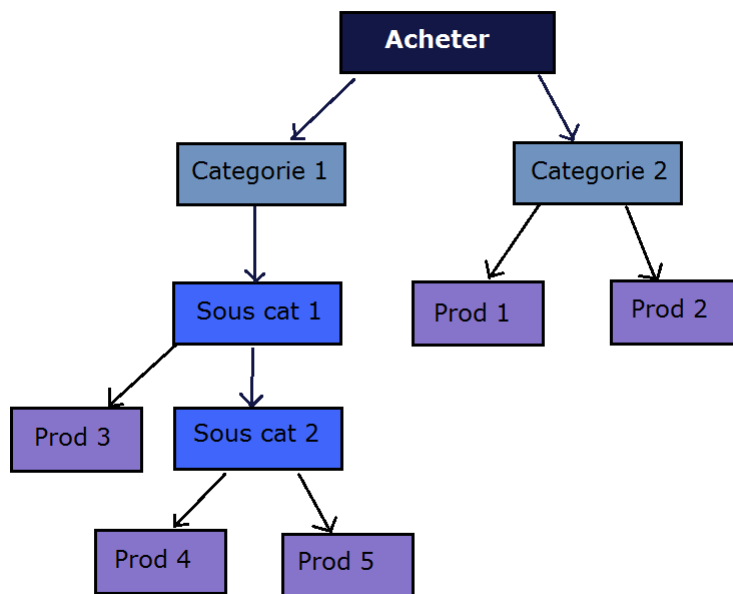
La page html ainsi générée fera appel à certaines feuilles de design de /css et à certains scripts /js

2.2 Catégories

Comme nous l'avons dit dans les objectifs du Site, la navigation se fait par un système de catégories qui peuvent contenir d'autres catégories et d'autres produits.

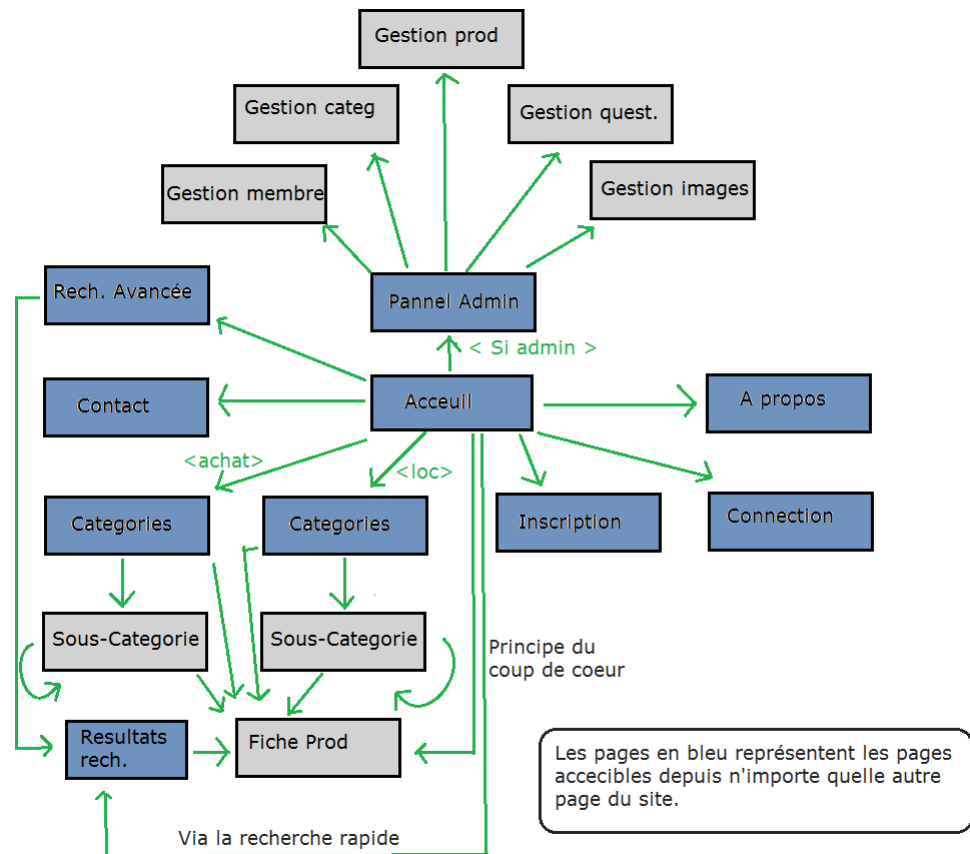
(c'est à l'administrateur à faire attention à ne pas de creer de catégories vides, pour des raisons d'ergonomie, il est aussi conseillé d'éviter de mettre catégories et produits ensemble)

