Compte rendu

Développement en PHP de la partie comptable de l'application de gestion de frais des visiteurs médicaux du laboratoire

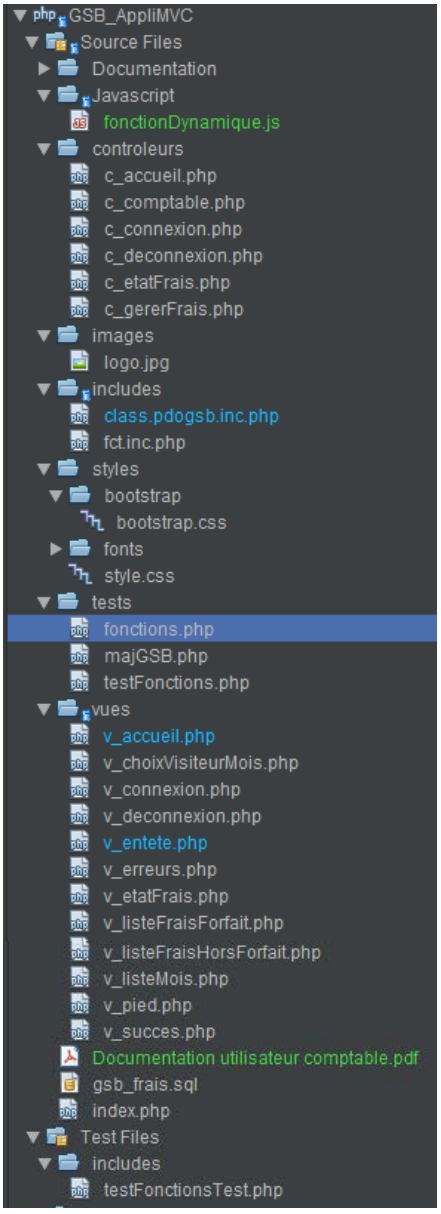
Table des matières

1. Logiciels utilisés

- IDE Netbeans 11.2
- Php unit
- Php Documentor
- Phpmyadmin
- Wampserver64
- Windows 64bit
- Git et Github

2. Arborescence des fichiers

Application créée sur le modèle MVC

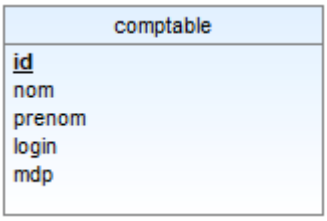


3. Modification BDD

Création d'une table comptable (colonne id, nom, prenom, login, mdp) dans la base gsb_frais.

COMPTABLE (id, nom, prenom, login, mdp)

id : clé primaire



Modification table « etat ». Ajout d'un tuple « PM » « Mise en paiement » et modification du tuple « Validée » et « Mise en paiement » en « Validée ».

id	libelle
CL	Saisie clôturée
CR	Fiche créée, saisie en cours
PM	Mise en paiement
RB	Remboursée
VA	Validée

4. Modification code source

Création de la méthode de classe « getInfosComptable » :

```php
/**
 * Retourne les informations d'un comptable
 *
 * @param String $login Login du comptable
 * @param String $mdp Mot de passe du comptable
 *
 * @return Array l'id, le nom et le prénom du comptable sous la forme d'un tableau associatif
 */
public function getInfosComptable($login, $mdp)
{
    $requetePrepare = PdoGsb::$monPdo->prepare(
        'SELECT comptable.id AS id, comptable.nom AS nom, '
        . 'comptable.prenom AS prenom '
        . 'FROM comptable '
        . 'WHERE comptable.login = :unLogin AND comptable.mdp = :unMdp'
    );
    $mdpHash = hash('sha256',$mdp);
    $requetePrepare->bindParam(':unLogin', $login, PDO::PARAM_STR);
    $requetePrepare->bindParam(':unMdp', $mdpHash, PDO::PARAM_STR);
    $requetePrepare->execute();
    return $requetePrepare->fetch();
}
```

