

Le serveur web du centre NTE : nouveau dispositif

Rapport de stage et documentation du dispositif

Olivier Clavel

Diplome staf - TECFA

Table des matières

I	Introduction et informations préliminaires	1
A	Présentation du centre NTE et du stage.....	1
	Le centre NTE et ses missions.....	1
	Coordonnées du centre	1
	Collaborateurs permanents du centre.....	1
B	Le stage.....	2
II	Cahier des charges du dispositif.	2
A	Description des différentes sections publiques du site	2
	Composition.....	2
	4
	Accueil.....	4
	Réalisations.....	4
	En cours	4
	Ressources	5
	Les publications	5
B	Autres aspects du site réservés aux administrateurs.....	5
	Emploi du temps administrateur.	5
	Projets complet en Intranet.	5
III	Matériel et logiciels requis.....	5
IV	Installation (quick-start).....	6
V	Structure du système de fichier et fonctions des fichiers/répertoires.....	7
A	Répertoire de base (centre_NTE).....	7
B	Sous répertoire <i>db</i> : base de donnée	9
C	Sous répertoire <i>files</i> : fichiers constituant les pages du site (référéncés dans la base de donnée) ⁹	
D	Sous répertoire <i>images</i> : les images utilisées dans le site.....	9
E	Sous répertoire <i>lib</i> : les librairies de scripts (php, javascript, html) utilisées.	9
F	Sous répertoire <i>pimages</i> : les photos des participants à des projets	10
G	Sous répertoire <i>projects</i> : les rapports html des projets	10
H	Sous répertoire <i>projects_doc</i> : les rapports de projet sous forme téléchargeable	10
VI	Structure de la base de donnée.....	11
A	Schéma récapitulatif de toutes les tables et de leurs relations.....	11
B	table <i>admin</i> (administrateur du site).....	11
C	table <i>categories</i> (catégories des projets).....	11
D	table <i>conferences</i> (donnée sur les conférences).....	11
E	table <i>news</i> (données des nouvelles affichées sur la première page du site).....	12
F	table <i>pieces</i> (entrées des sous-menus du site).....	12
G	table <i>projets</i> (information sur les projets).....	12
H	table <i>projets_categories</i> (table relationnelle <i>projets</i> <-> <i>categories</i>)	13
I	table <i>projets_responsables</i> (table relationnelle <i>projets</i> <-> <i>responsables</i>).....	13
J	table <i>responsables</i> (responsables des projets)	13
K	table <i>Schedule</i> (agenda des administrateurs).....	13
L	tables <i>Schedule98</i> et <i>Schedule99</i>	14
M	table <i>Seminaires</i>	14
N	table <i>zones</i> (entrées principales des menus du site).....	14
O	table <i>zones_pièces</i> (table relationnelle <i>zones</i> <-> <i>pièces</i>)	14
VII	Structure d'appel des librairies	15

VIII	Documentation des fonctions et paramètres	16
A	paramètres du site : <i>lib\parameters.lib</i>	16
	Paramètres généraux	16
	Paramètres pour l'envoi de mail	16
	Couleurs	17
B	Fonctions usuelles : <i>lib\usual_fonctions.lib</i>	18
C	Fonctions de courrier électronique : <i>lib\smtp.lib</i>	20
D	Fonctions d'administration	20
	administration de l'agenda : <i>lib\agenda.lib</i>	20
	administration des conférences : <i>lib\conferences.lib</i>	22
	Administration des menus et soue-menus : <i>lib\menus.lib</i>	23
	Administration des news : <i>lib\news.lib</i>	25
	Administration des projets : <i>lib\projets.lib</i>	26
	Administration des séminaires : <i>lib\sminaires.lib</i>	29
IX	Conclusions.....	30
A	Travail sur le projet.....	30
B	Futurs développements	31

I Introduction et informations préliminaires

A Présentation du centre NTE et du stage

Le centre NTE et ses missions

Voici la description brève donnée sur le site internet du centre (<http://www.unifr.ch/nte/welcome.html>).

“Le Centre "Nouvelles Technologies et Enseignement" (NTE), créé par le Rectorat de l'Université de Fribourg, a pour mission de favoriser le développement de nouveaux supports et moyens de communication pédagogiques pour les études universitaires. Les travaux engagés s'inscrivent ainsi dans la recherche prospective que conduisent actuellement les Hautes Ecoles sur la manière notamment de faire face à l'augmentation des effectifs des étudiants. “

A ce titre, le centre NTE agit comme un centre de ressources pour les enseignants de l'université en offrant un certain nombre de services :

- Ateliers de formations
- Colloques
- Forums en ligne.
- Participation à des projets de développement (au niveau financier, technique et conceptuel)

Les différents projets soutenus font l'objet de descriptions et de publications mises en ligne sur le site internet du centre.

Coordonnées du centre

Centre NTE (Nouvelles Technologies et Enseignement)
Université de Fribourg
Rue Faucigny 2
CH-1700 Fribourg (Switzerland)

email: NTE@unifr.ch
Tél.: +41 26 300 8334 ou 8349
Fax: +41 26 300 9726
Web: <http://www.unifr.ch/nte>

Collaborateurs permanents du centre

- Responsable informatique : Gérald Collaud
- Responsable pédagogique : Jean-François Perret
- Collaborateurs associés : Jacques Monnard et Rolf Brugger

B Le stage

J'ai été contacté par Gérald Collaud en juin 1999 lors de la fin de ma première année de diplôme. Dans le cadre de ses activités, le centre cherchait à se doter de nouveau moyen d'administration pour l'information contenue sur le site web. Suite à plusieurs réunions, nous avons, avec le reste des collaborateurs intéressés, défini un cahier des charges pour ce nouveau dispositif.

Lors de la réflexion, un point a particulièrement été mis en avant. Ce dispositif devait pouvoir être complété par des programmeurs (modification des fonctionnalités existantes, ajout de nouvelles fonctionnalités). C'est pourquoi, après une brève description du cahier des charges, l'accent a été mis sur la documentation exhaustive du système qui constitue la partie la plus fournie de ce rapport.

II Cahier des charges du dispositif.

Voici le cahier des charges tel qu'il a été défini après une réunion à Fribourg le 27/10/1999 en présence des collaborateurs intéressés :

Le site reprendra au maximum les unités graphiques du site déjà existant : couleurs, disposition de l'écran (en mettant le menu à gauche au lieu de la droite), polices, noms pour ne pas dérouter les utilisateurs. Il n'utilisera pas les frames pour que les pages puissent être bookmarquées dans la mesure du possible.

Le site sera programmé en PHP avec liaison vers une base de donnée Access par ODBC, le but étant de simplifier sa maintenance pour les administrateurs en passant par des formulaires de saisie pour créer, ajouter, modifier des ressources. C'est sur ce premier point qu'il faut mettre l'accent pour que, quel que soit l'état de finition du dispositif à la fin du stage, le travail puisse être continué par l'équipe NTE.

Le site est public à l'exception d'une interface administration qui permet de modifier les ressources. Cette interface est disponible sur chaque page après que l'administrateur se soit authentifié par l'intermédiaire d'un formulaire qui dépose un cookie sur son disque dur.

A Description des différentes sections publiques du site

Composition

L'utilisateur arrive sur une page de garde. Le site est ensuite décomposé en plusieurs catégories. Elles seront décrites plus en détails dans les paragraphes suivants.

- Accueil : description du centre NTE, de ses actions et de ses missions, carte d'accès, description du site et des catégories, aide à la navigation.
- Réalisations : ensemble des projets finis.
- En cours : ensemble des projets en cours qui peuvent être ensuite archivés dans la section réalisation.

- Ressources : ensemble des outils et de la documentation accessible par les professeurs. Cette section peut être naviguée par outils ou par usage.
- Publications : les publications du centre NTE, commande de rapports, actes de séminaires/ conférences internationales, mailing listes.

