UNIVERSITE D'ARTOIS
I.U.T. de Lens

DEPARTEMENT
INFORMATIQUE

Année 2006/2007

Compte rendu de projet

Réalisation d'un site de bookmarks

http://rantanplanbook.free.fr/

BOUTCHICH Anissa CARUYER Perrine SZYMKOWIAK Rémi

I.U.T. de Lens, Département Informatique, Rue de l'Université – SP16, 62307 Lens Cedex
Tél : 03.21.79.32.32 Email : info@iut-lens.univ-artois.fr Fax : 03.21.79.32.40

Table des matières

I.U.T. de Lens, Département Informatique, Rue de l'Université – SP16, 62307 Lens Cedex
Tél : 03.21.79.32.32 Email : info@iut-lens.univ-artois.fr Fax : 03.21.79.32.40

1) Développement :

1) HTML :

Nous avons utilisé pour le développement de notre projet web un langage informatique connu dans le domaine de l'Internet : l'HTML. Pour ce faire, nous avons appris à programmer sur le principe de ce langage, à savoir les balises.

Elles permettent de définir une zone d'application selon la balise choisie. Une balise ouvrante <balise> marque le début de la zone, alors qu'une balise fermante </balise> en marquera la fin. Par exemple et définissent un paragraphe. En html, il faudra définir les « limites » de la page grâce aux balises <html> et </html> , elle même constituée de la « tête » représentée par <head> et </head> , et du corps de page défini par <body> et < /body>

Certaines balises ne s'appliquent qu'en en point donné, ce sont des balises auto-fermantes <balise />. Par exemple le passage à la ligne ne peut s'appliquer à une zone, sa balise sera donc auto-fermante

Il est possible pour les balises de définir un caractère particulier, tel que la couleur, la taille etc. Cependant on utilisera ces propriétés uniquement lorsque le caractère ne s'applique qu'une seule fois.

L'HTML met à disposition du programmeur un système de tableaux, que nous avons utilisés notamment pour la mise en forme des formulaires. Par exemple, lorsqu'il s'agit de présenter la liste des tags parmi lesquels l'utilisateur peut choisir, on génère un tableau à l'intérieur duquel les tags peuvent s'aligner en colonnes. On obtient ainsi un affichage propre et soigné.

Le formulaire en HTML permet de mettre en place l'interface via laquelle l'utilisateur « communiquera » avec le site. Il s'agit alors de présenter un formulaire, entre les balises <form> et </form>. Ici la balise ouvrante <form> doit disposer d'attributs indispensables :

- method="…" : permet de choisir entre un envoi en GET ou en POST. La méthode GET enverra les données via l'URL, elles seront donc visibles pour l'utilisateur. Nous avons utilisé cette méthode lorsqu'il s'agissait par exemple d'une page particulière du site. En revanche, certaines données comme les mots de passe méritaient de rester « invisibles ». On a alors préféré une méthode POST, qui envoie les données d'une page à l'autre sans qu'elles soient visibles dans l'adresse.
- action="…" : permet de spécifier l'adresse de la page vers laquelle l'utilisateur sera dirigé en soumettant son formulaire, et donc vers laquelle les données seront envoyées.

Exemple extrait de la page préférences :

```
1 <form method="post" action="./Accueil.php?page=pref">
2 ...
3 </form>
```

Les données seront cachées à l'utilisateur. Après soumission du formulaire, l'utilisateur sera renvoyé vers la page « Accueil.php », avec en GET le nom de la page des préférences (ouverte en include dans la page Accueil) : il sera donc de nouveau sur la page où il se trouve.

Il existe de nombreux types d'objets à intégrer aux formulaires. :

- Le type « textarea » pour rentrer la description des URLs.
- Nous avons utilisé les champs « text » lorsqu'il s'agissait par exemple de rentrer le pseudonyme ou une url à ajouter.

Exemple extrait de la page Menu :

```
1 <input class="boite" type="text" name="surch" size="16">
```

Ici l'input s'appellera « surch », le champ aura une taille de 16 pixels.

- Un type « password » pour que le mot de passe ne s'affiche pas quand l'utilisateur le rentre.
- Un type « select » lorsqu'il s'agit de sélectionner une note dans un menu déroulant.

I.U.T. de Lens, Département Informatique, Rue de l'Université – SP16, 62307 Lens Cedex
Tél : 03.21.79.32.32 Email : info@iut-lens.univ-artois.fr Fax : 03.21.79.32.40

```
<select name="noteRentree">
    <option>1</option>
    ...
    <option>10</option>
</select>
```

Ici les options spécifient les choix des utilisateurs. La note choisie sera retournée dans la variable qui portera le nom « noteRentree ». (accessible par $_POST[‘noteRentree’])

- Un type « radio » pour choisir si le mail doit être privé ou public (il sera soit l’un, soit l’autre, l’utilisateur ne peut donc pas choisir les deux à la fois).

Un type « checkbox » quand il s’agissait de choisir les tags associés à chaque url. L’utilisateur peut en choisir autant qu’il veut s’il juge qu’ils sont appropriés.

