Master CCI, promotion 2012—2013

F. ARNOULD
M. CAMARA
J. DURAND
D.PAYANT
J. PHILIBERT
C. THURIER

Projet LAPI.NET

Rapport technique

Table des matières

1 Introduction

Ce rapport a pour but de présenter le site dans sa réalisation et ses différents aspects techniques. En premier lieu, on rappellera les principaux besoins fonctionnels et non fonctionnels du site LAPI.NET. Cela nous amènera à la description des principales contraintes techniques et au choix des langages. Nous étudierons ensuite les répercutions en terme d'architecture et expliqueront ainsi que le déploiement des fichiers qui en résulte. Le mécanisme des différents composants sera ensuite détaillé et nous aborderons les aspects de sécurité. Nous terminerons par une conclusion sur les améliorations possibles.

2 Rappel sur le site et ses besoins

Le site LAPI.NET propose une interface permettant à des propriétaires de lapins de communiquer ensemble afin de se connaître et de partager leurs expériences de passionnés.

Le site se devant de fonctionner comme un « cercle privé » protégeant ses utilisateurs, il dispose d'un système d'inscription et de login / logout. Par ailleurs, certaines informations sont classées privées (e-mail, noms et prénoms réels) alors que d'autres sont publiques. Chaque utilisateur à la possibilité d'inscrire plusieurs de ses animaux sur le site. Ceux-ci deviennent affichables sur le profil qui sert de vitrine. Il est possible d'assigner un autre utilisateur comme ami, ce qui permet de ne pas avoir à le rechercher à chaque connexion et de pouvoir s'entretenir régulièrement avec lui avec facilité.

Afin de repérer des utilisateurs avec qui communiquer, le site propose un service de recherche. Différents moyens permettent ensuite aux utilisateurs de communiquer ensemble :

- un service de tchat.
- une messagerie, proche d'un service mail « local ».

Pour finir, différentes pages viennent compléter le site pour fournir les différentes informations nécessaires aux utilisateurs :

- une page « contactez-nous »
- une page « qui sommes nous »
- une page « liens externes ».

Différents besoins non fonctionnels sont pris en compte parmi lesquels :

- L'hébergement se fait sur les pages perso de free.fr ce qui entraine certaines limitations.
- La maintenance et l'évolution du site doivent être facilitées.
- La sécurité du site est gérée.
- Le site doit avoir une esthétique satisfaisante.

Note : l'administration et la modération du site sont laissé pour le moment en en back-office.

3 Limitations dues à l'hébergement chez free.fr

Le fait d'être hébergé sur les pages perso free.fr impose en tout premier lieu les langages utilisables côté serveur :

- le php version 4 pour le langage, ce qui implique quelques particularités en terme de codage.
  - Les blocs try ..catch ne fonctionnent pas.
  - Les champs de classes n'acceptent pas les descripteurs public et private.
- une base unique MySQL est disponible.

Certaines bibliothèques et fonctions php sont désactivées. Ainsi, l'absence de sockets et de possibilité de multi-thread nous ont obligé à développer un service de tchat reposant sur des appels ajax côté client et l'émulation d'un état de connexion.

De plus, les requêtes ajax doivent impérativement utiliser la méthode « get » et l'utilisation de variables de session en php nécessite la présence d'un dossier « Session ».

4 Choix des langages et des framework

En tant que site internet, le logiciel repose sur les technologies HTML 5 et CSS 2.

Côté serveur, comme vu précédemment, le langage php 4 couplé a MySQL est imposé par l'hébergement chez free.fr. Nous avons utilisé la bibliothèque GD pour le redimensionnement des images envoyé par les utilisateurs sur le serveur. Nous avons pu utiliser la programmation orientée objet, notamment pour la gestion des discussions du tchat. L'utilisation d'objets ayant un constructeur a permis de créer des objets « discussions » comprenant des utilisateurs et des messages directement triés à l'insertion, ce qui a facilité leur utilisation. Cela simplifie par exemple la comparaison entre deux discussions.