Dénomination du centre	Logo
Page en consultation	Menu de la section en consultation
	Menu complet du site
	Contact Recherche Map

FIGURE 1. Eléments communs à toutes les pages

- Le logo NTE identifie le site.
- La partie menu se trouve à gauche. Dans chaque section, on ajoute un menu de la section en cours
- Un lien « contact » envoie vers un formulaire de saisie pour envoyer des commentaires/ remarques/ demandes.
- Un lien « recherche » permet de rechercher des informations par mots clefs avec le moteur Infoseek.
- Un lien Map permet de visualiser graphiquement l'endroit où l'on se trouve.



FIGURE 2. Page de garde

La zone « annonces » donne des nouvelles entrées par les administrateurs à l’aide d’un formulaire. Les annonces peuvent avoir un affichage limité dans le temps.

L’interface administrateur permet de créer de nouvelles entrées dans la zone de menu et d’entrer de nouvelles annonces.

Accueil

L’accueil permet aux nouveaux venus de s’informer sur le centre NTE et sur son site :

- Description détaillée du centre et de ses missions.
- Personnel avec adresses mail
- Adresse physique et plan d’accès
- Description du site et aide à la navigation.

Réalisations

Cette section contient tous les projets réalisés par le centre. Ils sont entrés par les administrateurs à l’aide d’un formulaire. Ils comportent tous une en-tête type (titre, responsable, date...) et un fichier html qui décrit le projet. Ce dernier est stocké dans l’arborescence du serveur et son adresse est mémorisée dans la base de donnée.

Un lien permet de télécharger le projet dans un format imprimable (word , PDF...) et éventuellement de commander le rapport imprimé s’il existe.

Les administrateurs peuvent également décider d’archiver de « vieux » projets pour qu’ils n’apparaissent plus dans la liste.

En cours

Cette section contient les projets en cours déclarés par les administrateurs sur la même base que les réalisations (en-tête type, etc.). ils peuvent être mis à jour et basculés vers la zone réalisation quand ils sont finis.

On trouve également sur cette page un lien vers des projets suisses nationaux

Ressources

Les utilisateurs trouvent ici toutes les ressources qui sont mises à leur disposition pour leurs enseignements et leurs projets. Ils peuvent choisir de naviguer à travers ces ressources avec une approche orientée « outils » ou « usages ». On y trouve :

- La documentation pédagogique
- Les FAQ
- Les forums (comment utiliser)
- Les conseils
- Les supports de cours
- Les mailing listes

L'interface administrateur permet d'ajouter de nouvelles ressources.

Les publications

Dans cette section, l'utilisateur trouve les publications du centre autres que celles liées aux projets.

- Rapports avec possibilité de commande
- Participation à des séminaires, conférences internationales. Actes de colloques.

B Autres aspects du site réservés aux administrateurs

Emploi du temps administrateur.

Chaque administrateur a la possibilité de maintenir à jour son emploi du temps consultable par les autres administrateurs. Le lien vers cette fonction n'apparaît que dans l'interface administrateur.

Projets complet en Intranet.

La liste complète des projets avec tous les détails est consultable en Intranet.

III Matériel et logiciels requis

Pour pouvoir installer ce dispositif, vous devez disposer :

- d'un ordinateur avec un serveur web installé et en fonctionnement (apache, IIS, PWS... sous linux, unix ou windows)
- du langage de script php (version 3 ou supérieure - version 4 vivement recommandée) correctement installé et configuré. Entre autre, l'installation doit supporter les

fonctionnalités ODBC de php et reconnaître l'extension de fichier *.php pour l'exécution des scripts (voir <http://www.php.net> pour la documentation de php)

- d'une couche odbc permettant la liaison à la base de donnée Microsoft Access (native sur Windows, à installer sur Unix ou Linux).
- de microsoft access 97 ou supérieur pour éditer la base de donnée manuellement (notamment pour ajouter des logins et mots de passe pour de nouveaux administrateurs). Cet outil est indispensable pour continuer le développement du site et modifier la base de données (nouvelles tables, nouveaux champs...)

Remarque : la base de donnée actuelle est au format access 97. Le format a évolué dans la version suivante (access 2000). L'auteur déconseille formellement de convertir la base de donnée au nouveau format sans avoir fait des test sérieux de compatibilité au préalable.

Le dispositif ayant été développé sur un serveur NT4Sp6 disposant de IIS3 et de php3, c'est pour cet environnement que sont données les instructions d'installation ci-après.

IV Installation (quick-start)

Ces directives s'inspirent de l'installation préalable qui a été faite du site par l'auteur sur le serveur du centre NTE au début de l'année 2001. Pour les détails sur le fonctionnement du dispositif, voir les chapitres suivants.

- Décompresser le fichier *centre_nte.tar.gz* sur le disque dur du serveur (à l'aide de winzip par exemple). Tous les fichiers décompressés se trouvent dans le répertoire *centre_NTE*. Déposez ces fichiers où bon vous semble sur le disque selon l'arborescence de votre système.
- [Optionel] : Pour des raisons de sécurité, vous pouvez déplacer le répertoire *db* hors du répertoire *centre_NTE* de manière à ce que ce répertoire ne soit pas visible depuis le web.
- Configurez IIS pour qu'il puisse voir le répertoire *centre_NTE*. Vous pouvez au choix :
 - Mettre le répertoire dans votre arbre web actuel
 - Créer un répertoire virtuel
 - Créer un nouveau serveur virtuelDans tous les cas, vous devez :
 - Rendre ce répertoire exécutable pour IIS (pour l'exécution des scripts PHP)
 - Donner l'accès en écriture à l'utilisateur par défaut du serveur web sur votre machine (pour l'écriture des fichiers en mode administration).
 - Configurer IIS pour qu'il accepte *welcome.php* comme page par défaut possible.
 - Configurer un filtre ISAPI pour que les pages *.php soit exécutées par php
- Configurer un DSN (Data Source Name) système pour la base de donnée access à l'aide du panneau de configuration/ODBC
Driver : microsoft access database (*.mdb)

Nom du DSN : nte_services (Peut être changé sous réserve d'éditer les paramètres *\$database* dans le fichier *centre_NTE/lib/parameters.lib* - voir chapitre VIII-A pour les détails des paramètres)

Source de données : <install-dir>\centre_NTE\db\service_nte.mdb (ou chemin spécifique si vous avez déplacé ce fichier lors de l'installation)

- Editer le fichier *centre_NTE/lib/parameters.lib* et changez les paramètres suivants (voir chapitre VIII-A pour les détails des paramètres) :
 - *\$home_page* = <chemin absolue de la page de démarrage>
par exemple, si l'adresse est <http://monserv.com/new-serv/welcome.php>, donnez comme chemin */new-serv/welcome.php*
 - *\$smtp_host* = “<nom de votre serveur smtp>”
 - *\$local_host* = “<nom de votre serveur e.g. monserveur.domain.ch>”
 - *\$sender_address* = “<adresse email affichée dans le “reply-to” des mails envoyés par le serveur>”
 - *\$comment_address*, *\$order_address*, *\$subscribe_address* = “<adresse(s) email(s) où sont envoyés les mails respectivement pour les commentaires sur le site, les commandes de rapports, les inscriptions à des séminaires/ conférences>”
- Chargez la page de démarrage du dispositif dans un navigateur et vérifiez que tout fonctionne.

V Structure du système de fichier et fonctions des fichiers/ répertoires.

La totalité des fichiers requis pour l'installation et la mise en route se trouvent dans le fichier compressé *centre_nte.tar.gz*.

Une fois décompressés, les fichiers se trouvent dans un répertoire *centre_NTE*.

A Répertoire de base (centre_NTE\)

→ *administer.php*

fichier d'administration du site, il regroupe toutes les fonctions d'administration et fait appel à plusieurs librairies.

→ *check_login.php*

fichier qui vérifie le login administrateur et crée un cookie en cas de succès.