2) PHP :

Pour Pouvoir Dynamiser notre site web et ainsi établir une communication entre MySQL et l’HTML, nous avons mis en œuvre des pages contenant du code PHP. Cela nous permettra entre autre, d’utiliser les classes.

PHP étant utilisé dans divers scripts à générer dans le site, nous allons vous présenter et expliquer un exemple de divers scripts spécifiques :

Vérification de session :

inscription.php

```php
<?php
session_start(); //Ouverture de la session
if( array_key_exists('ip',$_SESSION) && ($_SESSION["ip"] != $REMOTE_ADDR) )
{
    session_destroy(); // Destruction des données de la session
    header ('Location: Accueil.php'); // Rechargement de la page d'accueil
    exit;
}
?>
```

Pour ce premier exemple, nous pouvons remarquer que le code PHP doit se trouver entre les balises « < ?php »et « > ».

Avant tout, nous devons créer une session (ou restaurer celle trouvée sur le serveur, via l'identifiant de session passé dans une requête GET, POST ou par un cookie). Ceci grâce au « session_start() ». La condition du if vérifie qu’une variable nommée « ip » existe dans la session existante et vérifie si cette variable est différente de l’ip de l’utilisateur courant. Si cela est le cas, nous pouvons supposer qu’il s’agisse d’une tentative de piratage, donc nous détruisons les données de la session avec un « session_destroy() ». Ceci fait, nous chargeons la page d’accueil grâce au « header() ».

Include/Require :

inscription.php

```php
include("./include/Menu.php");
```

sortirDuSite.php

```php
require("BD.class.php");
```

Ces fonctions de php permettent de charger une page puis de l’exécuter. Les deux structures de langage sont identiques, hormis dans leur gestion des erreurs. Ils produisent tous les deux une alerte mais « require() » génère une erreur fatale, en d'autres termes, nous avons utilisé « require() » lorsque nous voulions qu'un fichier d'inclusion manquant interrompe notre script.

I.U.T. de Lens, Département Informatique, Rue de l’Université – SP16, 62307 Lens Cedex
Tél : 03.21.79.32.32 Email : info@iut-lens.univ-artois.fr Fax : 03.21.79.32.40

Fonction SQL :

mesURLs.php

```php
// On récupère le nombre total de messages
$sql='SELECT COUNT(*) AS \'nb_messages\'
        FROM `link`
        WHERE `userLogin`="'.$_SESSION['login'].'"
            AND `nomUrl` != "";';

$sql = mysql_query($sql) or die('Erreur SQL !<br>'.$sql.'<br>'.mysql_error());
$donnees = mysql_fetch_array($sql);
$NbrTotDUrl = $donnees['nb_messages'];
```

Tout d'abord, nous enregistrons une requête dans une variable « $sql ». La fonction « mysql_query() » retourne le tableau résultat de la requête passée en paramètre. Nous utilisons aussi la fonction « die() » qui renvoie un message d'erreur en cas d'erreur de « mysql_query() ». Afin de pouvoir récupérer les données du tableau résultat, nous utilisons « mysql_fetch_array() » qui retourne un tableau qui correspond aux lignes du tableau rentré en paramètre. Dans ce cas particulier, nous sommes sûr de n'avoir qu'une seule ligne, nous affectons donc directement le nombre de message dans une variable « $nbrTotDUrl ».

mdpPerdu.php

```php
$result = mysql_query($sql) or die ('Erreur SQL !<br>'.$sql.'<br>'.mysql_error());

if ( mysql_num_rows($result)!==0 )// Si le pseudo rentrée existe dans la BD, on affiche la question
{
    $pseudoOk = true;
    while($donnees = mysql_fetch_array($result))
    {
        echo $donnees['questionMdp']."<br />";
```

Dans cet exemple, nous voyons l'utilisation de la fonction « mysql_fetch_array() » dans une boucle « while() ». Il faut savoir que la fonction déplace le pointeur de données interne d'un cran à chaque appel (à la prochaine itération, on accédera à l'élément suivant). De plus, elle retourne FALSE s'il n'y a plus de lignes. De ce fait, grâce à la boucle « while() », nous parcourons l'ensemble du tableau « $result » ligne par ligne.

Nous pouvons remarquer la présence d'un « !== » dans le « if ». Celui-ci regarde si deux variables sont de valeurs ou de types différents, contrairement à un simple « != » qui ne vérifie que les valeurs. Notons aussi l'existence du « = = = »dans le même esprit.

Foreach :

recherche.php

```php
foreach($_SESSION['tagSurch'] as $choix)
{
    $sql = $sql.' AND `tag`.`name` = "'.$choix.'" ';
```

Utilisant des « checkbox » dans un formulaire, nous avons utilisé une boucle « foreach() ». Celle ci passe en revue le tableau « $_SESSION['tagSurch'] » contenant les différentes « checkbox », cochées précédemment par un utilisateur. A chaque itération, la valeur de l'élément courant est assignée à « $choix » et le pointeur interne de tableau est avancé d'un élément.