Tout ceci se résume à un simple schéma :



2.3 Pages

Plutôt que de parler une à une de chaque page 'logique' du site, voici un schéma plus explicite :



3 Architecture des pages

3.1 Un modèle

L'architecture des pages est définie par les templates de modèle. Le modèle 'principal', main.tpl contient donc les éléments principaux de la page et le design leur donne leur position et les 'décore'.

Nous avons utilisé et sur-utilisé la balise `< div >` afin de pouvoir manipuler les éléments plus aisément avec le design. Dans le modèle et dans le contenu.

3.2 Le résultat

Une page va donc contenir :

- Un header contenant :
 - Le logo
 - l'espace membre et recherche
- Un corps contenant :
 - Le Menu : "Acheter, Louer, Contact et About"
 - Le Contenu : seul élément qui change (visuellement)
- Un pied de page contenant :
 - Le Sitemap (non codé)
 - Les droits d'utilisation/Les crédits
 - Les partenaires du site/de Brico-Bob

4 Technologies utilisées

4.1 Support du code

Pour le projet nous avons utilisé plusieurs supports :

Github : Il s'agit d'un système de partage de projets gratuit -si on accepte de les mettre open-source- très pratique.

Notepad++ : Il s'agit de notre éditeur de texte favori, nous l'avons tous utilisé

Wamp/Lampp : Il s'agit d'un émulateur de serveur Apache/MySQL/Php pour pouvoir tester le site en local.

Apache : Il émule le serveur en lui même, il contient notamment une interface "phpMyAdmin" et "Wamp" (pour les différents projets)

MySQL : Il émule la base de données

Php : Le programme qui, suivant la requête, interprétera la bonne page php.

The GIMP : C'est un outil pour manipuler les images très puissant : beaucoup d'éléments ont été réalisés grâce à lui.

4.2 Les langages de programmation

Nous avons utilisé, comme pour la plupart des sites, les langages :

HTML : Il définit les éléments de la page dans les templates

CSS : Il gère le design du site, nous avons séparé ses fichiers en plusieurs parties (Au départ il s'incluait tous, mais nous avons fait en sorte que Smarty n'appelle que ceux qui nous importent)

PHP : Il permet de créer la page HTML suivant la requête qu'on lui donne.

SQL : Il est utilisé par PHP pour manipuler la base de données, laquelle contient tous les éléments du site, lesquels sont stockés au chargement de la page dans des objets (PHP)

JavaScript : Il permet de faire "Bouger la page" chez le client, sans avoir besoin d'envoyer de requête PHP.

Cela permet, par exemple, de faire un premier contrôle local des données que le client veut envoyer mais cela permet aussi, de rendre certains éléments plus ergonomiques, par exemple pour la notation des produits.

4.3 Les bibliothèques utilisées

Nous avons utilisé deux bibliothèques :

AJAX : Exploité par le JavaScript, elle permet de récupérer une autre page et de traiter ces données alors que la page ayant appelé AJAX est encore active.

Elle permet, par exemple, alors que la page d'inscription est affichée, de demander au serveur si tel ou tel pseudo est utilisé ou non.