Côté client, le javascript a été choisi pour son « universalité » sur les principaux navigateurs existants. En effet, il est présent par défaut sur l'ensemble des navigateurs récents sans besoin d'installation de plugin. Ce langage est relativement portable, bien que des variations existent d'une plate-forme ou d'un navigateur à l'autre. Ces éventuelles problèmes de portabilité peuvent être contournés par l'utilisation d'une bibliothèque telle que Jquery ou Prototype. Concernant ces bibliothèques, différentes possibilités nous sont offertes parmi de nombreux frameworks :

- JQuery
- Prototype
- Script.aculo.us
- Dojo
- Yui (Yahoo)
- Mootools
- …

L'intérêt s'est rapidement focalisé sur JQuery et Prototype, les deux frameworks les plus populaires sur le net. Il a été très difficile de les départager, par manque d'expérience. La consultation de différents blogs ou forum n'a aidé en rien, les utilisateur faisant généralement un choix motivé par leurs goûts propres et chaque Api semble avoir ses avantages et ses inconvénients. Nous avons donc utilisé au départ Prototype pour le tchat en laissant la librairie JQuery disponible pour chaque développeur. La cohabitation des deux Api se fait par l'inclusion du code suivant :

chat.js :

```
/**********************************
 * assurer la compatibilite entre *
 * jquery et prototype            *
 **********************************/
// use jQuery as $j() and Prototype's $()
var $j = jQuery.noConflict();
```

L'utilisation de JQuery se fait alors via l'appel **`$j( … )`** et Prototype avec **`$( … )`**.

Néanmoins l'essentiel du code javascript présent dans le site n'utilise aucune Api particulière et est réalisé à partir des fonctionnalités standard de javascript.

5 Architecture du site et des répertoires

Au démarrage du projet nous avons fait le choix entre deux possibilités d'architecture pour le site. En effet, chaque pages affichée contient une partie commune constituée de

- la bannière,
- du menu de navigation
- d'une barre latérale qui contient différentes données liées à l'état de connexion :
  - un formulaire de login ou logout,
  - l'avertissement de nouveaux messages si l'on est connecté, *etc*.
- Un pied de page comportant quelques informations sur le site est également présent.

Il était donc possible :

- soit : d'inclure dans chaque page affichable la partie commune.
- soit : d'avoir une page unique affichable (index.php) dans laquelle nous pouvons inclure un contenu variable.

Nous avons opté pour cette dernière solution, qui nous a permis d'avoir un site visuellement très homogène en ne faisant qu'une seule inclusion par page. Le contenu « variable » est inclus dans un tag **`<section />`** dédié. La sélection se fait par un paramètre « page » passé dans l'url.

6 Déploiement

A partir de ce choix nous avons donc placé à la racine du site :