Les cookies :

Accueil.php

```php
if ( array_key_exists('style',$_POST) && $_POST['style'] !== 0)
{
    setcookie('style'.$_SESSION['login'],$_POST['style'],time()+3600*24*365);
    header("Location: Accueil.php"); // Rechargement de la page d'accueil
    exit;
}
```


I.U.T. de Lens, Département Informatique, Rue de l'Université – SP16, 62307 Lens Cedex
Tél : 03.21.79.32.32 Email : info@iut-lens.univ-artois.fr Fax : 03.21.79.32.40

L'utilisation des cookies n'étant pas conseillée pour diverses raison, nous nous sommes appliqués à n'en utiliser aucun pour les données importantes du site. Cependant, étant donné l'aspect pédagogique de ce projet, nous nous en sommes tout de même servis afin d'enregistrer le style graphique que l'utilisateur souhaite avoir pour le site. Pour cela, nous avons utilisé la fonction « setcookie() » qui définit un cookie qui sera envoyé avec le reste des en-têtes. Les cookies doivent passer avant tout autre en-tête (c'est une restriction des cookies, pas de PHP). Cela nous à donc imposé d'appeler cette fonction avant toute balise <html> ou <head>.

Générateur de mot de passe :

mdpPerdu.php

```
78  $tableau = array("a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p",
79                   "q", "r", "s", "t", "u", "v", "w", "x", "y", "z", "A", "B", "C", "D", "E", "F",
80                   "G", "H", "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V",
81                   "W", "X", "Y", "Z", "1", "2", "3", "4", "5", "6", "7", "8", "9", "0");
82  $valeursAleatoires = array_rand($tableau, 8);
83  $motDePasse = "";
84  $cpt=0;
85  for($cpt;$cpt<=8;$cpt++)
86  {
87      $motDePasse = $motDePasse . $tableau[$valeursAleatoires[$cpt]];
88  }
```

Afin de pouvoir fournir un nouveau mot de passe aux utilisateurs ayant perdu le leur, nous avons créé un générateur de mot de passe. Son principe est très simple. Dans un premier temps nous créons un tableau contenant les lettres de l'alphabet en minuscules et majuscules ainsi que les chiffres de 0 à 9. Ensuite nous mettons, dans un second tableau, 8 valeurs aléatoires grâce à la fonction « array_rand() ». Pour finir nous concaténons les valeurs du tableau se trouvant aux positions des valeurs aléatoires. Cela nous permet d'avoir un nouveau code totalement aléatoire à chaque appel.

Encodage : Cf annexe 1

Pour des raisons de sécurité évidente, nous nous sommes servis du codage « md5 » (*Message Digest 5*) pour mots de passe. « Md5 » est une fonction de hachage cryptographique qui permet d'obtenir pour chaque message une *empreinte numérique* (en l'occurrence une séquence de 128 bits ou 32 caractères en notation hexadécimale) avec une probabilité très forte que, pour deux messages différents, leurs empreintes soient différentes.

Les fonctions :

inscription.php

```
 9  function mailOk($email)
10  {
11  $regex = "^([~._a-z0-9-]+[~._a-z0-9-]*)@(([a-z0-9-]+\.)*([a-z0-9-]+)(\.[a-z]{2,3}))$";
12  return (eregi($regex, $email));
13  }
```

Pour un code plus propre, nous avons créé une fonction nommée « mailOk() ». Celle-ci, constituée principalement avec des expressions régulières, vérifie la bonne syntaxe d'une adresse e-mail entrée en paramètre. Cela permet une exclusion relativement fiable d'adresses électroniques à la syntaxe erronée. Cela dit, une adresse électronique dont la syntaxe est correcte n'est pas forcément une adresse électronique valide! Afin de comparer l'adresse e-mail rentrée et l'expression régulière, nous avons utilisé la fonction « eregi » qui est une alternative de la fonction « ereg ».

Pour ce qui est de l'appel, il se fait simplement en mettant le nom de la fonction puis son(ses) paramètre(s) entre parenthèses comme dans l'exemple ci-dessous :

inscription.php

```
78  if(mailOk($m)) // Vérification de la syntaxe de l'adesse e-mail
79  {

115 }
116 else // L'adresse e-mail est incorrecte
117 {
118     echo "Veuillez entrer une adresse mail correcte.";
119 }
```


I.U.T. de Lens, Département Informatique, Rue de l'Université – SP16, 62307 Lens Cedex
Tél : 03.21.79.32.32 Email : info@iut-lens.univ-artois.fr Fax : 03.21.79.32.40

Les classes :

BD.class.php

```
1  <?php
2  class DB {
3    private $debug;
4
5  //constructeur : connexion a la base
6  //if $d=1 then debug is active
7    function __construct($d = 0) {
8      $this->debug = $d;
14     $Base = "marksbook";          // Nom de la base
15     $utilisateur = "root";        // Utilisateur
16     $mdp = "";                    // Mot de passe
17     $hote = "localhost";
18
19     $db=mysql_connect($hote,$utilisateur,$mdp)
20       or die("Connexion impossible");
21     $se = mysql_select_db($Base,$db)
22       or die("Connexion base impossible");
23   }
```

L'utilisation de PHP5 nous a permis l'utilisation des classes. Nous nous en sommes principalement servis pour simplifier les commandes envoyées à MySQL. Nous pouvons remarquer à la ligne 8 le mot clé « $this » qui permet de désigner l'objet dans lequel on se trouve.

