Documentation technique

Base de données :

C'est le SGBDR MySQL qui a été utilisé pour l'architecture des données de cette application. Grâce au Framework Cakephp utilisé pour réaliser le projet, aucune dépendance fonctionnelle n'est a créée directement dans la base de données, il suffit d'utiliser la syntaxe propre à Cakephp et expliquer dans la documentation technique du Framework.
Par exemple : Si il existe un élève « A » dans une classe « 1 », alors l'élève aura pour clé étrangère l'id de la classe. Grâce à Cakephp, il suffit d'écrite la clé étrangère « classe_id » dans la table élève.
De même, il suffit de mettre un « s » à la fin de chacune des tables, et d'utiliser l'outil offert par Cakephp pour générer directement l'ensemble des modèles, des contrôleurs et des vues par défaut (Ajout, suppression, modification).

Framework et autres outils :

- Comme dit précédemment, le Framework utilisé est Cakephp 2.3 (http://cakephp.org). Cakephp respecte le principe du MVC (modèle, vue et controller).
- L'éditeur de code utilisé est Coda2 (http://panic.com/coda/).
- Le serveur de développement est MAMP.

Règles et contraintes :

L'utilisation de Cakephp entraine certaines contraintes. Tel qu'une syntaxe propre qui doit être utilisé dans la base de données. On a parlé, précédemment, des clés étrangères et du nom des tables. Cette syntaxe s'applique aussi aux champs propre aux tables, par exemple, l'identifiant de la table devra se nommer « id », si il y a un nom, pour une utilisateur par exemple, ce sera « name ».
Le Framework possède aussi l'avantage d'avoir recours à deux champs qui sont « created » et « modified », qui, lorsqu'ils sont crées, entraine leur modification automatique à chaque ajout ou modification d'un objet d'un modèle généré avec Cakephp.

Grâce à une série de commande assez simple, on peut générer les modèles, vues et controllers associées aux tables de la base à l'aide de la console :

- cakebake all => affiche l'ensemble des tables de la base numérotées.
- On sélectionne ensuite les tables pour lesquelles on souhaite procéder à une génération automatique.
- On valide

Affichage dans le navigateur

Dans le navigateur, on remarquera que les vues affichées sont toujours reliées à un modèle. Par exemple, pour la connexion d'un utilisateur, l'url sera :

http://localhost/compteRendu/users/login

C'est donc la fonction login du controller du modèle users, qui permet d'afficher la vue qui s'y rapporte.

http://localhost/compteRendu/rapports_visites/user_add affichera donc la vues qui permet à un « user » d'ajouter une rapport de visite.

L'url rewriting est donc très cohérent.

Certaine fonction ont dues être ajoutées aux fonctions existantes du Framework :

```
public function sha1SaltForAll(){
        $users=$this->User->find('all');
        foreach($users as $user):
                $this->User->save($user['User']);
        endforeach;
}
```

Cette fonction permet de sauvegarder tous les « users » déjà présents dans la base de données. En procédant de cette manière, on met à jour l'ensemble des utilisateurs.

Or, dans le modèle user, existe une fonction beforeSave (déjà existante dans le Framework) :

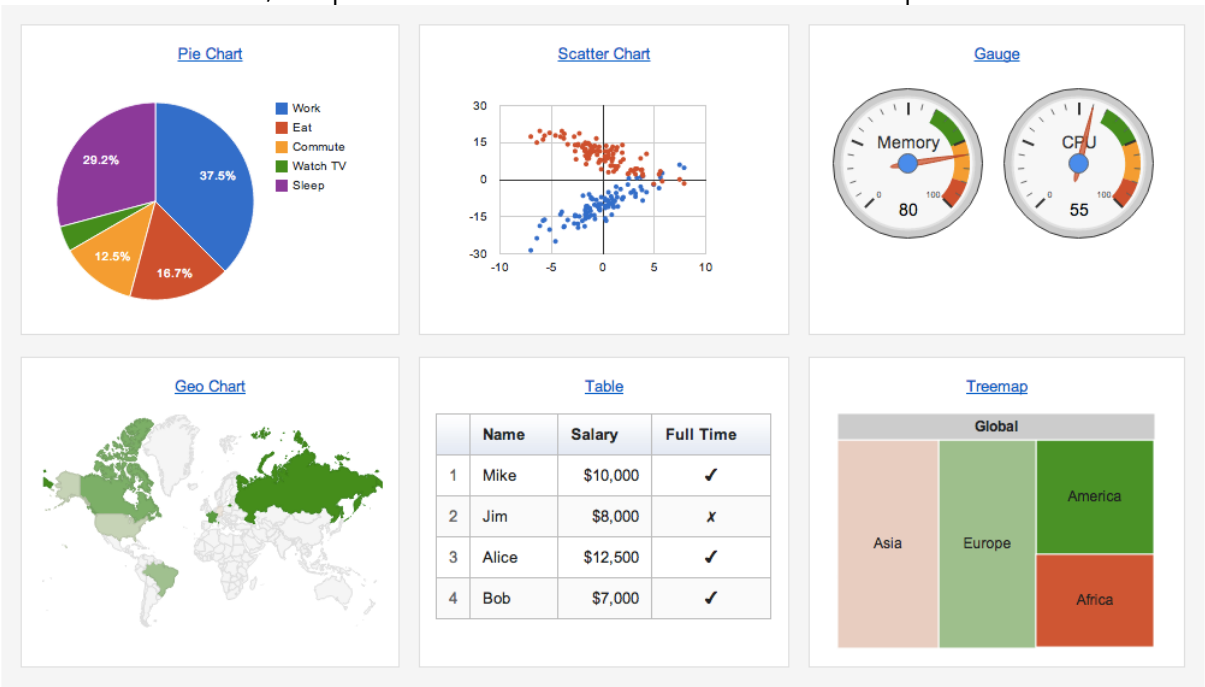
```
	public function beforeSave($options=array()) {
		App::uses('Utility','Security');

		if(!empty($this->data['User']['password'])) {
			$this->data['User']['password'] = Security::hash($this->data['User']['password'], 'sha1', true);
		}
		return true;
	}
```

Cette fonction permet d'effectuer une tâche avant l'enregistrement d'un user, on l'a donc modifié pour que le password de chaque user se hash grâce au salt et au hashage sha1.

Statistiques :

Pour les statistiques, on a utilisé un outil fourni par Google : Google Charts. Grâce à cet outil, on peut créer toute sorte d'outils statistiques.



Les divers outils offerts par Google se présente sous la forme de snippets (Morceaux de code). On copie le snippets de Google et on le colle à l'endroit où l'on souhaite voir s'afficher les statistiques. On ne reste plus qu'à lui envoyer les données souhaitées.

Structure de Cakephp :

DOSSIER / FICHIERS

- CONFIG
  - Core.php (Contient la valeur d'encryptage de l'application)
  - Database.php (Contient les informations de connexion à la BDD)
- CONTROLLER (Contient tous les controllers de l'application)
- MODEL (Contient tous les modèles de l'application)
- VIEW
  - Contient toutes les vues de l'application par dossier suivant la règle suivante :
    - Dossier : nomDuController–s
    - Views : add, delete, index, view (par défaut).
    - Contient également l'affichage principal de l'application dans le dossier Layout.
    - Et des parties de l'affichage principal chargeable suivant les droits accordés à chaque utilisateur.
- WEBROOT
  - Contient les autres éléments associés à l'application : css, images, js …