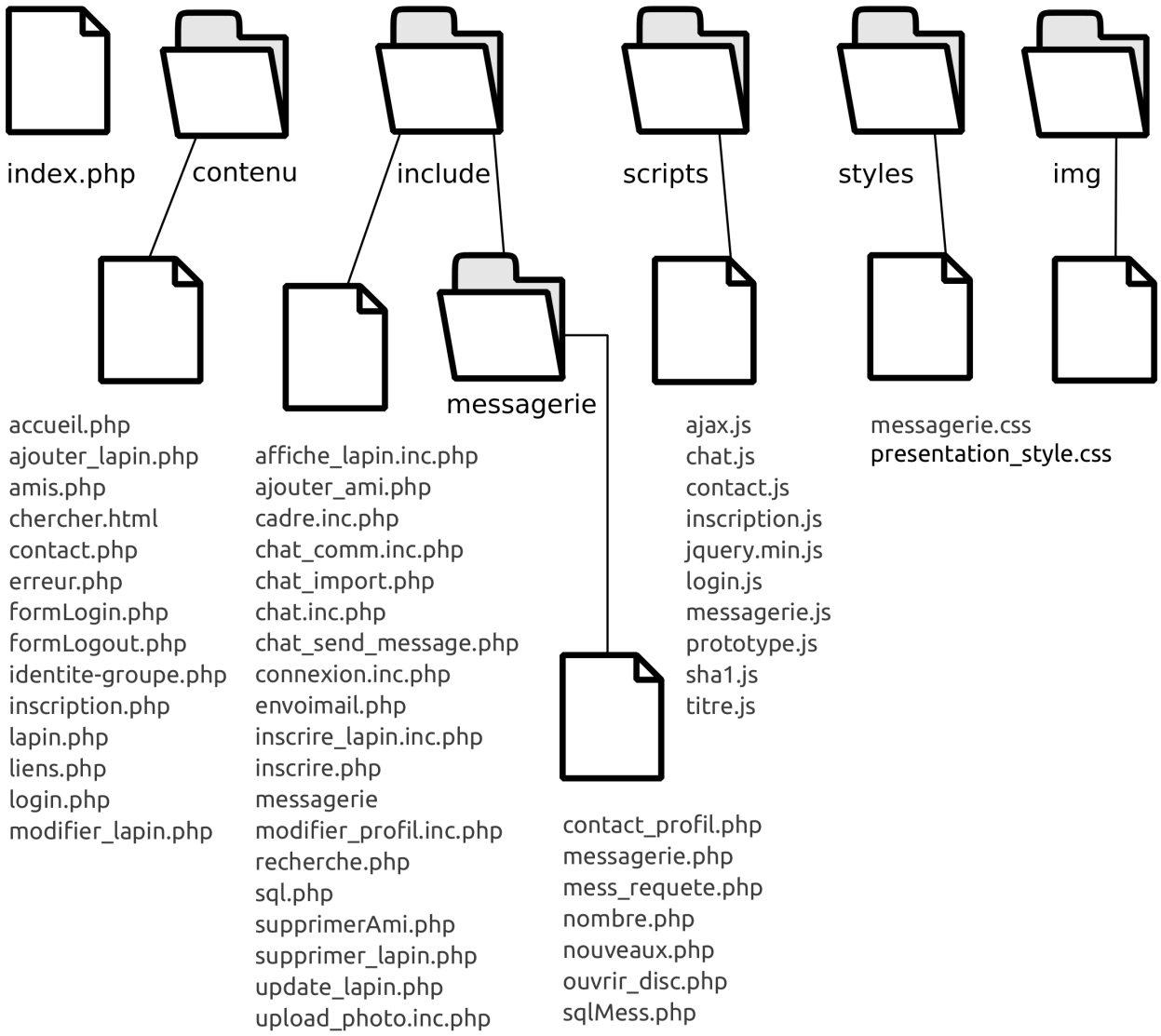
- le fichier **index.php** qui est l'interface utilisateur principale du site.
- un répertoire « **contenu** » qui contient les différents parties variables affichables (les « pages ») :
  - les formulaires d'inscriptions, de modification
  - les profils
  - le formulaires de recherche
  - les pages de contact, de liens...
- un répertoire « **include** » qui contient des scripts php appelés ou inclus :
  - le script de connexion à la base données.
  - les scripts d'insertion ou de modification de la base de données : inscription, gestion du tchat et de la messagerie...
  - des bibliothèques de fonctions : traiter l'upload d'image, afficher un profil.
- un répertoire « **scripts** » pour tous les fichiers javascripts externes.
- un répertoire « **styles** » pour les feuilles de styles externes.
- un répertoire « **img** » pour le stockage des images, ayant ses droits chmod en 777.

A noter que le répertoire « include » contient deux types de fichiers qui aurait pu être séparés. D'une part on y trouve des fichiers appelés pour réelle inclusion (par **`include`** ou **`require_once`**), d'autre part on y trouve des fichiers de scripts métier appelés souvent lors de la validation d'un formulaire, exécutant des commandes et se terminant par une redirection de type :

```
header('Location: ../index.php?page= … ' );
```

Compte-tenu du nombre de fichier, il serait intéressant de créer des sous-dossier afin de séparer les différents composants.

Le schéma de déploiement des fichiers est présenté page suivante.



7 Description des fichiers

Les fichiers sont décrits par composant.