Ajout d'un argument à la méthode « connecter » (« typeUtilisateur » pour différencier comptable et visiteur) :

```php
/**
 * Enregistre dans une variable session les infos d'un visiteur
 *
 * @param String $idVisiteur        ID de l'utilisateur
 * @param String $nom               Nom de l'utilisateur
 * @param String $prenom            Prénom de l'utilisateur
 * @param String $typeUtilisateur   Type de l'utilisateur
 *
 * @return null
 */
function connecter($idVisiteur, $nom, $prenom, $typeUtilisateur)
{
    $_SESSION['idVisiteur'] = $idVisiteur;
    $_SESSION['nom'] = $nom;
    $_SESSION['prenom'] = $prenom;
    $_SESSION['utilisateur'] = $typeUtilisateur;
}
```

Modification du contrôleur « c_connexion » :

```php
switch ($action) {
case 'demandeConnexion':
    include 'vues/v_connexion.php';
    break;
case 'valideConnexion':
    $login = filter_input(INPUT_POST, 'login', FILTER_SANITIZE_STRING);
    $mdp = filter_input(INPUT_POST, 'mdp', FILTER_SANITIZE_STRING);
    $visiteur = $pdo->getInfosVisiteur($login, $mdp);
    $typeUtilisateur = "visiteur";

    if ($visiteur == null) {
        $visiteur = $pdo->getInfosComptable($login, $mdp);
        $typeUtilisateur = "comptable";
    }

    if (!is_array($visiteur)) {
        ajouterErreur('Login ou mot de passe incorrect');
        include 'vues/v_erreurs.php';
        include 'vues/v_connexion.php';
    } else {
        $id = $visiteur['id'];
        $nom = $visiteur['nom'];
        $prenom = $visiteur['prenom'];
        connecter($id, $nom, $prenom, $typeUtilisateur);
        header('Location: index.php');
    }
    break;
default:
    include 'vues/v_connexion.php';
    break;
}
```

Modification des vues « v_entete » et « v_accueil » pour différencier l'affichage selon le type d'utilisateur connecté (grâce à la valeur de $_SESSION['utilisateur']).

Vue en-tête :

```php
<div class="col-md-8">
    <ul class="nav nav-pills pull-right" role="tablist">
        <li <?php if (!$uc || $uc == 'accueil') { ?>class="active" <?php } ?>>
            <a <?php echo "class=",$_SESSION['utilisateur'] , "couleur "; ?>
                href="index.php">
                <span class="glyphicon glyphicon-home"></span>
                Accueil
            </a>
        </li>
        <li <?php if ($uc == 'gererFrais') { ?>class="active"<?php } ?>>
            <a <?php echo "class=",$_SESSION['utilisateur'] , "couleur "; ?>
                href="index.php?uc=gererFrais&action=saisirFrais">
                <span class="glyphicon glyphicon-pencil"></span>
                <?php if ($_SESSION['utilisateur'] == 'visiteur'){?>
                    Renseigner la fiche de frais
                <?php } else {?>
                    Valider les fiches de frais
                <?php }?>
            </a>
        </li>
```

```
<li <?php if ($uc == 'etatFrais') { ?>class="active"<?php } ?>>
    <a <?php echo "class=",$_SESSION['utilisateur'] , "couleur "; ?>
        href="index.php?uc=etatFrais&
            <?php if ($_SESSION['utilisateur'] == 'comptable') {
            ?>action=voirEtatFrais<?php } else { ?>action=selectionnerMois<?php }?>">
        <span class="glyphicon glyphicon-list-alt"></span>
        <?php if ($_SESSION['utilisateur'] == 'visiteur'){?>
            Afficher mes fiches de frais
        <?php } else {?>
            Suivre le paiement des fiches de frais
        <?php }?>
    </a>
</li>
<li
<?php if ($uc == 'deconnexion') { ?>class="active"<?php } ?>>
    <a <?php echo "class=",$_SESSION['utilisateur'] , "couleur "; ?>
        href="index.php?uc=deconnexion&action=demandeDeconnexion">
        <span class="glyphicon glyphicon-log-out"></span>
        Déconnexion
    </a>
</li>
```