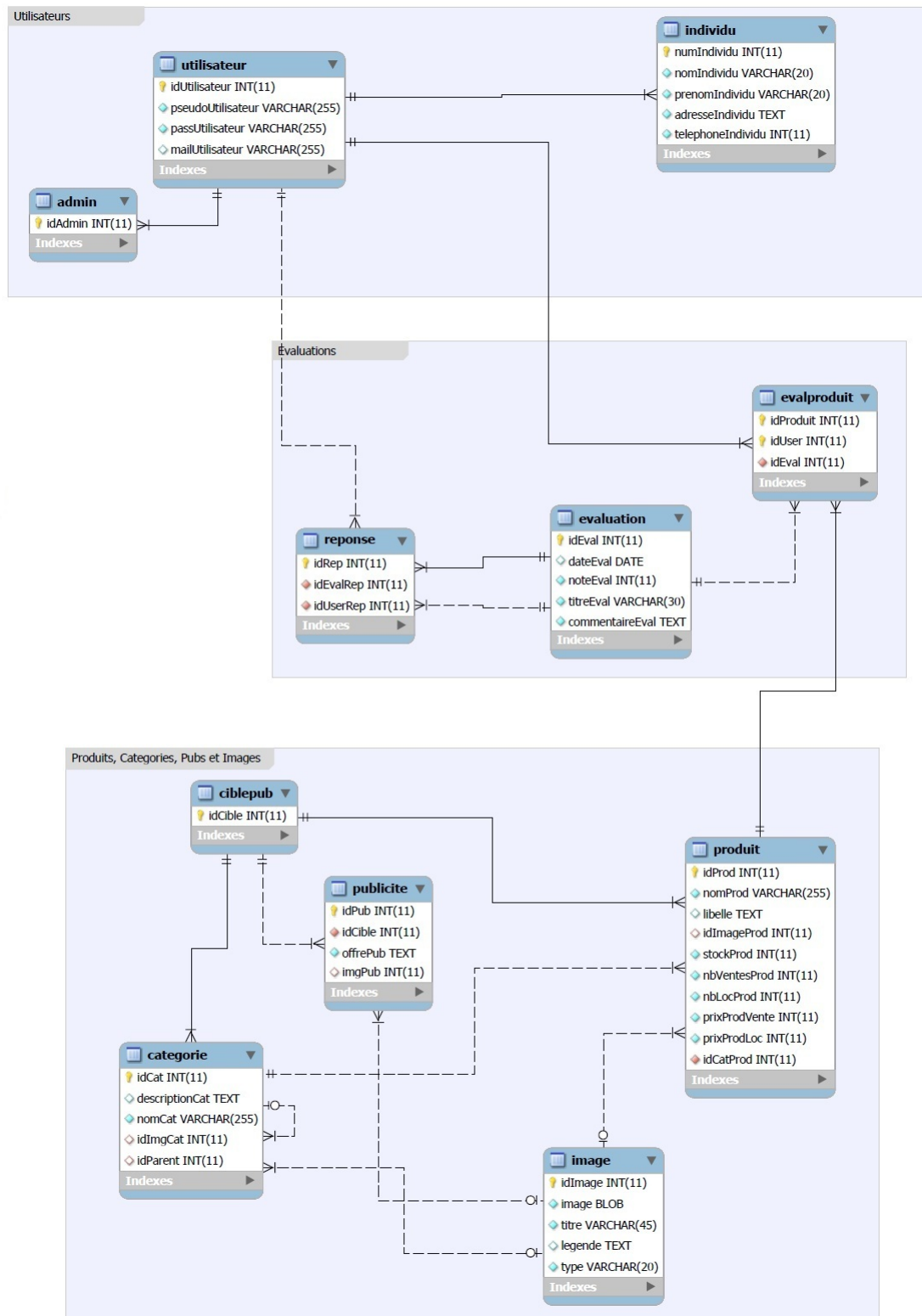
Smarty : C'est le moteur de templates utilisé, il permet de séparer traitement des données et mise en page : Le code PHP qui l'appelle va commencer par analyser la requête, et suivant son analyse va appeler tel ou tel template en lui 'passant' des variables qu'il pourra alors utiliser pour l'affichage.

5 Base de données

Elle contient toutes les infos du site, notamment :

- Les membres
- Les catégories
- Les produits
- Les commentaires
- Et même les images !