7.1 Composant « page principale »

Décrit l'interface graphique telle que se présente à l'utilisateur.

index.php :

La page principale. Elle contient, la bannière, le menu de navigation, une interface de connexion et quelques liens. L'insertion des différentes pages de contenu s'effectue dans une section :

```
<section id="contenu">
	<?php
		include("include/cadre.inc.php");
	?>
</section>
```

La page principale contient par défaut la page d'accueil.

include/cadre.inc.php :

Script qui permet d'inclure le bon fichier php provenant du répertoire « contenu » en fonction du paramètre page récupéré para la méthode GET.

contenu/accueil.php :

La page d'acueil.

styles/presentation_style.css :

La feuille de style générale du site.

7.2 Composant « connexion »

Les utilitaires de connexion.

include/sql.php :

Les fonctions de connexion, de déconnexion à la base MySQL ainsi que quelques fonctions de requêtes.

7.3 Composant « inscription »

L'inscription du propriétaire, tout commence ici pour lui.

contenu/inscription.php :

Le formulaire d'inscription du propriétaire de lapins qui nécessite javascript – ou ne s'affiche pas sinon. Le formulaire est vérifié par javascript (cf *scripts/inscription.js*) et le mot de passe est encodé (cf *scripts/sha1.js*). En cas de succès le script *include/inscrire.php* est appelé.

scripts/inscription.js *:*

Fichier javascript externe qui permet la vérification des différents formulaires d'inscription ou de modification.

scripts/sha1.js *:*

Fichier de fonctions javascript pour l'encodage du mot de passe. Voir aussi la partie « le cryptage des mots de passe ».

include/inscrire.php :

Vérifie les données du formulaire d'inscription côté serveur et effectue l'inscription effective dans la base MySQL.

7.4 Composant « profil »

Le profil du propriétaire, modifiable.

contenu/profil.php :

L'affichage du profil d'un propriétaire. Le contenu de l'affichage est variable, en effet plusieurs cas de figure sont possibles :

- visiteur non connecté / paramètre user non renseigné : redirection vers une page d'erreur.
- visiteur non connecté / paramètre user renseigné : affichage d'un profil public (limité).
- visiteur connecté / paramètre user non renseigné : affichage du profil personnel privé, permettant des modifications.
- visiteur connecté / paramètre user = identifiant du visiteur : idem. L'identifiant est connu grâce à la variable `$_SESSION['identifiant'].`
- visiteur connecté / paramètre user renseigné : affichage du profil public avec possibilité d'envoi de messages, d'ajout aux amis.

Le profil public contient quelques informations comme la localisation et les lapins possédés. Lorsque l'utilisateur demande à modifier son profil privé, la page de l'utilisateur est rechargé avec un nouveau paramètre modifier en méthode GET. Dans ce nouveau cas, les champs modifiables font place à des zones de saisie de formulaire. La validation conduit à des vérifications javascript similaires à celles réalisées lors de l'inscription. En cas de succès, le fichier *include/ modifier_profil.inc.php* est appelé.

Cette méthode consistant à utiliser le même fichier pour l'affichage du profil et la modification génère un fichier assez peu lisible. Pour modifier les profils « lapin », nous avons fait le choix de renvoyer vers une autre page dédiée aux modifications.

include/modifier_profil.inc.php :

Effectue les modification du profil dans la base après vérifications côté serveur.

7.5 Composant « login »

une fois inscrit, l'utilisateur a la possibilité de se connecter lorsqu'il revient sur le site.

contenu/login.php :

C'est un fichier qui affiche un formulaire de login ou de logout selon l'état de connexion de l'utilisateur. Ce fichier est également rappelés par ces deux formulaires pour gérer la connexion ou la déconnexion effective par les variables de Session. Il prend notamment en charge le hashage du mot de passe lors de la connexion.

contenu/formLogin :

Le formulaire de connexion. Appelle *contenu/login.php*. Crée une variable pour le hashage du mot de passe.

contenu/formLogout :

Le formulaire de déconnexion. Appelle *contenu/login.php*.