Vue accueil :

```
<div id="accueil">
    <h2>
        Gestion des frais<small> -
        <?php if ($_SESSION['utilisateur'] == 'visiteur'){ ?>
            Visiteur :
        <?php } else { ?>
            Comptable :
            <?php
            }
            echo $_SESSION['prenom'] . ' ' . $_SESSION['nom']
            ?></small>
    </h2>
</div>
```

```
<div class="row">
    <div class="col-md-12">
        <div class="panel panel-primary <?php echo $_SESSION['utilisateur']; ?>">
            <div class="panel-heading">
                <h3 class="panel-title">
                    <span class="glyphicon glyphicon-bookmark"></span>
                    Navigation
                </h3>
            </div>
```

```
<div class="col-xs-12 col-md-12">
    <a
        href="index.php?uc=gererFrais&action=saisirFrais"
        class="btn btn-success btn-lg" role="button">
        <span class="glyphicon glyphicon-pencil"></span>
        <br>
        <?php if ($_SESSION['utilisateur'] == 'visiteur'){ ?>
            Renseigner la fiche de frais
        <?php } else { ?>
            Valider les fiches de frais
        <?php } ?>
    </a>
    <a href="index.php?uc=etatFrais&
            <?php if ($_SESSION['utilisateur'] == 'comptable') {
            ?>action=voirEtatFrais<?php } else { ?>action=selectionnerMois<?php }?>"
       class="btn btn-primary btn-lg
       <?php echo $_SESSION['utilisateur'] ?>" role="button">
        <span class="glyphicon glyphicon-list-alt"></span>
        <br>
        <?php if ($_SESSION['utilisateur'] == 'visiteur'){ ?>
            Afficher mes fiches de frais
        <?php } else { ?>
            Suivre le paiement des fiches de frais
        <?php } ?>
    </a>
</div>
```

J'ai décidé de réutiliser un maximum les vues déjà existantes afin de ne pas alourdir le projet.

Décision de modifier la partie « choix visiteur et mois » car après réflexion, je me suis rendu compte que logiquement une unique fiche visiteur doit être disponible pour chaque visiteur. Effectivement les fiches visiteurs sont clôturés le 1er du mois suivant et que le 20eme jour du mois N+1 les fiches passent à l'état mise en paiement. J'ai quand même envisagé qu'une fiche ai pu être oubliée dans un certain état d'où l'idée d'une liste déroulante pour les nom et prénom des visiteurs et une liste déroulante pour les mois.

Création du contrôleur « c_comptable » :

```
switch ($uc) {
    case 'gererFrais':
        $listeDeVisiteur = $pdo->getListeVisiteurFicheEtat('CL', null);
        break;
    case 'etatFrais':
        $listeDeVisiteur = $pdo->getListeVisiteurFicheEtat('VA', 'PM');
        break;
    default:
        $uc = 'Erreur';
        break;
}
if (($listeDeVisiteur != null) && ($uc != 'Erreur')) {
    $action = filter_input(INPUT_GET, 'action', FILTER_SANITIZE_STRING);
    $listeNomPrenomVisiteur['nomPrenom'] = creerListeNomPrenom($listeDeVisiteur);
    $listeNomPrenomVisiteur['idVisiteur'] = extraireListe($listeDeVisiteur,'idVisiteur');
    $indexNom = filter_input(INPUT_POST, 'indexListeNom', FILTER_SANITIZE_NUMBER_INT);
    if ($indexNom == null) {
        $indexMois = 0;
        $indexNom = 0;
    } else {
        $indexMois = filter_input(INPUT_POST, 'indexListeMois', FILTER_SANITIZE_NUMBER_INT);
    }
    $listeMois = $pdo->rechercheListeMois($listeNomPrenomVisiteur['idVisiteur'], $uc);
    $listeNomPrenomVisiteur['mois'] = $listeMois;
    $indexListeNom = verificationIndex($listeNomPrenomVisiteur['idVisiteur'], $indexNom);
    $idVisiteur = $listeNomPrenomVisiteur['idVisiteur'][$indexListeNom];
    $indexListeMois = verificationIndex($listeNomPrenomVisiteur['mois'][$indexListeNom], $indexMois);
    $moisOrdonne = $listeNomPrenomVisiteur['mois'][$indexListeNom][$indexListeMois];
    $numAnnee = substr($moisOrdonne, 5);
    $numMois = substr($moisOrdonne, 0, 2);
    $mois = $numAnnee . $numMois;
    if ($action == 'validerFicheDeFrais') {
        $nbJustificatifs = filter_input(INPUT_POST, 'nbJustificatifs', FILTER_SANITIZE_NUMBER_INT);
        valideJustificatifs($nbJustificatifs);
    }
    if (($_SESSION['utilisateur'] == 'comptable')
        && $listeDeVisiteur != null ) {
        require 'vues/v_choixVisiteurMois.php';
    }
} else if (($listeDeVisiteur == null) && ($uc != 'Erreur')){
    ajouterErreur('Il n\'éxiste aucune fiche nécessitant une intervention.');
    include 'vues/v_erreurs.php';
}
```

