CONTEXTE GSB – APPLICATION COMPTE-RENDU

Application Web permettant la gestion des comptes-rendus de visite.

Application accessible dans le futur aux terminaux mobiles.

Application sous ACCESS 2003 existante et obsolète (voir script SQL de création BD et insertion données sous Mysql).

ETUDE DES DONNÉES

Prise en compte de nouvelles règles de gestion afin de compléter le modèle existant :

Notion de collaborateur inclut les délégués et les responsables.
On gère le motif des visites.
Un responsable gère un secteur.
Un délégué suit une région.
Une région est rattachée à un secteur.
On gère les médicaments (échantillons) présentés au cours d'une visite ainsi que le motif de la visite.
On gère le remplacement d'un praticien et on ajoute un coefficient de confiance pour le praticien.

EXPLICATIONS COMPLÉMENTAIRES SUR LES TRAITEMENTS

L'application **GSBApplicompteRendu** a été commencée en respectant l'architecture MVC.

Le fichier constants.php (dossier config) contient différentes constantes permettant le paramétrage de certains éléments de configuration (URL, serveur, base de données). Vérifiez et corrigez les valeurs de ces constantes en fonction de votre environnement de travail.

Les données de la base mysql sont gérées par l'intermédiaire de requêtes SQL dans le fichier **class.pdogsb.inc.php** (dossier include). Ce fichier est à compléter. Il contient une classe PdoGsb contenant des données et des méthodes statiques permettant la connexion et l'exploitation de la base de données.
Attention à modifier si nécessaire les données concernant le serveur, le nom de la base, le login et mot de passe dans le fichier **constants.php** (dossier config).
Les méthodes existantes sont des fonctions retournant le résultat d'une requête SQL ou un résultat de type booléen.

Le fichier **fct.inc.php** (dossier include) contient différentes fonctions utilitaires. Ce fichier est également à compléter.

Un fichier **index.php** vous est fourni.
Il doit déterminer quel type d'utilisateur est connecté (visiteur, délégué ou responsable).
Si personne n'est connecté, le contrôleur **c_connexion.php** est appelé. La vue **v_connexion.php** est affichée. A la validation de la connexion, la vue **v_sommaireVisiteur.php** ou **v_sommaireDelegue.php** ou **v_sommaireResponsable.php** devra s'afficher selon le type d'utilisateur connecté.
Lorsqu'un collaborateur est connecté, son nom, son prénom et sa fonction s'affichent. Pour cela, nous utilisons des variables de session définies dans des fonctions situées dans le fichier **fct.inc.php.**
Les informations du collaborateur connecté sont à rechercher dans la base de données par l'intermédiaire d'une fonction exécutant une requête SQL.

Le contrôleur **c_connexion.php** ne gère pour l'instant que la connexion pour un collaborateur visiteur. Il doit être complété ainsi que le fichier **index.php** qui doit définir les appels (include) vers les différents contrôleurs.

Le module visiteur (le plus urgent à programmer) est composé des traitements suivants :

- ajouterCompteRendu : appel du contrôleur **c_ajouterCompteRendu.php**
- modifierCompteRendu : appel du contrôleur **c_modifierCompteRendu.php**
- ajouterPraticien : appel du contrôleur **c_ajouterPraticien.php**
- consulterPraticien : appel du contrôleur **c_consulterPraticien.php**
- consulterMedicament : appel du contrôleur **c_consulterMedicament.php**
- consulterCompteRenduPeriode : appel du contrôleur **c_consulterCompteRenduPeriode.php**

Le module délégué est composé des traitements du visiteur et du traitement suivant :

- consulterCompteRenduRegion : appel du contrôleur **c_consulterCompteRenduRegion.php**

Ces différents contrôleurs doivent être programmés en tenant compte des scénarios de chaque cas d'utilisation fournis dans le document "**Spécifications fonctionnelles.pdf**". Ils doivent faire appel à des vues (IHM). Certaines de ces vues existent et sont incomplètes, d'autres n'existent pas et devront être créées.
Les sommaires **v_sommaireVisiteur.php, v_sommaireDelegue.php** et **v_sommaireResponsable.php** devront être complétés en fonction de la programmation des actions des contrôleurs.
Attention : Le graphisme est géré pour l'instant par un fichier style.css qui ne tient pas compte de la problématique du **responsive design** (utilisation dans le futur de l'application sur tablette ou téléphone).
Les IHM fournies ne sont que des propositions et peuvent être modifiées à condition de respecter les règles de gestion concernant les traitements.

Proposition de l'IHM v_ajoutCompteRendu.php

- Si visite auprès d'un praticien remplaçant alors enregistrement du praticien principal et du remplaçant qui doit être dans la liste des praticiens.
- Si choix d'un autre motif alors saisir le motif dans la zone de texte
- Eléments présentés (de 0 à 2 possibles) : cocher ou non la case correspondante afin de pouvoir définir si un médicament a été présenté.
- Documentation offerte : cocher ou non la case.
- Echantillons offerts (de 0 à 10 possibles) : la quantité est renseignée si un échantillon est offert.

Vous pouvez utiliser la fonction empty.

Une transaction devra être gérée afin d'ajouter ou de modifier un compte-rendu car plusieurs tables de la base de données doivent être mises à jour.

Vous utiliserez dans les contrôleurs correspondants l'instruction suivante :

```
try {
        $pdo->demarreTransaction();
...
        $pdo->valideTransaction();
...
}
catch (Exception $e) {
        $pdo->annuleTransaction();
...
}
```

Dans le fichier **class.pdogsb.inc.php**, les instructions et les méthodes afin de gérer les exceptions et les transactions ont été programmés.

```
private function __construct(){
        PdoGsb::$monPdo = new PDO(PdoGsb::$serveur.';'.PdoGsb::$bdd, PdoGsb::$user, PdoGsb::
$mdp);
        PdoGsb::$monPdo->query("SET CHARACTER SET utf8");
        PdoGsb::$monPdo->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
}

/**
 * Démarre une transaction
*/
        public function demarreTransaction(){
                PdoGsb::$monPdo->beginTransaction();
        }

/**
 * Valide une transaction
*/
        public function valideTransaction(){
                PdoGsb::$monPdo->commit();
        }

/**
 * Annule une transaction
*/
        public function annuleTransaction(){
                PdoGsb::$monPdo->rollback();
        }
```