7.6 Composant « lapin »

Une fois inscrit, le propriétaire peut présenter ses lapins à la communauté.

contenu/ajouter_lapin.php :

Le formulaire d'inscription d'un nouveau lapin. Il est validé par javascript et appelle *include/inscrire_lapin.inc.php* en cas de succès.

include/inscrire_lapin.inc.php :

Vérifie les données du formulaire d'inscription côté serveur et effectue l'inscription effective dans la base MySQL.

include/affiche_lapin.inc.php :

Contient la fonction affiche_lapin. Cette fonction permet d'afficher le profil d'un lapin sous la forme d'une **`<div class="profil_lapin" />`** après une requête sql, ex :

```
require_once("include/affiche_lapin.inc.php");
$resultat = mysql_query( "SELECT * FROM lapin_lapin WHERE ... );
while ($lapin = mysql_fetch_array($resultat) ){
      affiche_lapin( $lapin );
}
```

contenu/modifier_lapin.php :

Formulaire de modification d'un profil lapin, prérempli avec les données initiales. Appelle *include/update_lapin.php* après vérification en javascript.

include/update_lapin.php :

Effectue les modification du profil auprès de la base après vérifications côté serveur.

include /supprimer_lapin.php :

Script appelé à partir du profil propriétaire pour supprimer un lapin (de la base).

7.7 Composant « recherche par critère »

Disponible lorsque le propriétaire possède un lapin, permet de lui trouver un compagnon.

contenu/rencontre.php :

Recherche d'un partenaire selon des goûts sous la forme d'un formulaire comprenant des cases à cocher.

contenu/rencontrer.php :

Traitement de la requête de recherche d'un partenaire et affichage des résultats triés grâce au score.

7.8 Composant « recherche générique »

La recherche générique sur toute la base.

contenu/chercher.html :

Le formulaire de recherche générique. La requête est envoyée au script *include/recherche.php*.

include/recherche.php :

Sépare les critères, leur assigne un type, puis effectue une recherche pertinente multi-critères. Tous les critères doivent être vérifiés pour sélectionner une entrée. Le résultat est affiché par table SQL en tableau ou en affichant les fiches des lapins à l'aide de *include/affiche_lapin.inc.php*. Par ailleurs les propriétaires trouvés pointent vers leur fiche et les messages trouvés redirigent vers la messagerie.

7.9 Composant « tchat » ou « chat »

Discussion « live » ou presque...

include/chat.inc.php :

L'interface graphique « Lapiphone », inclus dans la page index.php.

scripts/chat.js :

Feuille de script pour le tchat avec 3 rôles :

- gérer l'affichage des différentes conversations avec les fonctions switch_conversation(nom) et afficher_conversation(conversation).
- ecouter() : faire un appel ajax à *include/chat_import.php* toutes les 10 secondes qui va récupérer les discussions concernant l'utilisateur et maintenir la présence par la mise à jour d'une date de signal sur la base.
- envoyer() : faire un appel ajax à *include/chat_send_message.php* pour envoyer un nouveau message dans la conversation courante.

include/chat_comm.inc.php :

Bibliothèque de fonctions php pour le tchat comprenant les fonctions de créations de messages, de créations de discussion, de récupération des discussions et parsing XML, de nettoyage de la base. Le nettoyage de la base se fait a chaque discussion entre deux personnes : on efface alors leur discussions précédentes. Aléatoirement, on efface également les discussions des utilisateurs non connectés. On connaît l'état de connexion via les date de session et de dernier signal envoyé.

include/chat_import.php :

Script appelé en requête ajax pour importer les messages et mettre à jour la date de dernier signal de l'utilisateur.

include/chat_send_message.php :

Script appelé en requête ajax pour envoyer un nouveau message.

7.10 Composant «messagerie»

Discussion persistante entre lapins.

include/messagerie/contact_profil.php :

Le formulaire de création d'une nouvelle discussion, disponible sur la page de profil du propriétaire de lapins par sur un lien. Le formulaire inclus la liste des lapins du propriétaire du profil et du membre connecté obtenue à l'aide de *include/sql.php*. Il envoi ses informations au script *include/messagerie/ouvrir_disc.php*.

include/messagerie/ouvrir_disc.php :

Script recevant les informations du formulaire de création d'une discussion qui se charge de contrôler la validité des données transmises avant de créer la nouvelle discussion dans la base sql et de retourner à la page d'origine. Il n'utilise que la bibliothèque SQL *include/sql.php*.