Création de la méthode de classe « getListeVisiteurFicheEtat » :

```php
/**
 * Retourne les noms et prénom dans l'ordre alphabétique des visiteurs pour
 * lequelles il éxiste une fiche de frais dans un ou deux états souhaité.
 *
 * @param String $etatA premier état voulu.
 * @param String $etatB second état voulu.
 *
 * @return Array les noms et prénoms des visiteurs pour lequel il éxiste
 * une fiche de frais aux états souhaité sous la forme d'un tableau associatif.
 */
public function getListeVisiteurFicheEtat($etatA, $etatB)
{
    if ($etatB == null) {
        $requetePrepare = PdoGSB::$monPdo->prepare(
            'SELECT DISTINCT visiteur.nom as nom, '
            . 'visiteur.prenom as prenom, visiteur.id '
            . 'as idVisiteur, fichefrais.idetat as idEtat '
            . 'FROM visiteur INNER JOIN '
            . 'fichefrais ON visiteur.id = '
            . 'fichefrais.idvisiteur WHERE '
            . 'fichefrais.idetat = :unEtatA ORDER BY '
            . 'visiteur.nom'
        );
        $requetePrepare->bindParam(':unEtatA', $etatA, PDO::PARAM_STR);
    } else {
        $requetePrepare = PdoGSB::$monPdo->prepare(
            'SELECT DISTINCT visiteur.nom as nom, '
            . 'visiteur.prenom as prenom, visiteur.id '
            . 'as idVisiteur FROM visiteur INNER JOIN '
            . 'fichefrais ON visiteur.id = '
            . 'fichefrais.idvisiteur WHERE '
            . 'fichefrais.idetat = :unEtatA OR fichefrais.idetat = :unEtatB '
            . 'ORDER BY visiteur.nom'
        );
        $requetePrepare->bindParam(':unEtatA', $etatA, PDO::PARAM_STR);
        $requetePrepare->bindParam(':unEtatB', $etatB, PDO::PARAM_STR);
    }
    $requetePrepare->execute();
    return $requetePrepare->fetchAll();
}
```