Pour l'utilisation d'une classe, plusieurs étapes doivent être réalisées. Tout d'abord, Nous devons inclure la classe dans la page dans laquelle nous en avons besoin. Ceci grâce aux fonctions « include » ou « require », vue précédemment, afin de la déclarer. Après avoir déclaré une classe, il faut instancier des objets pour pouvoir l'exploiter. Cette opération se fait à l'aide du mot clé « new » permettant de faire des objets découlant d'une classe :

inscription.php

```
71  if ($c == null)
72      include("BD.class.php");
73  $c = new DB(1);
```

Ensuite, il ne nous reste plus qu'à accéder aux fonctions de l'objet créé. Ceci se fait grâce au nom de l'objet, suivi d'une flèche « -> », puis du nom de la donnée membre (sans le signe $) :

inscription.php

```
98  $c->insert("user",$u);
```

3) CSS :

Nous utilisons maintenant pour la mise en page des sites web les feuilles de style css. Il s'agit de définir le style de la page dans un fichier annexe, ce qui permet de regrouper que l'on joint à la page de la manière suivante :

```
1  <head>
2      <link rel="stylesheet" type="text/css" href="styleDalton.css" />
3  </head>
```

Il existe différentes façons de définir le style :

- on applique les caractéristiques souhaitées à toutes les balises correspondant au type sélectionné. Exemple extrait de la feuille de style styleDalton.css :

```
1  h3 {
2  color : #ffff00;
3  background-image:url(./design/black/fond_menu.png);
4  text-align:center;
5  }
```


I.U.T. de Lens, Département Informatique, Rue de l'Université – SP16, 62307 Lens Cedex
Tél : 03.21.79.32.32 Email : info@iut-lens.univ-artois.fr Fax : 03.21.79.32.40

Ici tous les titres situées entre les balises <h3> et </h3> seront de couleur jaune (=#ffff00), ils auront une image de fond correspondant au chemin choisi, et enfin ils seront centrés.

- on défini des caractéristiques en fonction du type et du « sous type ». Exemple :

```
a:visited {
 color : #ffff00;
 text-decoration:none;
}
```

Ici, les instructions concernent les liens qui ont été visités, ils seront de couleur jaune et sans décorations (c'est-à-dire en texte normal).

- on défini un style particulier à n'appliquer que lorsqu'il est appelé. Exemple :

```
.grandTitre {
text-align:center;
padding-top:10px;
padding-bottom:5px;
background-image:url(./design/black/fond_titre.png);
}
```

Ici on appliquera ce style au grand titre. Il sera centré, avec limage de fond située au chemin spécifié. Un espace de 10 pixels sera réservé au dessus du titre, et 5pixels en dessous.

- on utilise les feuilles de style pour la mise en forme globale, c'est à dire pour placer les objets sur la page. Exemple :

```
.flottant {
float : left;
}
```

Dans notre projet, nous avons placé les includes « identification » et « menu » entre les balises et . Ainsi, la division bénéficiera du style « flottant », et sera située sur la gauche de la page.

4) MySQL :

1) Mcd :

Au cours de la conception de notre projet, nous avons reconsidéré le Modèle Conceptuel de Données qui nous était fourni, et nous lui avons apporté quelques modifications :

- Nous avons ajouté une table livre d'or avec son identifiant, un champ pseudo et un autre message, afin que l'utilisateur puisse nous donner son avis.
- Nous avons ajouté une table minichat avec son identifiant, un champ pseudo et un autre message, pour que les utilisateurs puissent discuter entre eux.
- Nous avons préféré relier la table des tags à celle des URLs directement, plutôt que de l'associer à la table link. Cela nous semblait plus logique d'associer a chaque URL les tags qui lui correspondent via une table isTagged (représentée par l'association 1,n 0,n entre link et tags).
- Nous avons rajouté, dans la table user, un champ « mailPublic » qui retourne 1 si le mail est public, 0 s'il ne l'est pas. D'autres champs questionMdp, et reponseMdp qui permettront de vérifier l'identité de l'utilisateur s'il venait à perdre son mot de passe.
- Nous avons déplacé le champ nbClics de la table link, vers la table url. Nous pensions qu'il ne nous serait pas utile de connaître le nombre de clics par utilisateur et lien, nous avons préféré comptabiliser uniquement par URL.
- Nous avons ajouté à la table link un champ nomUrl, qui permettra à l'utilisateur d'attribuer à chaque url le nom qu'il souhaite, et ce afin qu'il puisse prendre ses marques parmi ses urls.
- Nous avons rajouté à la table url un champ premUser, dans lequel on conservera le nom du premier utilisateur à avoir rentré cette url dans son marksbook. Cet utilisateur bénéficiera

I.U.T. de Lens, Département Informatique, Rue de l'Université – SP16, 62307 Lens Cedex
Tél : 03.21.79.32.32 Email : info@iut-lens.univ-artois.fr Fax : 03.21.79.32.40

des droits sur cette url (il pourra changer son nom, sa description etc…). Si cet utilisateur venait à supprimer l'URL de son marksbook, ce serait l'utilisateur l'ayant rentrée le plus anciennement qui hériterait des droits.