Voici un schéma relationnel de notre base (généré à l'aide de "MySQL Workbench")



6 Les sessions

Nous n'avons pas surutilisé les sessions : Nous ne les avons utilisé en fait, que pour la connexion :

Si l'internaute est connecté, alors une session contenant le booleen 'connecte' et les informations sur lui (dont l'id)

Avantages : Permet de détruire 'connecte' pour déconnecter l'utilisateur sans avoir à recharger la page.

Inconvénient : N'utilisant pas QUE l'id mais surtout les autres informations que nous avons entré à la connexion du membre, et bien si des modifications sont faites, l'utilisateur ne le verra qu'à sa prochaine connexion. (Il suffira d'utiliser l'id pour changer cela)

Nous pourrions utiliser les sessions pour un éventuel captcha lors de l'inscription.

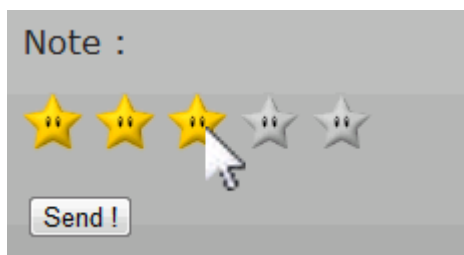
7 Portabilité

7.1 Différents navigateurs

Notre application est fonctionnelle sur les versions récentes de Google chrome, internet explorer et Firefox. Le site fonctionne également sur des versions un peu plus anciennes, par exemple sur IE6. (Nous n'avons pas utilisé de HTML5). La différence est que les navigateurs un peu anciens ne reconnaissent pas certaines propriétés CSS3 utilisées par exemple box-shadow ou les effets de dégradé. Cependant, nous avons fait en sorte que l'affichage reste "décent", même si le site sera quelque peu moins beau.

7.2 Javascript

Si le visiteur désactive le Javascript, le site ne sera pas moins fonctionnel, seulement, certains menus seront légèrement moins intuitifs et beaux. Il devra par exemple re-remplir tout le formulaire d'inscription si il fait une erreur mais cela ne changera rien au fait que l'inscription se déroulera normalement.



8 Flexibilité

8.1 Changement du design du site

Si le client veut changer le design du site, il suffira de changer les fichiers .css ainsi que les images dans le dossier img. Par exemple pour changer les couleurs, la commande "remplacer tout" de notepad++ permettrait de changer les codes hexa assez rapidement.

8.2 Type de matériel

Si le client veut que son site soit conçu pour une taille d'écran bien précise, il faudrait alors changer la taille de la page dans les fichiers ,css ainsi que d'agrandir certaines boites grâce a la propriété CSS width.

8.3 Changement du contenu

Notre site est adapté au cas ou on voudrait changer son contenu. Si le client ne souhaite plus vendre de produits de bricolage, mais de l'électroménager. Il suffirait de changer les noms des produits, catégories et de changer quelques images pour coller avec le nouveau thème du site.

Troisième partie

Manuel

8.4 Installation des fichiers

Pour installer tout ce dont vous avez besoin, vous pouvez passer par plusieurs méthodes :

- Vous pouvez aller sur : <https://github.com/sion77/Bob-Project>
- Vous pouvez aussi aller sur votre boîte mail où nous avons du vous envoyer le projet (allégé), dans ce cas vous devriez télécharger une version de Smarty (nous avons pris la 3.1.4, mais elle est bugée (nous avons fait avec car nous nous sommes pas posé de questions quant à certains dysfonctionnements))

8.5 Installation du serveur

Pour le serveur, plusieurs choix s'offrent à vous :

1. Vous pouvez louer un serveur ou utiliser un vrai serveur distant
2. Utiliser un émulateur tel que Lamp (Xampp) ou bien Wamp

8.6 Installation de la base de données

8.6.1 La base de données

Si vous avez utilisé un serveur loué ou un émulateur, il y a de fortes chances que la base de donnée soit déjà installée, si ce n'est pas le cas vous devez installer MySQL, voici une documentation pour :

<http://dev.mysql.com/doc/refman/5.5/en/installing.html>

8.6.2 PhpMyAdmin

Une fois la base de donnée installée, vous devriez pouvoir utiliser, si vous l'avez installé, PhpMyAdmin ou bien MySQL Server pour pouvoir l'utiliser : Wamp installe PhpMyAdmin et on peut y accéder sur <http://localhost/phpmyadmin>.