→ *comment.php*

formulaire pour envoyer un commentaire sur le site. Ce fichier trace l'endroit où l'on se trouve dans le site au moment où l'on désire envoyer un commentaire.

→ ***conferences.php***

fichier utilisé pour l'affichage des conférences. Ce fichier est prévu pour être inclus par un/des fichier(s) du répertoire files\ contenant seulement un query à envoyer à la base de donnée.

→ ***display-participant.php***

fichier pour l'affichage des données d'un participant à un projet.

→ ***display_report.php***

fichier pour l'affichage du rapport html d'un projet.

→ ***kill_cookie.php***

appelé lorsque un administrateur désire repasser en vue utilisateur. Ce fichier efface simplement le cookie créé par check-login.php

→ ***login.php***

le fichier contenant le formulaire pour l'authentification des administrateurs du site. Le résultat du formulaire est envoyé à check-login.php

→ ***old_news.php***

fichier pour afficher les news qui ne sont plus affichées sur la home page car trop vieilles.

→ ***order_report.php***

fichier d'affichage du formulaire pour commander un rapport de projet.

→ ***projects.php***

fichier utilisé pour l'affichage des projets. Ce fichier est prévu pour être inclus par un/des fichier(s) du répertoire files\ contenant uniquement un query à envoyer à la base de donnée.

→ ***seminaires.php***

fichier utilisé pour l'affichage des séminaires. Ce fichier est prévu pour être inclus par un/des fichier(s) du répertoire files\ contenant seulement un query à envoyer à la base de donnée.

→ ***subscribe_sem.php***

fichier d'affichage du formulaire pour s'inscrire à une conférence.

→ ***welcome.php***

fichier de base d'affichage du site. C'est lui qui est responsable de l'affichage et de l'appel des librairies pour toute la navigation au niveau utilisateur.

B Sous répertoire *db* : base de donnée

→ *service_nte.mdb*

le fichier access 97 contenant toute la base de donnée du site.

Ce répertoire peut-être déplacé hors de l'arbre web pour sécuriser la base de donnée et éviter les accès depuis un navigateur.

C Sous répertoire *files* : fichiers constituant les pages du site (référéncés dans la base de donnée)

Les fichiers présents dans ce répertoire ont les noms qui leur ont été donnés lors de l'édition ou de la création d'une zone par un administrateur. Bien que les fichiers aient tous pour la plupart l'extention .html, ils peuvent contenir du texte seul voire des scripts. Les fichiers contenant du html doivent contenir uniquement le corps du fichier (sans les tags <body> et </body>)

D Sous répertoire *images* : les images utilisées dans le site

Toutes les images, logos, boutons... utilisées dans les pages (à l'exception des photos des membres participants à un projet). On trouve notamment dans ce dossier les fichiers:

→ *blank_button.psp*

fichier original au format Paint Shop Pro 5 d'un bouton vide de la barre des fonctions globales d'administration.

→ *blank_button2.psp*

fichier original au format Paint Shop Pro 5 d'un bouton vide de la barre des fonctions contextuelle d'administration.

E Sous répertoire *lib* : les librairies de scripts (php, javascript, html) utilisées.

→ *agenda.lib*

fichier regroupant les fonctions administrateur pour la gestion de l'agenda.

→ *conferences.lib*

fichier regroupant les fonctions administrateur pour la gestion des conférences.

→ *js_delconf.li*

petite fonction javascript qui est incluse dans le fichier de gestion des conférences pour confirmer la suppression d'une conférence.

→ ***menus.lib***

fichier regroupant les fonctions d'administration pour la gestion des menus et des sous menus du site.

→ ***news.lib***

fichier regroupant les fonctions d'administration pour la gestions des nouvelles.

→ ***parameters.lib***

fichier permettant d'initialiser un certain nombre de variables utiles pour la gestion du site (nom du DSN, page d'accueil, couleurs, etc...). Une description plus complète des variables est donnée à la section "Documentation des fonctions et variables". Ce fichier est inclus dans toutes les pages du site pour permettre l'utilisation de ces variables dans le programme et les fonctions.

→ ***projets.lib***

fichier regroupant les fonctions d'administration pour la gestion des projets.

→ ***seminaire.lib***

fichier regroupant les fonctions d'administration pour la gestion des séminaires.

→ ***smtp.lib***

fichier contenant une classe php permettant d'envoyer des mail par smtp depuis php. Utilisé par les formulaires et autres pages qui envoient des courriers électroniques.

→ ***usual_fonctions.lib***

fichier regroupant toutes les fonctions utilisées pour l'affichage du site en mode utilisateur.

F Sous répertoire *pimages*\ : les photos des participants à des projets

Les photos doivent préalablement être déposées dans ce dossier au format gif, jpeg ou png pour pouvoir être ensuite associées à des personnes dans l'interface administrateur.

G Sous répertoire *projects*\ : les rapports html des projets

Ce répertoire fonctionne comme le répertoire *files*\. Les fichiers contiennent le corps du fichier html uniquement (sans les tags <body> et </body>)

H Sous répertoire *projects_doc*\ : les rapports de projet sous forme téléchargeable

Les fichiers (word, pdf, txt) doivent être enregistrés dans ce répertoire avant de pouvoir les référencer dans la base de donnée.

VI Structure de la base de donnée

Toutes les informations pour la génération des pages du site sont contenues dans la base de donnée service_nte.mdb. C'est une base de donnée microsoft access 97. Voici un schéma récapitulatif de la base de données dans son ensemble ainsi que le détails des champs pour chaque table.

A Schéma récapitulatif de toutes les tables et de leurs relations.

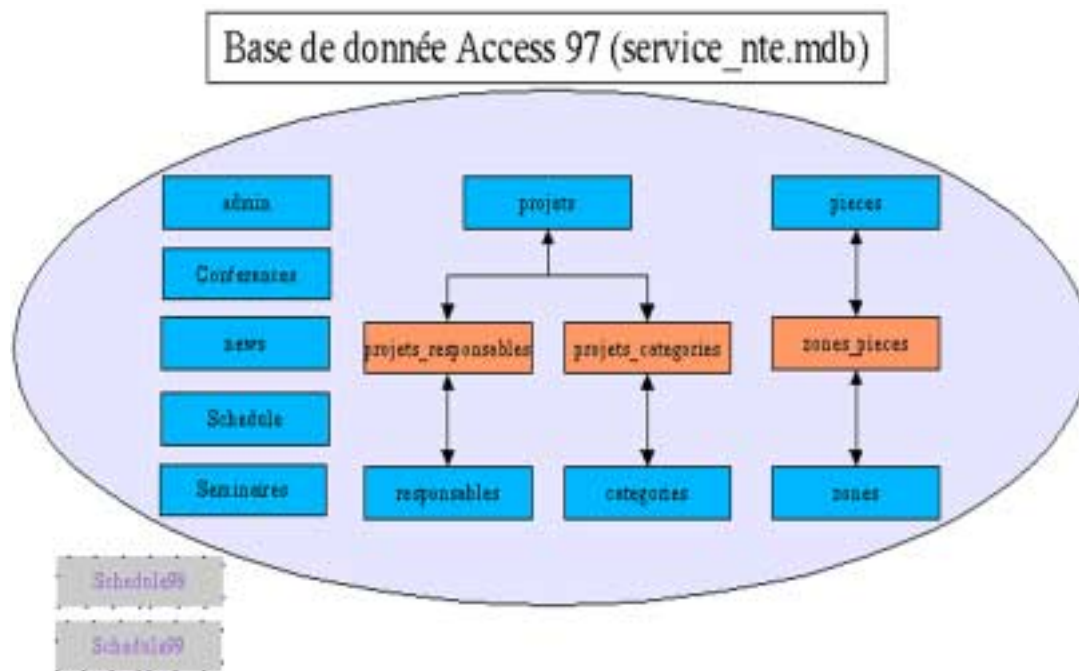


FIGURE 3. Schéma de la base de donnée

B table *admin* (administrateur du site)