Le Modèle Conceptuel de Données que nous avons obtenu et disponible en annexe 2.

2) Mld :

On déduit de ce MCD les tables suivantes, dont les clés primaires sont soulignées et les clés étrangères sont en gras :

- user(login : varchar(32), mdp : varchar(32), email : varchar(255), date : date, questionMdp : text, reponseMdp : varchar(32)) : Cette table correspond à un utilisateur inscrit le « date ». Les mots de passe ainsi que les « reponseMdp » seront encodés dans la base de donnée grâce à la fonction « md5() » de php.
- url(id : int,url : varchar(255), premUser : varchar(32), nbClic : int) : Les urls sont inscrits une seule fois dans la base. Nous stockons le nombre de clics de chaque url dans « nbClic ».
- link(id : int, description : text, date : date, note : int, public : tinyint, nomUrl : varchar(255)) : Cette table correspond au lien « idUrl » d'un utilisateur « userLogin » créé à la date. Le champ public sert à savoir si un utilisateur non inscrit voit le lien lors d'une recherche.
- tag(id : int, name : varchar(100), date : date) : table référençant les différents tags.
- isTagged(idURL : int, idTag : int) : Table construite à partir d'une relation (0n,0n).
- livredor(id : int, **pseudo** : varchar(32), description : text)
- minichat(id : int, **pseudo** : varchar(32), description : text)

3) Les requêtes

Une fois les tables construites, nous avons du écrire de nombreuses requêtes SQL. Que ce soit simplement pour récupérer une donnée ou pour en rajouter, l'ensemble du site repose sur la bonne écriture des requêtes. Il nous a donc fallu prendre notre temps afin d'écrire, de vérifier et de tester chaque requêtes.

Nous allons maintenant voir les principales fonctions des requêtes que nous avons construites.

Les bases :

desinscription.php

```
55  $sql = 'SELECT *
56          FROM link
57          WHERE userLogin="'.$_SESSION['login'].'";';
```

Dans cette requête, nous pouvons distinguer les trois mots clés de base :
« SELECT » <liste des noms de colonnes>
« FROM » <Liste des tables>
[« WHERE » <condition logique>]

La **liste des noms de colonnes** indique la liste des colonnes choisies, séparées par des virgules. Lorsque l'on désire sélectionner l'ensemble des colonnes d'une table il n'est pas nécessaire de saisir la liste de ses colonnes, l'option ***** permet de réaliser cette tâche.

La **liste des tables** indique l'ensemble des tables (séparées par des virgules) sur lesquelles on opère.

La **condition logique** permet d'exprimer des qualifications complexes à l'aide d'opérateurs logiques et de comparateurs arithmétiques.

I.U.T. de Lens, Département Informatique, Rue de l'Université – SP16, 62307 Lens Cedex
Tél : 03.21.79.32.32 Email : info@iut-lens.univ-artois.fr Fax : 03.21.79.32.40

GROUP BY / ORDR BY DESC / LIMIT :

top10mark.php

```
19  $sql='SELECT count(*) AS `nbUser`,
26      FROM    `url`,
27              `link`
28      WHERE   `url`.`id` = `link`.`idUrl`
29      GROUP BY `url`.`id`
30      ORDER BY `nbUser`
31      DESC LIMIT 0, 10;';
```

Afin de faire des opérations par groupe (nombre total d'utilisateurs sur cet exemple), il nous a fallu regrouper des résultats. Nous avons réalisé cette opération à l'aide de la clause « *GROUP BY* », suivie du nom de la colonne sur laquelle nous voulions effectuer le regroupement.

La commande « *ORDER BY* » permet de classer alphabétiquement les données retournées par la requête de sélection. Pour trier dans l'ordre inverse, nous avons ajouté le mot-clé « DESC » (descendant) au nom du champ.

La clause « LIMIT » limite le nombre d'enregistrements retournés par la commande « SELECT ». Les deux arguments numériques sont des entiers constants : le premier indique le décalage du premier enregistrement à retourner, le second donne le nombre maximum d'enregistrement à retourner.

Sous requête :

desinscription.php

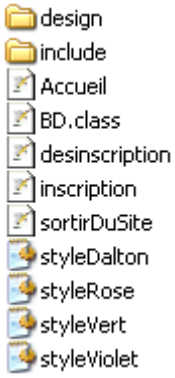
```
68  $sql1 = 'SELECT *
69          FROM url
70          WHERE url.premUser = "'.$_SESSION['login'].'"
71              AND url.id NOT IN ( SELECT link.idUrl
72                                  FROM link);';
```

Les sous requêtes résolvent bien des problèmes en bases de données relationnelles. Cela consiste à faire une sélection sur des éléments qui ne sont pas (« NOT IN »)dans un autre ensemble que l'on définit à l'aide d'une requête (la sous requête).