Sur Free on peut y accéder sur sql.free.fr

Sur phpMyAdmin, veuillez cliquer sur Importer et choisissez le fichier située dans (site)/docs/DB.sql

Si vous ne voulez/pouvez pas installer phpMyAdmin, vous pouvez toujours utiliser MySql Workbench, qui ne nécessite pas d'être installé sur le serveur (il s'y connecte)

8.6.3 Configurer le site

Pour indiquer au site comment se connecter à la base de données, veuillez ouvrir header.php et y modifier les lignes suivantes :

```
// Informations relatives a la BDD  
define("database_host", "localhost"); // Le serveur de la base de donnee  
define("database_port", 3306); // Le port du serveur  
define("database_name", "projet_bob"); // Le nom de la base  
define("database_user", "root"); // Le nom d'utilisateur  
define("database_pass", ""); // Le passe de l'utilisateur
```

8.7 Déplacement des fichiers

Ensuite il faudra déplacer le repertoire contenant le site au bon endroit : en local (emulateur), vous devez aller à l'endroit que localhost regarde, par exemple sous wamp, c'est dans le repertoire www situé à l'endroit où wamp est installé Et sur un serveur local, il s'agit de la racine, tout simplement.

Quatrième partie

Conclusion

Un site web regroupe beaucoup de langages et de technologies qui prennent du temps à utiliser, bien que leur utilisation soit redondante. Comme dans tout projet, une phase de conception s'impose, même si elle n'est pas forcément respectée.

Faute de temps et d'organisation, beaucoup de fonctionnalités n'ont pas encore été créées, cependant, tout a été fait pour pouvoir les ajouter facilement : si nous ne les avons pas codées ce n'est pas parce que nous n'y sommes pas arrivés.

En plus de tout cela, un site s'adresse généralement au grand public et se doit d'être extrêmement ergonomique, du design, au script en passant par l'architecture des pages.

Ce projet nous a permis de créer un site web plus ou moins complet, dans un délai assez court (nous trouvons).

```
<?php
// (...)
// Nous sommes dans Bob
private function initMembres()
{
    // Au depart : pas de membres.
    $this->nbMembres = 0;

    // Rappel : Admin est une classe speciale de Membre

    // On fait notre requete qui a pour but de recuperer
    // les membres et administrateurs
    $req = $this->query("

SELECT idUtilisateur AS \"id\",
        pseudoUtilisateur AS \"pseudo\",
        passUtilisateur AS \"pass\",
        '0' AS \"admin\"
FROM utilisateur
WHERE idUtilisateur NOT IN( SELECT idAdmin FROM admin )

UNION

SELECT idUtilisateur AS \"id\",
```



```

        pseudoUtilisateur AS \"pseudo\",
        passUtilisateur AS \"pass\",
        '1' AS \"admin\"
FROM utilisateur
WHERE idUtilisateur IN( SELECT idAdmin FROM admin )

ORDER BY id\"

);

// Pour chaque ligne du resultat
while($rep = $req->fetch())
{
    // On stocke le membre dans le bon objet
    if($rep[\"admin\"]
        $m = new Admin($this, $rep[\"id\"],
                        $rep[\"pseudo\"], $rep[\"pass\"]);
    else
        $m = new Membre($this, $rep[\"id\"],
                        $rep[\"pseudo\"], $rep[\"pass\"]);

    // On ajoute le membre au tableau de membres de l'objet Bob
    // (on aurait pu utiliser $this->ajouterMembre($m); )
    $this->membres[$this->nbMembres] = $m;
    $this->nbMembres++;
}

// On libere le resultat
$req->closeCursor();

return true;
}
// (...)
?>

```