- *id* : numéro automatique
- *login* : text
- *password* : text
- *real_name* : text

C table *categories* (catégories des projets)

- *id* : numéro automatique
- *categorie* : text

D table *conferences* (donnée sur les conférences)

- *id* : numéro automatique
- *CreationDate* : date jj/mm/aaaa
- *NET* : oui/non

- *NTE* : oui/non
- *Title* : text
- *Description* : text
- *StartDate* : date jj/mm/aaaa
- *EndDate* : date jj/mm/aaaa
- *Location* : text
- *Country* : text
- *URL* : text (url du site de la conférence)
- *EMail* : text

E table *news* (données des nouvelles affichées sur la première page du site)

- *news_id* : numéro automatique
- *title* : text
- *author* : text (tiré du champ real name de la table admin, liaison logicielle par php)
- *content* : Memo
- *creation_date* : date jj/mm/aaaa hh:mm:ss
- *valid_start* : date jj/mm/aaaa (date de début de validité de la news)
- *valid_end* : date jj/mm/aaaa (date de fin de validité de la news)

F table *pieces* (entrées des sous-menus du site)

- *id* : numéro automatique
- *nom* : text
- *creation_date* : date jj/mm/aaaa hh:mm:ss (date de création du menu)
- *update_date* : date jj/mm/aaaa hh:mm:ss (date de mise à jour du menu)
- *pointer* : text (chemin relatif du fichier à afficher par rapport au répertoire de base du serveur : centre_NTE\)

G table *projets* (information sur les projets)

- *id* : numéro automatique
- *titre* : text
- *description* : memo
- *rapport* : text (nom du fichier contenant le rapport html dans le répertoire centre_NTE\projects\)
- *en_cours* : numérique (0 = non, 1 = oui - possibilité de rajouter d'autres états)
- *début* : date jj/mm/aaaa (date de début du projet)
- *fin* : date jj/mm/aaaa (date de fin du projet)

- *parution* : nombre (date aaaa - gardé par compatibilité avec l'ancien système de classement - année de parution du projet - pas utilisé)
- *rapport_doc* : text (nom du fichier dans le répertoire centre_NTE\projects_doc\ contenant le rapport téléchargeable .doc, .pdf ...)
- *rapport_comm* : nombre (possibilité de commander le rapport. 0 = non, 1 = oui. On peut rajouter d'autres états)

H table *projets_categories* (table relationnelle *projets* <-> *catégories*)

- *id* : numéro automatique
- *projet* : nombre (id du projet dans la table projets)
- *categorie* : nombre (id de la catégorie dans la table categories)

I table *projets_responsables* (table relationnelle *projets* <-> *responsables*)

- *id* : numéro automatique
- *projet* : numéro (id du projet dans la table projets)
- *responsable* : numéro (id du responsable dans la table responsable)

J table *responsables* (responsables des projets)

- *id* : numéro automatique
- *titre* : text (M., Mme, Melle, Dr, etc.)
- *prenom* : text
- *nom* : text
- *adresse_1* : text
- *adresse_2* : text
- *code_postal* : text
- *ville* : text
- *pays* : text
- *tel* : text
- *fax* : text
- *email* : text
- *photo* : text (nom du fichier image contenant la photo d'identité dans le répertoire centre_nte\pimages\)
- *commentaire* : text

K table *Schedule* (agenda des administrateurs)

- *id* : numéro automatique
- *CreationDate* : date jj/mm/aaaa (date de création de l'entrée dans l'agenda)

- *EventDate* : date jj/mm/aaaa (date de l'évènement)
- *EventTime* : date hh:mm:ss (heure de l'évènement)
- *Location* : text (endroit où se déroule l'évènement)
- *Description* : text
- *GC* : oui/non
- *JFP* : oui/non
- *JM* : oui/non
- *RB* : oui/non
- *JP* : oui/non
- *JLG* : oui/non

L tables *Schedule98* et *Schedule99*

Ce sont les anciens agendas originaux envoyés par les collaborateurs du centre NTE. Gardés pour des raisons d'archivage et de sauvegarde, ils ne sont pas utilisés par l'application. Tous les évènements sont gérés dans la table *Schedule*.

M table *Seminaires*

- *id* : numéro automatique
- *titre* : text
- *date_sem* : date jj/mm/aaaa (date du séminaire)
- *resume* : mémo (résumé du séminaire)

N table *zones* (entrées principales des menus du site)

- *id* : numéro automatique
- *nom* : text (nom apparaissant dans le menu)
- *creation_date* : date jj/mm/aaaa hh:mm:ss
- *update_date* : date jj/mm/aaaa hh:mm:ss
- *pointer* : text (chemin relatif du fichier à afficher pour cette entrée de menu par rapport au répertoire centre_NTE\)

O table *zones_pièces* (table relationnelle *zones* <-> *pièces*)

- *id* : numéro automatique
- *zone* : numéro (id de la zone dans la table *zones*)
- *piece* : numéro (id de la pièce dans la table *pieces*)

VII Structure d'appel des librairies

- Une des difficultés principales de la compréhension du fonctionnement du site réside dans les appels successifs qui sont faits page après page à des librairies de fonctions. Nous présentons dans la figure Figure 4 on page 15, pour chaque] fichier important du site, un schéma des appels de librairies de fonctions/paramètres par les pages du site.