2) Analyse :

1) L'arborescence :

Nous avons représenté graphiquement l'arborescence et les échanges entre les pages de notre site web. Ce graphique c'est disponibles en annexe 3.



- *Le dossier « design » contient toutes les images des différents styles dont nous nous servons pour le design.*
- *Le dossier « include » contient tous les fichiers appelés grâce à la fonction « include ».*
- *« BD.class » est la classe que nous avons créée.*
- *Les quatre fichiers « style » sont les feuilles CSS que nous utilisons.*

I.U.T. de Lens, Département Informatique, Rue de l'Université – SP16, 62307 Lens Cedex
Tél : 03.21.79.32.32 Email : info@iut-lens.univ-artois.fr Fax : 03.21.79.32.40

2) Problèmes rencontrés :

Notre premier soucis a été de pouvoir tester notre site web à partir de nos ordinateurs personnels. Nos premiers essais se sont avérés être des échecs, nous ne pouvions utiliser la classe fournie (BD.class.php). En effet, nous devions pour ce faire disposer de l'installation php5. Or nous ignorions que le logiciel « easyphp » n'avait évolué que jusque php4. Ainsi, après avoir recherché la cause du problème, nous avons entrepris d'installer « wamp », qui mettait alors à notre disposition php5, ainsi que l'interface phpmyadmin. Nous avons enfin pu tester nos pages sur nos ordinateurs.
Le second problème a été de gérer le travail en équipe. Nous n'avions encore jamais eu à gérer un tel projet en groupe, et nous avons du nous organiser. En effet, nous devions toujours savoir qui, changeait quelle page, à quel moment…

Par ailleurs, nous avons rencontrés de nombreux problèmes (« bug ») au cours de la réalisation de ce projet. La plupart d'entres eux ont été liés aux requêtes SQL. Leurs syntaxes doivent être scrupuleusement exactes. Les plus gros ennuis ont été principalement dus aux « petites » erreurs.

Exemple : Un « idUrl » se transformant en « id ». Le champ « id » existant aussi, aucune erreur n'est renvoyée par le code. Mais, bien évidemment, le résultat de la requête était totalement faux…

3) BILAN :

Pour réaliser notre projet, nous avons dans un premier temps modifié les bases de données selon notre façon de concevoir le site. Nous avons déplacé des champs, nous en avons ajouté d'autres, nous avons ajouté des tables, et ce de façon à imaginer, selon nous, un site cohérent.

Puis, nous avons mis en place les pages Internet grâce à une combinaison d'HTML et de PHP5. Ainsi, nous avons pu mettre en place un site dynamique, permettant à l'utilisateur d'enregistrer ses propres informations en ligne, ou d'en consulter d'autres. Nous avons voulu mettre à sa disposition un nombre suffisant de possibilités :

- Il peut naturellement s'inscrire et se désinscrire.
- Il peut changer son mot de passe et son adresse e-mail à tout moment, ou décider que son adresse e-mail soit publique ou privée.
- S'il venait à perdre son mot de passe, il devrait répondre à une question qu'il aurait choisie et un nouveau mot de passe lui serait envoyé par e-mail.
- Il peut ajouter et supprimer des URLs comme il le souhaite, en leur choisissant le nom qu'il désire pour les retrouver facilement. Il peut également décider qu'elle soit visible ou non pour les autres utilisateurs.
- Rechercher des URLs rapidement grâce à un champ de recherche dans le menu, ou lancer une recherche plus sélective dans la recherche avancée.
- Il dispose d'un tchat et d'un livre d'or.
- Il peut consulter les 10 sites les mieux notés, les plus visités ou les plus souvent ajoutés dans les marksbooks.
- Il peut consulter une page expliquant le fonctionnement, une autre qui présente notre équipe, une Foire Aux Questions.
- Il peut nous contacter par e-mail en cas de problème.

Ainsi, nous avons voulu apporter à l'utilisateur un maximum de disponibilités.

Ce projet, qui fera l'objet pour nous d'une première soutenance technique, aura donc été une expérience très enrichissante. Il nous a permis d'apprendre la programmation web (HTML et PHP), à concevoir une arborescence logique et ordonner nos idées. Enfin nous avons appris a gérer un travail en équipe, ou tout simplement, à mener à bien notre premier projet de programmation.

I.U.T. de Lens, Département Informatique, Rue de l'Université – SP16, 62307 Lens Cedex
Tél : 03.21.79.32.32 Email : info@iut-lens.univ-artois.fr Fax : 03.21.79.32.40

Annexes

I.U.T. de Lens, Département Informatique, Rue de l'Université – SP16, 62307 Lens Cedex
Tél : 03.21.79.32.32 Email : info@iut-lens.univ-artois.fr Fax : 03.21.79.32.40