include/messagerie/messagerie.php :

C'est le moyeu de la messagerie autour duquel s'agencent les autres scripts. Dans un premier temps il affiche seulement un résumé des discussions. L'interaction avec la liste est alors gérée par javascript *script/messagerie.js* à l'aide de *script/ajax.js*. Les requêtes SQL sont centralisées dans *include/sqlMess.php.* La mise en forme est assurée par le style *style/messagerie.css*.

script/messagerie.js :

Gestionnaire, via *script/ajax.js*, de l'échange d'informations avec le serveur sans mettre à jour la page de messagerie. Lors du développement d'une discussion, de la sélection d'un message, de la décision de répondre ou de l'envoi de la réponse il contacte le serveur qui traite la demande par le script *include/messagerie/mess_requete.php* et ne retourne que ce qui est nécessaire à chaque étape. Il traite la réponse de façon à l'afficher à l'endroit voulu ou d'afficher un message d'erreur.

include/messagerie/mess_requete.php :

Il s'occupe de gérer les demandes envoyées par *messagerie.php*. Selon les paramètres fournis il renvoie la liste des messages dans la discussion, le corps d'un message en particulier, le formulaire de rédaction d'un message ou valide le bon envoi d'un nouveau message. Cette réponse est traitée côté client par *script/messagerie.js*. Les requêtes SQL sont centralisées dans *include/sqlMess.php.* La mise en forme est assurée par le style *style/messagerie.css*.

***scripts/ajax.js** :*

Script javascript qui regroupe les fonctions de communication http avec le serveur.

***include/messagerie/nouveau.php** :*

Averti le membre de la réception de nouveaux messages en affichant leur nombre dans la colonne de droite de la page. La consultation de la base ce fait à intervalles réguliers. La connexion http se fait par *script/ajax.js*.

include/messagerie/nombre.php :

Régulièrement consulté par *include/messagerie/nouveau.php* il relève le nombre de messages plus récents que le dernier accès à la messagerie du membre connecté. L'ouverture de la messagerie remet ce compte à zéro même si les messages ne sont pas lus. Les requêtes SQL sont centralisées dans *include/sqlMess.php* et la connexion à la base se fait par *include/sql.php*.

7.11 Composant « amis »

Lorsque l'utilisateur arrive sur un profil, la possibilité est offerte de l'enregistrer comme ami afin d'entretenir une relation suivie.

contenu/amis.php :

La page qui affiche les différents amis de l'utilisateurs.

include/ajouter_ami.php :

Script appelé à partir d'un profil pour enregistrer quelqu'un comme ami.

include/supprimerAmi.php :

Script appelé à partir de la page « amis » pour supprimer un ami.

7.12 Autres composants

contenu/contact.php :

Une page pour envoyer un mail à un administrateur du site. Appelle include/envoimail.php.

include/envoimail.php :

L'envoi effectif du mail à administrateur.

contenu/erreur.php :

Une page d'erreur.

contenu/identite-groupe.php :

La page « qui sommes nous ».

contenu/liens :

Une page de liens externes, sur le thème de la cuniculture.

include/upload_photo.inc.php :

Fonction qui gère le stockage des images sur le site. Les images jpg, png ou gif sont automatiquement redimensionnées au besoin. Utilise la bibliothèque GD.

8 La sécurité

La sécurité est une caractéristique importante pour une application web. Pour notre part, nous avons tenté de protéger au mieux le site en faisant avec les contraintes imposées par l'hébergement.

8.1 Les failles d'injection SQL

Les champs des formulaires sont vérifier côté serveurs. Les chaînes de caractères sont transformées par des fonctions pour éviter l'insertion de script malveillant.

include/inscrire.php :

```
$user = mysql_real_escape_string($_POST['user']);
$nom = mysql_real_escape_string($_POST['nom']);
$prenom = mysql_real_escape_string($_POST['prenom']);
$password = mysql_real_escape_string($_POST['pass']);
...
```

8.2 les failles d'upload

Le site permet l'upload d'images pour les profil propriétaire et lapins. Afin d'éviter des attaques par exécution de code malveillant en contrôlant l'extension du fichier et si le nom de fichier ne contient pas de caractères spéciaux.