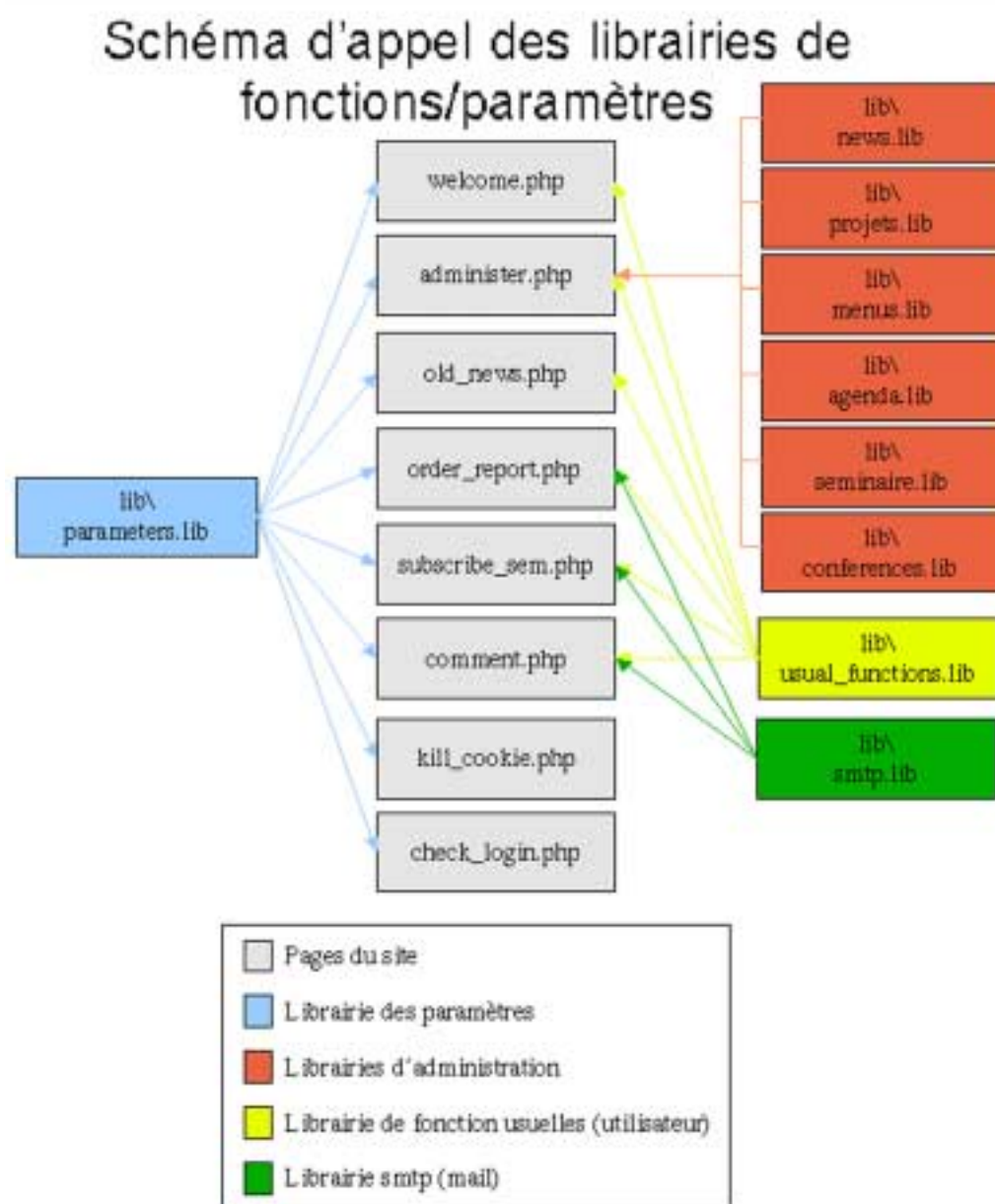


FIGURE 4. Schéma d'appel des librairies

VIII Documentation des fonctions et paramètres

Ce chapitre présente en détail toutes les fonctions et paramètres des bibliothèques du dispositif. Pour de futurs développements ou modifications, ces fichiers doivent être édités à la main avec un éditeur de type texte (comme Xemacs). Il est possible d'utiliser des éditeurs spécialisés pour faciliter la visualisation du code php (comme le mode perl de Xemacs, GoLive, aceHTML...)

A paramètres du site : *lib\parameters.lib*

Ce fichier consiste en une suite d'initialisations de variables qui permettent de contrôler plusieurs aspects et paramètres du site (couleur des menus, adresse email pour les formulaires, etc.)

Paramètres généraux

→ *\$database*

DSN de la base de données access. Vous pouvez changer ce nom dans le fichier si vous souhaitez en utiliser un autre. Vous devez bien entendu mettre en place un DSN système du même nom pointant vers le fichier de la base de données.

→ *\$home_page*

c'est l'adresse absolue par rapport à la racine du serveur web de la page de garde du site. Cette adresse est utilisée par les pages dotées de fonctions de redirection (check_login.php, kill_cookie.php....)

→ *\$start_page*

page à afficher à côté des menus pour la page de garde. Comme pour les fichiers qui s'affichent pour les menus, cette page ne doit contenir que le corps du fichier html (sans les tags <body> et </body>)

Paramètres pour l'envoi de mail

→ *\$smtp_host*

serveur smtp à utiliser pour l'envoi des mails (commentaires, commandes de rapports, inscriptions aux conférences)

→ *\$local_host*

nom du serveur local (eg monserveur.domain.com). Cette adresse est utilisée par les fonctions smtp.

→ *\$sender_address*

adresse affichée dans le "reply-to :" des mails envoyés par le dispositif.

→ *\$comment_address*

adresse mail où sont envoyés les commentaires sur le site.

→ ***\$order_address***

adresse mail où sont envoyés les commandes de rapports.

→ ***\$subscribe_address***

adresse mail où sont envoyés les inscriptions aux séminaires.

→ ***\$comment_mail_subject***

sujet affiché dans les mails de commentaire.

→ ***\$order_mail_subject***

sujet affiché dans les mails de commandes de rapports.

→ ***\$subscribe_mail_subject***

sujet affiché dans les mails d'inscriptions aux séminaires:

Couleurs

La définition des couleurs est faite en deux temps. On définit d'abord des couleurs génériques avec leur code html (ou le nom de la couleur prédéfinie). Ces couleurs génériques sont ensuite réutilisées pour les couleurs contextuelles. Ne sont données ici que les significations des couleurs contextuelles. Voir directement le fichier *parameters.lib* pour les couleurs génériques.

→ ***\$selected_menu_bg_color***

couleur de fond du menu sélectionné

→ ***\$selected_menu_color***

couleur du texte du menu sélectionné

→ ***\$menu_bg_color***

couleur de fond des autres menus

→ ***\$menu_color***

couleur du texte des autres menus

→ ***\$selected_sub_menu_bg_color***

couleur de fond du sous-menu sélectionné

→ ***\$selected_sub_menu_color***

couleur du texte du sous-menu sélectionné

→ ***\$sub_menu_bg_color***

couleur de fond des autres sous-menus

→ *\$sub_menu_color*

couleur du texte des autres sous-menus

→ *\$odd_news_bg_color*

couleur de fond des lignes paires dans l’affichage des news

→ *even_news_bf_color*

couleur de fond des lignes impaires dans l’affichage des news.

B Fonctions usuelles : *lib\usual_fonctions.lib*

Il s’agit des fonctions utilisées pour l’affichage normal du site (en mode utilisateur simple). Liste des fonctions de la librairie par ordre alphabétique.

→ *close_table()*

void close_table()

Envoie en output la balise `</table>` pour la fermeture d’une table html

→ *close_td()*

void close_td([boolean header])

Envoie en output la balise `</td>` (ou `</th>`) pour la fermeture d’une cellule de table html. Le paramètre header prend la valeur vrai si la cellule est une en-tête (faux par défaut)

→ *close_tr()*

void close_tr()

Envoie en output la balise `</tr>` pour la fermeture d’une ligne de table html

→ *echo_td()*

void echo_td(string string)

Envoie en output le string précédé de caractère de tabulation pour garder l’indentation du fichier html et en faciliter la lecture.

→ *get_dir_files()*

array get_dir_files([string dir])

Retourne dans un array la liste de tous les fichiers d’un répertoire sans les références systèmes au répertoire courant (.) et au répertoire parent (..). Si aucun paramètre n’est donné, la fonction retourne par défaut la liste des fichiers du répertoire courant (celui de la page qui appelle la fonction).

→ *get_next_index()*

int get_next_index(string table, string index_field, int db_pointer)

Retourne un integer correspondant au prochain index auto-incrémenté du champ “index_field” qui sera inséré dans la table “table”. La requête est envoyée à mysql en utilisant le pointeur “db_pointer” qui doit être ouvert préalablement à l’aide de la fonction `odbc_connect()`

➔ ***open_table()***

void open_table([string width[, int border[, string align[, int padding[, int spacing]]]])

Envoie en output la balise `<table ...>` pour l’ouverture d’une table html. Les paramètres contrôlent la disposition et les caractéristiques de la table.

Paramètres et valeurs par défaut :

paramètre	description	valeur par défaut
width	largeur de la table (en pixel ou en %)	n/a (automatique par défaut dans le navigateur)
border	épaisseur de la bordure	0
align	alignement horizontal de la table	n/a (à gauche par défaut dans le navigateur)
padding	marge en points à l’intérieur des cellules	n/a (défaut du navigateur)
spacing	espacement en point entre les cellules	n/a (défaut du navigateur)

exemples d’utilisation:

- bordure de 1, largeur de 20% : `open_table (“20%”, 1)`
- bordure de 0, alignée au centre, spacing 5 : `open_table (“”, 0, “center”, “”, 5)`

➔ ***open_td()***

void open_td([string width[, string color[, string align[, string valign[, int cspan[, boolean header[, int rspan]]]])])

Envoie en output la balise `<td ...>` (ou `<th...>`) pour l’ouverture d’une cellule de table html. Les paramètres contrôlent la disposition et le format de la cellule.

Paramètres et valeurs par défaut

paramètre	description
width	largeur de la cellule (en pixel ou en %)
color	couleur de fond (format html ou couleur prédéfinie) de la cellule
align	alignement horizontal du texte (left center right)
valign	alignement vertical du texte (top center bottom)
cspan	nombre de lignes sur laquelle la cellule doit s’étendre
header	vrai si la colonne est une en-tête de table, faux par défaut
rspan	nombre de colonnes sur lesquelles la cellule doit s’étendre.