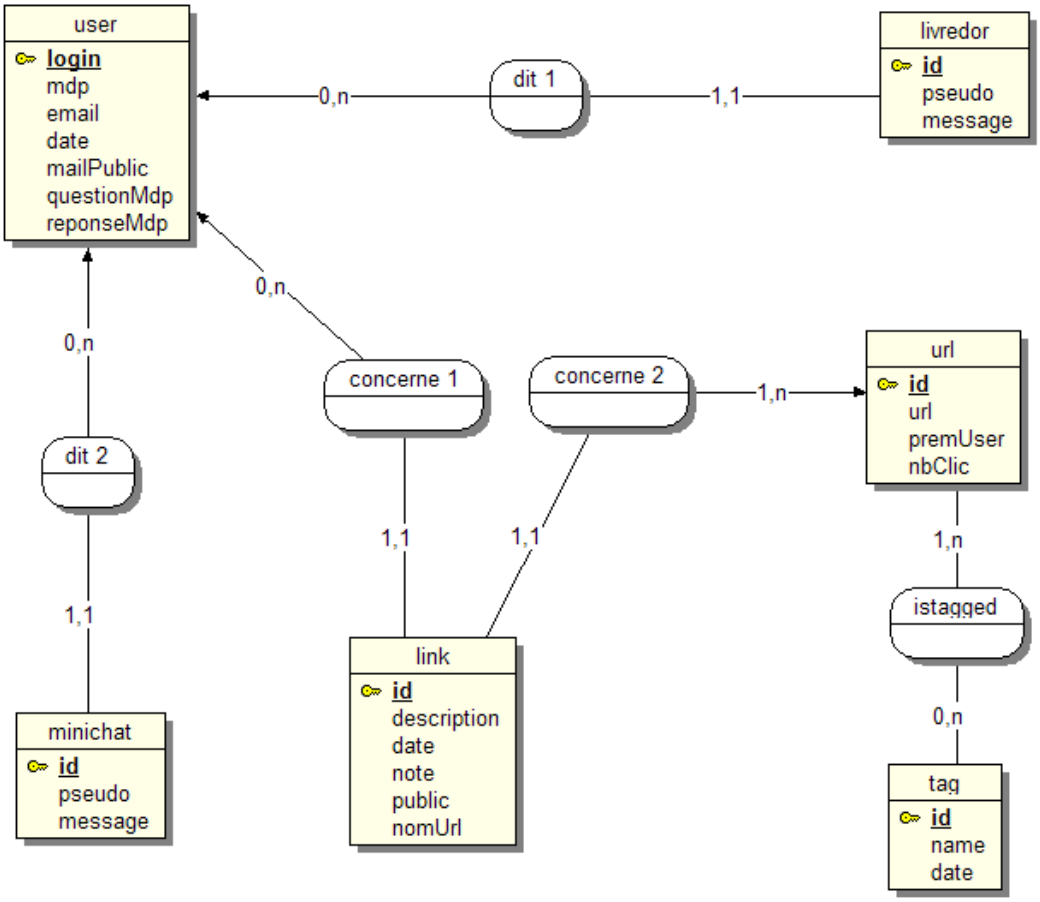
Annexe 1

MD5 peut s'écrire sous cette forme en pseudo-code

```
//Note: Toutes les variables sont sur 32 bits

//Définir r comme suit :
var int[64] r, k
r[ 0..15] := {7,  12,  17,  22,   7,  12,  17,  22,   7,  12,  17,  22,   7,  12,  17,  22}
r[16..31] := {5,   9,  14,  20,   5,   9,  14,  20,   5,   9,  14,  20,   5,   9,  14,  20}
r[32..47] := {4,  11,  16,  23,   4,  11,  16,  23,   4,  11,  16,  23,   4,  11,  16,  23}
r[48..63] := {6,  10,  15,  21,   6,  10,  15,  21,   6,  10,  15,  21,   6,  10,  15,  21}

//MD5 utilise des sinus d'entiers pour ses constantes:
pour i de 0 à 63 faire
    k[i] := floor(abs(sin(i + 1)) × 2^32)
fin pour

//Préparation des variables:
var int h0 := 0x67452301
var int h1 := 0xEFCDAB89
var int h2 := 0x98BADCFE
var int h3 := 0x10325476

//Préparation du message (padding) :
ajouter "1" bit au message
ajouter "0" bits jusqu'à ce que la taille du message en bits soit égale à 448 (mod 512)
ajouter la taille du message codée en 64-bit little-endian au message

//Découpage en blocs de 512 bits:
pour chaque bloc de 512 bits du message
    subdiviser en 16 mots de 32 bits en little-endian w[i], 0 ≤ i ≤ 15

    //initialiser les valeurs de hachage:
    var int a := h0
    var int b := h1
    var int c := h2
    var int d := h3

    //pour principale:
    pour i de 0 à 63 faire
        si 0 ≤ i ≤ 15 alors
            f := (b et c) ou ((non b) et d)
            g := i
        sinon si 16 ≤ i ≤ 31 alors
                f := (d et b) ou ((non d) et c)
                g := (5×i + 1) mod 16
            sinon si 32 ≤ i ≤ 47 alors
                    f := b xor c xor d
                    g := (3×i + 5) mod 16
                sinon si 48 ≤ i ≤ 63 alors
                        f := c xor (b ou (non d))
                        g := (7×i) mod 16
                    fin si
                fin si
            fin si
        fin si

        var int temp := d
        d := c
        c := b
        b := ((a + f + k[i] + w[g]) leftrotate r[i]) + b
        a := temp
    fin pour

    //ajouter le résultat au bloc précédent:
    h0 := h0 + a
    h1 := h1 + b
    h2 := h2 + c
    h3 := h3 + d
fin pour

var int empreinte := h0 concaténer h1 concaténer h2 concaténer h3 //(en little-endian)
```