include/upload_photo.inc.php :

```
//erreur d'upload
if( $fichier['error'] > 0 ) return false;
      //pb probable : taille de fichier accepte par php
//erreur d'extensions
$extensions_valides =  array('jpg', 'jpeg', 'gif',  'png' );
$extension_upload =  strtolower( substr( strrchr( $fichier['name'], '.') ,1) );
if( ! in_array( $extension_upload, $extensions_valides) ) return false;
// vérifie qu'il n'y a pas de caractères spéciaux
if(preg_match('#[\x00-\x1F\x7F-\x9F/\\\\]#', $fichier['name'])) return false ;
```

Le type mime de l'image est ensuite utilisé.

8.3 Les failles d'include

Notre site fonctionne grâce à l'inclusion de contenu HTML dans la page index.php. Néanmoins l'inclusion est sécurisée, puisque c'est un mot clé et non un nom de fichier qui définit l'inclusion.

include/cadre.inc.php :

```
// index.php?page='inscription'
if ($_GET['page']=='inscription') {
      include("contenu/inscription.php");
      $redirige = true;
}
...
```

8.4 Les usurpations de session

Afin de diminuer le risque d' usurpations de session de nos utilisateurs, les sessions sont régénérées avec al fonction :

```
session_regenerate_id() ;
```

lors de l'inscription et du login.

8.5 Le cryptage des mots de passe

La plupart des personnes utilisent un même mot de passe pour tous leur site.

Le hashage (en sha1) du mot de passe ne vise pas tant à sécuriser l'identification des utilisateurs sur notre site, qu'à les protéger de très graves actes de malveillances sur des sites plus sensibles que le notre (banque, autres réseaux sociaux,...).

Lors d'une connexion, seuls circulent le pseudo et le hashage du mot de passe ; mais jamais le mot de passe. Ce mot de passe n'est donc pas enregistré dans nos bases de données.

1) Un premier hashage initial, qui ne circule qu'une seule fois et au moment de la création du compte, est calculé par le poste client puis envoyé au serveur.
2) Pour les connexions suivantes, une seconde clé est calculée selon le principe suivant :

– le serveur tire au sort et envoie un numéro ;

– le client calcule un nouvelle clé en hashant la concaténation de la clé initiale et de ce numéro aléatoire ;

– le serveur compare la nouvelle clé postée avec le résultat attendu pour identifier l'utilisateur.

La première clé, si elle est interceptée, peut suffire à violer le compte sur notre site. Par contre elle n'est à priori d'aucune utilité pour pirater les accès d'un utilisateur à d'autres sites que le notre.

Les clés secondaires qui circulent par la suite ne peuvent pas être réutilisées (à moins d'un coup du sort suffisamment improbable pour se satisfaire d'un simple tirage aléatoire du numéro).

De plus, les identifiants de session sont réinitialisés à chaque identification afin de se prémunir d'une technique de vol de session assez classique.

Cette technique ne protège cependant pas complètement nos visiteurs (mot de passe trop facile à deviner, logiciel espion qui « écouterait » le clavier, etc).

9 Conclusion

Nous avons réussi à développé **un site fonctionnel qui répond aux principaux besoins**. De nombreux points pourraient être amélioré concernant la mise en forme des pages via CSS et l'ergonomie du site. Faute de temps, nous n'avons pas réalisé la partie d'administration du site. Concernant l'architecture, arrivé en fin de développement, nous nous sommes rendu compte que l'architecture de déploiement du site devient insuffisamment bien fractionnée. Le dossier include, par exemple, comprend beaucoup trop de fichiers (21). Il serait donc intéressant de séparer mieux les différents composants (c'est le cas pour la messagerie) et de renommer les fichiers aux nom trop ambigus (*sql.php*, *rencontre.php*, *nombre.php...*).

La réalisation du site LAPI.NET a été une belle expérience collective qui a été pour la plupart d'entre-nous la première expérience de gestion de projet de développement. Cela a été également l'occasion de rentrer plus en profondeur dans les différents langages abordés en cours.