→ *open_tr()*

void open_tr()

Envoie en output la balise <tr> pour l'ouverture d'une ligne de table html.

rem : pour de futurs développements, il pourrait être intéressant de modifier cette fonction pour qu'elle accepte des paramètres.

→ *throw_date()*

string throw_date(string string)

Retourne un string contenant seulement l'heure hh:mm:ss à partir d'un champ de la base de donnée contenant une date jj/mm/aaa hh:mm:ss

→ *throw_hour()*

string throw_hour(string string)

Retourne un string contenant seulement la date jj/mm/aaaa à partir d'un champ de la base de donnée contenant une date jj/mm/aaa hh:mm:ss

→ *today_date()*

string today_date()

Retourne un string contenant l'heure est la date actuelle au format dd/mm/aaaa hh:mm:ss (format pour insérer la date dans toute les tables de la base de donnée access).

C Fonctions de courrier électronique : *lib\smtp.lib*

Ce fichier contient une seule classe php qui doit être instanciée. Reportez vous à la documentation directement dans le fichier. Cette classe a été développée par Manuel Lemos qui la distribue gratuitement en Open-Source. Vous pouvez éventuellement en trouver une mise à jour s'il en existe une sur <http://phpclasses.upperdesign.com/>

D Fonctions d'administration

Les fonctions pour l'administration du site sont séparées dans des librairies distinctes. Il y a en effet un certain nombre de fonctions assez lourdes. La page administer.php appelle donc ces librairies une à une en fonction de l'action d'administration sélectionnée (news, agenda, menus, etc.). Ceci fait gagner du temps pour l'exécution de la page.

administration de l'agenda : *lib\agenda.lib*

Cette librairie se compose de 8 fonctions. La page administer.php n'appelle que la fonction manage_agenda() qui elle se charge d'appeler les autres pour des actions spécifiques.

→ *agenda_default()*

void agenda_default([array (or string) query_param])

Affiche les boutons de gestion (ajouter, supprimer, filtrer) et appelle la fonction `display_list()` pour afficher la liste des évènements. Si aucun paramètre n'est spécifié, la fonction affiche la liste des évènements à partir du jour même et les suivants.

Le paramètre `query_param` peut prendre 2 valeurs :

- string "all" : affiche tous les évènements de la table Schedule
- array indexé des choix fait dans le formulaire de sélection des évènements.

→ ***choose_event()***

void choose_event()

Affiche le formulaire pour sélectionner des évènements.

→ ***delete_event()***

boolean delete_event(int event_id, int db_pointer)

Supprime de la table Schedule l'évènement identifié par l'index unique `event_id` en utilisant le lien vers la base de donnée `db_pointer` (ouvert avec la fonction `odbc_connect()`). Retourne la valeur vrai en cas de succès ou faux si l'opération s'est mal passée.

→ ***display_list()***

void display_list(string query_string)

Affiche une liste d'évènements. Les évènements affichés dépendent du paramètre `query_string`, une requête sous forme littérale en langage SQL pour la base de donnée access.

→ ***filter_event()***

string filter_event (array query_param)

Retourne une requête littérale en langage SQL en fonction de l'array indexé `query_param`. Cette array contient en index le nom des paramètres qui ont été choisis dans le formulaire de sélection des évènements et leurs valeurs associées.

→ ***manage_agenda()***

void manage_agenda([int mode[, int event_id[, string event_day[, string event_month[, string event_year[, string event_hour[, string event_minute[, string location[, string description[, boolean gc[, boolean jfp[, boolean jm[, boolean rb[, boolean jp[, boolean jlg[, boolean query_param]]]]]]]]]]]]]]]]]]]]))

C'est la fonction centrale de la gestion de l'agenda. Elle reçoit tous les paramètres possibles provenant de tous les formulaires possibles de l'agenda. Appelée sans aucun paramètre, elle affiche la liste par défaut des évènements avec les boutons de gestion.

Le paramètre mode contrôle l'action demandée.

Valeurs possibles du paramètre mode

valeur	action associée
0	affichage de la page de gestion avec la liste des évènements sélectionnés.
1	formulaire pour insérer un évènement dans l'agenda
2	écriture du nouvel évènement dans la base de données
3	édition d'un évènement sélectionné
4	mise à jour de l'évènement édité dans la base de données
5	suppression d'un évènement sélectionné de la base de données
6	formulaire de sélection d'évènements

→ *write_event()*

void write_event([string switch, int event_id])

Affiche un formulaire pour insérer/éditer un évènement. Si elle n'est appelée avec aucun argument, on est en mode insertion. Appelée avec l'argument switch = "edit", on est en mode édition de l'évènement identifié par l'index unique event_id dans la table Schedule.

→ *write_event_entry()*

boolean write_event_entry(array parameters[, int event_id, string switch])

Appelée seulement avec l'argument parameters, insère un nouvel évènement dans la base de données. Appelée avec les 3 paramètres (dont switch = "update"), met à jour l'évènement identifié par l'index unique event_id dans la table Schedule.

Retourne vrai en cas de succès, faux autrement.

administration des conférences : lib\conferences.lib

Cette librairie se compose de 5 fonctions. La page administer.php n'appelle que la fonction manage_conf() qui elle se charge d'appeler les autres pour des actions spécifiques.

→ *conf_default()*

void conf_default()

Affiche la page principale pour la gestion des conférences avec la liste et les boutons (ajouter, supprimer, éditer)

→ *delete_conf()*

boolean delete_conf(int conf_id, int db_pointer)

Supprime de la base de données la conférence identifiée par l'index unique `conf_id` dans la table `conference` en utilisant le lien vers la base `db_pointer` (ouvert avec la fonction `odbc_connect()`)

→ ***manage_conf()***

```
void manage_conf([int mode[, int conf_id[, string title[, string text[,  
string start_day[, string start_month[, string start_year[, string end_day[,  
string end_month[, string end_year[, string location[, string country[,  
string url[, string email]]]]]]]]]]]]])
```

C'est la fonction centrale de la gestion des conférences. Elle reçoit tous les paramètres possibles de tous les formulaires possibles des conférences. Appelée sans aucun argument, elle affiche la page de gestion par défaut (affichée par la fonction `conf_default()`). Le paramètre `mode` contrôle l'action qui a été demandée

Valeurs possibles du paramètre `mode`

valeur	action associée
0	affichage de la page de gestion (defaut)
1	formulaire pour insérer une conférence
2	écriture de la nouvelle conférence dans la base de données
3	formulaire d'édition d'une conférence
4	mise à jour de la conférence éditée dans la base de données
5	suppression d'une conférence sélectionnée de la base de données

→ ***write_conf()***

```
void write_conf([string switch[, int conf_id]])
```

Appelée sans paramètre, affiche le formulaire pour insérer une nouvelle conférence. Appelée avec le paramètre `switch` = "edit", affiche les données de la conférence identifiée par l'index unique `conf_id` dans la table `conference` pour l'éditer.

→ ***write_conf_entry()***

```
boolean write_conf_entry(array parameters[, int conf_id, string switch])
```

Appelée seulement avec l'argument `parameters`, insère une nouvelle conférence dans la base de données. Appelée avec les 3 paramètres (dont `switch` = "update"), met à jour la conférence identifiée par l'index unique `event_id` dans la table `conference`.

Retourne vrai en cas de succès, faux autrement.

Administration des menus et sous-menus : *lib\menus.lib*

Cette librairie se compose de 6 fonctions qui permettent de gérer les menus du site présentés dans la vue utilisateur.

→ ***add_menu()***

```
void add_menu([int zone])
```

Affiche le formulaire pour ajouter un menu ou un sous menu.

Utilisation :

- `add_menu()` : affiche le formulaire pour un menu principal
- `add_menu(zone)` : affiche le formulaire pour un sous-menu à insérer dans le menu identifié par l'index unique `zone` dans la table `zone` de la base de données.