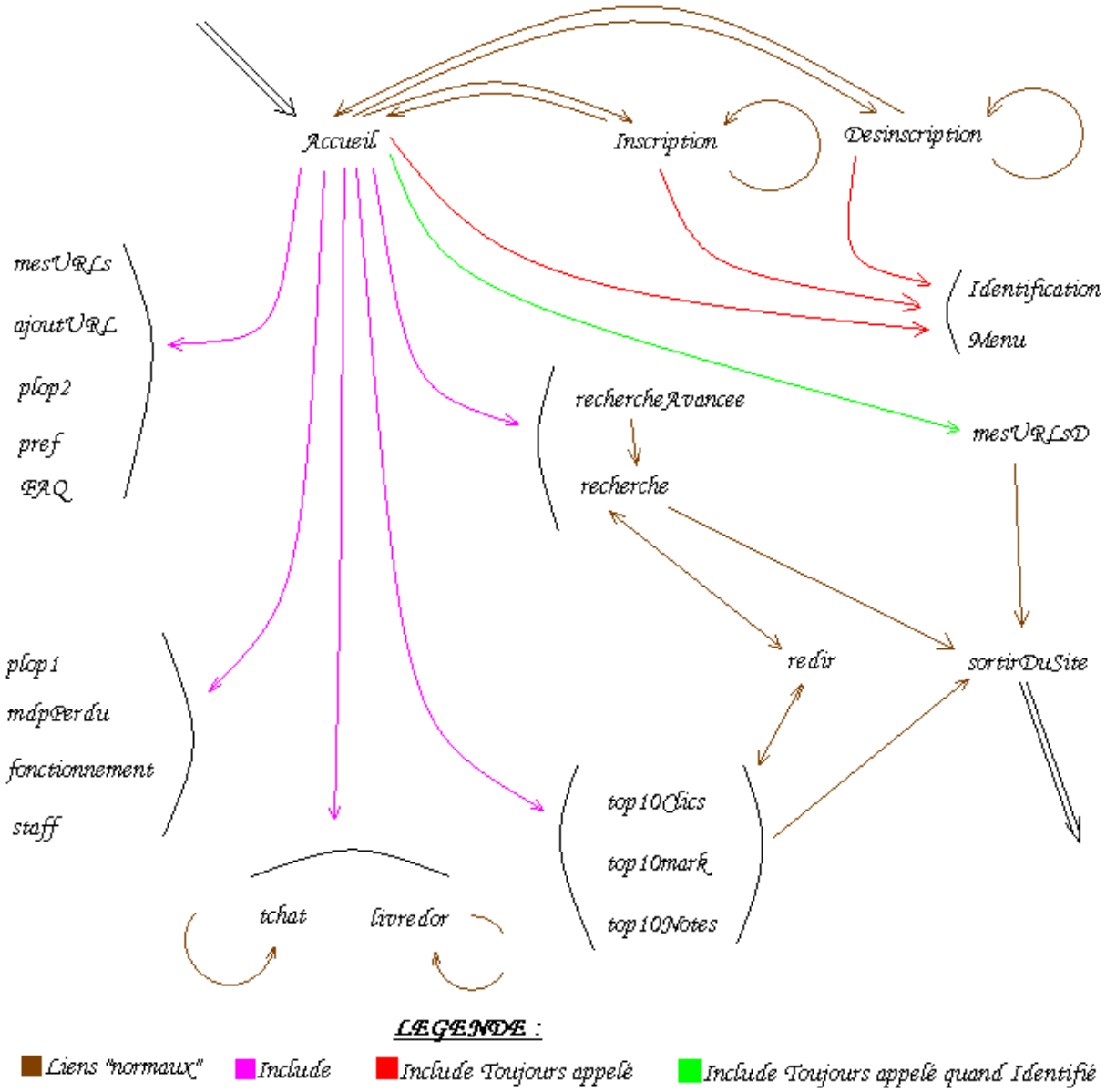

I.U.T. de Lens, Département Informatique, Rue de l'Université – SP16, 62307 Lens Cedex
Tél : 03.21.79.32.32 Email : info@iut-lens.univ-artois.fr Fax : 03.21.79.32.40

Annexe 2



I.U.T. de Lens, Département Informatique, Rue de l'Université – SP16, 62307 Lens Cedex
Tél : 03.21.79.32.32 Email : info@iut-lens.univ-artois.fr Fax : 03.21.79.32.40

Annexe 3



I.U.T. de Lens, Département Informatique, Rue de l'Université – SP16, 62307 Lens Cedex
Tél : 03.21.79.32.32 Email : info@iut-lens.univ-artois.fr Fax : 03.21.79.32.40