→ ***add_menu_entry()***

```
void add_menu_entry(string name, string file_path, string content[,  
string zone])
```

Ajoute dans la base de données les informations pour un menu (pas d'argument `zone`) ou un sous menu situé dans le menu identifié par l'index unique `zone` dans la table `zone` de la base de donnée.

- `name` : nom qui apparait pour le menu ou le sous-menu
- `file_path` : chemin du fichier à afficher pour ce menu
- `content` : contenu du fichier (qui sera écrit dans un fichier sur le disque)

→ ***del_room()***

```
void del_room(int current_room)
```

Efface de la table `pieces` et de la table `zones_pieces` toute les entrées relatives à la pièce (sous-menu) identifiée par l'index unique `current_room` dans la table `pieces`.

remarque : le fichier correspondant à ce sous-menu n'est pas effacé du disque.

→ ***del_zone()***

```
void del_zone(int current_zone)
```

Efface le menu identifié par l'index unique `current_zone` dans la table `zone` ainsi que tous les sous-menus qui y sont rattachés dans les tables `pieces` et `zones_pieces` de la base de données.

remarque : les fichiers correspondants aux menus et à ses sous-menus ne sont pas effacés du disque.

→ ***edit_menu()***

```
void edit_menu(int zone[, int piece])$
```

Affiche le formulaire pour éditer une entrée de menu ou de sous-menu. Utilisation:

- `edit_menu(zone)` : edite le menu identifié par l'index unique `zone` dans la table `zones` de la base de données.
- `edit_menu(zone, piece)` : edite le sous-menu identifié par l'index unique `pièce` dans la table `pieces` de la base de donnée est liée à au menu `zone`.

→ *edit_menu_entry()*

void edit_menu_entry(int id, string table, string name, string content, string file_path, boolean write_file)

Met à jour les informations pour le menu ou le sous menu dans la base de données ainsi que le fichier associé à cette entrée.

- id : indentifiant de l'entrée dans la table
- table : zones (menu) ou pieces (sous-menu)
- name : nom du menu
- content : contenu à écrire dans le fichier lié à la zone.
- file_path : chemin du fichier à afficher pour ce menu
- write_file : vrai si le contenu du fichier à changé, faux si ca n'a pas changé

Administration des news : lib/news.lib

Cette librairie se compose de 5 fonctions. La page administer.php n'appelle que la fonction *mange_news()* qui elle appelle les fonctions spécifiques selon l'action demandée.

→ *delete_news()*

void delete_news(int news_id, int db_pointer)

Supprime la nouvelle identifiée par l'index unique *news_id* dans la table *news* de la base de donnée en utilisant le lien *db_pointer* (créé avec la fonction *odbc_connect()*)

→ *manage_news()*

void manage_news([int mode[, int news_id[, string title[, string author[, string content[, string sday[, string smonth[, string syear[, string eday[, string emonth[, string eyear]]]]]]]]]]))

C'est la fonction centrale de la gestion des nouvelles. Elle recoit tous les paramètres possibles de tous les formulaires possibles dans la gestion des news. Appelée sans aucun paramètre, on obtient l'interface d'administration des news (affiché par la fonction *news_default()*). Le paramètre *mode* contrôle l'action demandée

Valeurs possibles du paramètre *mode*

valeur	action associée
0	affichage de l'interface par défaut
1	formulaire pour insérer une nouvelle
2	écriture de la nouvelle dans la base de données
3	formulaire d'édition d'une nouvelle
4	mise à jour de la nouvelle éditée dans la base de données
5	suppression d'une nouvelle sélectionnée de la base de données
6	purge : suppression de toutes les nouvelles dont la date de validité est inférieure à la date du jour.

→ ***news_default()***

void news_default()

Affiche l'interface de gestion des nouvelles avec la liste de toutes les nouvelles disponibles dans la base de données et les boutons (nouveau, editer, supprimer, purger)

→ ***write_news()***

void write_news([string mode[, int news_id]])

Affiche le formulaire pour insérer ou éditer une nouvelle dans la base de donnée. Utilisation :

- *write_news()* ou *write_news("insert")* : insertion d'une nouvelle news
- *write_news("edit", news_id)* : édite la nouvelle identifiée par l'index unique *news_id* dans la table news.

→ ***write_news_entry()***

boolean write_news_entry(array parameters[, int news_id[, string switch]])

Insère ou met à jour une nouvelle dans la base de données. L'array parameters contient toutes les informations concernant la nouvelle. Retourne vrai en cas de succès, faux autrement. Utilisation :

- *write_news_entry(parameters)* : insère une nouvelle.
- *write_news_entry(parameters, news_id, "update")* : met à jour la nouvelle identifiée par l'index unique *news_id* dans la table news.

Administration des projets : *lib\projets.lib*

Cette librairie se compose de 10 fonctions. La page administer.php n'appelle que la fonction *manage_project()* qui elle appelle les fonctions spécifique selon l'action demandée.

→ ***add_participant()***

void add_participant(string switch, int id)

Affiche le formulaire pour insérer/éditer un participant à un projet. Utilisation :

- *add_participant()* ou *add_participant("insert")* : formulaire pour ajouter un participant.
- *add_participant("edit", id)* : formulaire pour éditer le participant identifié par l'index unique *id* dans la table participants de la base de donnée.

→ ***add_participant_entry()***

boolean add_participant_entry (string switch, array parameters, int id)

Ajoute ou met à jour un participant dans la base de données. Les données pour le participant sont contenues dans l'array parameters. Retourne vrai en cas de succès, faux autrement. Utilisation :

- `add_participant_entry("insert", parameters)` : ajoute un nouveau participant à la base de données
- `add_participant_entry("update", parameters, id)` : met à jour le participant identifié par l'index unique id dans la table participants de la base de données

→ ***delete_participant()***

boolean delete_participant(int id)

Supprime le participant identifié par l'index unique id dans la table participants de la base de données. Retourne vrai si succès, faux autrement.

→ ***delete_project()***

boolean delete_project(int proj_id)

Supprime le projet identifié par l'index unique proj_id dans la table projects de la base de données ainsi que les entrées le concernant dans les tables relationnelles projects_participants et projects_categories. Retourne vrai en cas de succès, faux autrement.

→ ***display_participant()***

void display_participant()

Affiche la liste de tous les participants aux projets avec des boutons d'administration (ajouter, éditer, supprimer)

→ ***manage_project()***

void manage_project(int mode, int proj_id, string titre, string description, string rapport, int en_cours, string sday, string smonth, string syear, string eday, string emonth, string eyear, string parution, string rapport_doc, int rapport_comm, int responsableID, int categorieID, boolean write_file, string file_content, string prenom, string nom, string adresse_1, string adresse_2, string code_postal, string ville, string pays, string tel, string fax, string email, int inserting, int id, string commentaire, string photo)

C'est la fonction centrale de la gestion des projets. Elle recoit tous les paramètres possibles de tous les formulaires possibles de la gestion des projets. Le paramètre mode controle l'action qui a été demandée.

Valeurs possibles du paramètre mode

valeur	action associée
0	affichage de la liste des projets et des boutons d'administration (ajouter, supprimer, éditer, gestion des participants...)

Valeurs possibles du paramètre mode

valeur	action associée
1	formulaire pour ajouter un projet
2	insertion du nouveau projet dans base de données
3	formulaire pour éditer un projet
4	mise à jour du projet édité dans la base de données
5	suppression d'un projet
6	passer un projet de l'état "en cours" à l'état "fini"
7	archivage d'un projet (**non-implémenté**)
8	formulaire pour ajouter un participant
9	insertion du nouveau participant dans la base de données
10	affichage et gestion de la liste des participants
11	formulaire pour éditer un participant
12	mise à jour du participant édité dans la base de données
13	supprimer un participant

→ *project_default()*

void project_default()

Affiche la liste de tous les projets en cours et finit avec les boutons d'administration.

→ *set_project_finished()*

boolean set_project_finished(int proj_id)

Fait passer le projet identifié par l'index unique *proj_id* dans la table *projects* de l'état "en_cours" à l'état "fini". Retourne vrai en cas de succès, faux autrement.

→ *write_project()*

void write_project([string switch[, int proj_id]])

Affiche le formulaire pour insérer/éditer un projet. Utilisation :

- *write_project()* ou *write_project("insert")* : affiche le formulaire pour insérer un nouveau projet.
- *write_project("edit", proj_id)* : affiche le formulaire pour éditer le projet identifié par l'index unique *proj_id* dans la table *projects*.

→ *write_project_entry()*

boolean write_project_entry(array parameters, array responsableID, array categorieID, boolean write_file, string file_content, string switch, int proj_id)

Insère ou met à jour un projet dans la base de données.

- *parameters* : les données associées au projet (nom, description, etc.)
- *responsableID* : array des index uniques des responsables du projet dans la table *responsables*

- `categorieID` : array des index uniques des catégories auxquelles appartient le projet dans la table `categories`.
- `write_file` : vrai si le fichier associé au projet a changé, faux sinon.
- `file_content` : contenu du fichier à écrire sur le disque
- `switch` : “insert” pour un nouveau projet, “update” pour une édition
- `proj_id` : index unique identifiant le projet édité dans la table `projet` (switch “update”)

Administration des séminaires : *lib\seminaires.lib*

Cette librairie se compose de 5 fonctions. La page `administer.php` n’appelle que la fonction `mange_sem()` qui elle se charge d’appeler les autres en fonction de l’action demandée.

→ *delete_sem()*

boolean delete_sem(int sem_id, int db_pointer)

Supprime de la base de données le séminaire identifié par l’index unique `sem_id` de la table `seminaires` en utilisant le lien vers la base de données `db_pointer` (ouvert avec la fonction `odbc_connect()`). Retourne vrai en cas de succès, faux sinon.

→ *manage_sem()*

void manage_sem([int mode[, int sem_id[, string titre[, string resume[, string sem_day[, string sem_month[, string sem_year]]]]]]])

C’est la fonction centrale de la gestion des séminaires. Elle reçoit tous les paramètres possibles de tous les formulaires possibles. Appelée sans aucun paramètres, elle affiche la liste des conférences avec les boutons de gestion (en utilisant la fonction `sem_default()`). Le paramètre `mode` contrôle l’action demandée.

Valeurs possibles du paramètre `mode`

valeur	action associée
0	affichage par défaut de la liste des conférences avec les boutons de gestion
1	formulaire pour insérer un séminaire
2	insertion du nouveau séminaire dans la base de données
3	formulaire d’édition d’un séminaire
4	mise à jour du séminaire édité dans la base de données
5	suppression d’un séminaire dans la base de données

→ *sem_default()*

void sem_default()

Affiche la liste des séminaires avec les boutons d’administration (ajouter, éditer, supprimer)

→ *write_sem()*

void write_sem([string mode[, int sem_id]])

Affiche le formulaire pour insérer/éditer un séminaire. Utilisation :

- *write_sem()* ou *write_sem("insert")* : affiche le formulaire pour un nouveau séminaire.
- *write_sem("edit", sem_id)* : affiche le formulaire pour éditer le séminaire identifié par l'index unique *sem_id* dans la table *seminaires* de la base de données.

→ *write_sem_entry()*

boolean write_sem_entry(array parameters[, int sem_id[, string switch]])

Insère ou met à jour un séminaire dans la base de données. Toutes les informations associées au séminaire sont contenues dans l'array *parameters*. Exemples d'utilisation.

- *write_sem_entry(parameters)* ou *write_sem_entry(parameters, "", "insert")* : insère un nouveau séminaire dans la base de donnée.
- *write_sem_entry(parameters, sem_id, "update")* : met à jour le séminaire identifié par l'index unique *sem_id* dans la table séminaire.

IX Conclusions

A Travail sur le projet

Comment ne pas commencer cette conclusions par quelques regrets...

Tout d'abord, j'ai commencé ce stage alors que j'étais par ailleurs embauché comme ingénieur système à mi-temps à TECFA, charge qui a très rapidement dépassé les 50% d'activité qui était prévu. D'autre part, ce stage s'effectuait totalement à distance : mis à part les 2 réunions que nous avons eu à Fribourg, tout le reste a été fait par email ou par téléphone.

La conjonction de ces deux paramètres a fait que cette tâche s'est très souvent retrouvée en bas de la liste des priorités ce qui a occasioné énormément de retard:

- dépassement des délais prévues
- pertes de temps quand une direction prise dans le developpement n'avait pas été clairement explicitée et comprise de part et d'autre.
- supplément d'heures de travail due à l'allongement des délais (estimée à environs 10 jours plein temps par rapport à la charge de 1,5 mois plein temps initialement prévue)

Pour toutes ces raisons, je tiens à présenter mes plus vives excuses au centre NTE et à ses collaborateurs.

Mais il y a bien entendu des points très positifs.

Il s'agit tout d'abord de la première véritable application pour le web que je développe. Elle est conséquente (plus de 4000 lignes de code php) et j'ai mis un point d'honneur à la documenter de façon correcte. J'espère que le résultat est à la hauteur des espérances du centre NTE.

Commenter le code que l'on écrit n'est pas quelque chose de naturel : cela s'apprend et se travaille. Je suis très reconnaissant aux collaborateurs du centre de m'avoir imposé cette contrainte qui m'a permis d'acquiescer des automatismes qui se révèlent aujourd'hui salvateurs quand je reprends des dispositifs programmés de longue date. Et la rédaction de la documentation exhaustive de ce dispositif m'a mieux permis de me rendre compte, avec du recul, de la somme de travail que j'ai investie dans ce dispositif.

Il a fallu ensuite que je réfléchisse à un modèle et à une organisation des données et de leur traitement. Ceci m'a permis de me familiariser avec le design de base de données, les algorithmes de traitement, l'organisation des fichiers pour limiter la duplication de code. Cette expérience sera un atout majeur dans les futurs dispositifs que je serais amené à développer.

Le cahier des charges de cette "coquille vide" a été quasiment réalisé en intégralité, à l'exception des projets complets en intranet et de la carte dynamique du site. Reste maintenant à la charge du centre NTE de continuer ce développement et de rentrer les informations sur le site, et de l'utiliser s'il juge ce développement adapté à ses besoins dans sa forme actuelle.

B Futurs développements

Voici pour réflexion quelques points qui pourraient être modifiés ou ajoutés à ce dispositif.

Pour l'instant, le site compte 3 administrateurs qui ont des postes fixes. Si le nombre des administrateurs et leurs turn-over venaient à augmenter, il me semble indispensable de développer un outil pour ajouter/supprimer/modifier/gérer les logins des administrateurs.

Les formulaires pour ajouter des menus, des sous-menus, des projets... demandent encore une assez bonne connaissance du langage html puisque le contenu des pages est tapé directement dans un champ texte. Ceci peut devenir un problème si des personnes non techniciennes sont amenées à rentrer du contenu dans le site. Ces formulaires pourraient être améliorés par l'adjonction d'un éditeur interactif de type javascript qui permette de faire de la mise en forme de base (gras, souligné, titre, liens hypertextes...) sans avoir à connaître la syntaxe html.

Pour faciliter les changements de look du site, il pourrait être judicieux de développer une interface pour l'administration des couleurs qui permette de les changer sans avoir à éditer le fichier de paramètres.

Dans le dispositif actuel, un menu ou sous-menu ajouté vient automatiquement se placer après les menus ou sous-menus déjà existants. Pour des commodités d'utilisation du site et de mise à jour, il pourrait-être intéressant de permettre aux administrateurs de configurer l'ordre d'apparition des menus dans le site, quel que soient leur dates de création et leur position dans la base de données.