Création de la méthode de classe « getLesMoisEtat » :

```php
/**
 * Retourne les mois pour lesquels un visiteur a une fiche de frais à l'état
 * souhaité.
 *
 * @param String $idVisiteur ID du visiteur.
 * @param String $etatA premier état souhaité.
 * @param String $etatB second état souhaité.
 *
 * @return Array un tableau associatif de clé un mois -aaaamm- et de valeurs
 *         l'année et le mois correspondant.
 */
public function getLesMoisEtat($idVisiteur, $etatA, $etatB)
{
    if ($etatB == null) {
        $requetePrepare = PdoGSB::$monPdo->prepare(
            'SELECT fichefrais.mois AS mois FROM fichefrais '
            . 'WHERE fichefrais.idvisiteur = :unIdVisiteur '
            . 'AND fichefrais.idetat = :unEtatA ORDER BY '
            . 'fichefrais.mois desc'
        );
        $requetePrepare->bindParam(':unEtatA', $etatA, PDO::PARAM_STR);
    } else {
        $requetePrepare = PdoGSB::$monPdo->prepare(
            'SELECT fichefrais.mois AS mois FROM fichefrais '
            . 'WHERE fichefrais.idvisiteur = :unIdVisiteur '
            . 'AND (fichefrais.idetat = :unEtatA '
            . 'OR fichefrais.idetat = :unEtatB) '
            . 'ORDER BY fichefrais.mois desc'
        );
        $requetePrepare->bindParam(':unEtatA', $etatA, PDO::PARAM_STR);
        $requetePrepare->bindParam(':unEtatB', $etatB, PDO::PARAM_STR);
    }
    $requetePrepare->bindParam(':unIdVisiteur', $idVisiteur, PDO::PARAM_STR);
    $requetePrepare->execute();
    $lesMois = array();
    while ($laLigne = $requetePrepare->fetch()) {
        $mois = $laLigne['mois'];
        $numAnnee = substr($mois, 0, 4);
        $numMois = substr($mois, 4, 2);
        $lesMois[] = array(
            'mois' => $mois,
            'numAnnee' => $numAnnee,
            'numMois' => $numMois
        );
    }
    return $lesMois;
}
```

Création de la méthode « creerListeNomPrenom » :

```php
/**
 * Retourne un tableau associatif contenant nom et prénom de visiteur.
 *
 * @param Array $liste tableau associatif contenant id, nom et prénom
 * de visiteur.
 * @return Array liste de nom et prénom sous la forme Nom Prénom
 */
function creerListeNomPrenom($liste)
{
    for ($i=0;$i<count($liste);$i++) {
    $array = $liste[$i];
    $listeNomPrenom[] = $array['nom'] . ' ' . $array['prenom'];
    }
    return $listeNomPrenom;
}
```

Création de la méthode « extraireListe » :

```php
/**
 * Extrait une liste d'un tableau associatif.
 *
 * @param Array $array tableau associatif.
 * @param String $cle clé du tableau associatif.
 * @return Array retourne une liste.
 */
function extraireListe($array, $cle)
{
    foreach($array as $element) {
        $liste = null;
        if (is_array($element[$cle])) {
            $fin = count($element[$cle]);
            for($i=0;$i<$fin;$i++) {
                $liste[] = $element[$cle][$i];
            }
            $listeFinale[] = $liste;
        } else {
            $listeFinale[] = $element[$cle];
        }
    }
    return $listeFinale;
}
```

Création de la méthode de classe « rechercheListeMois » :

```php
/**
 * Recherche les mois correspondants au visiteur présent dans la liste de visiteur.
 *
 * @param Array $listeVisiteur liste de visiteurs.
 * @param String $uc sélection de l'utilisateur.
 *
 * @return Array un tableau de mois.
 */
public function rechercheListeMois($listeVisiteur, $uc) {
    $listeMois = null;
    foreach ($listeVisiteur as $valeur) {
        if ($uc == 'gererFrais') {
            $listeMois[] = formatMois($this->getLesMoisEtat($valeur,'CL',null));
        } else if ($uc == 'etatFrais') {
            $listeMois[] = formatMois($this->getLesMoisEtat($valeur,'VA','PM'));
        }
    }
    return $listeMois;
}
```

Création de la méthode de classe « getLesMoisEtat » :

```php
/**
 * Retourne les mois pour lesquels un visiteur a une fiche de frais à l'état
 * souhaité.
 *
 * @param String $idVisiteur ID du visiteur.
 * @param String $etatA      premier état souhaité.
 * @param String $etatB      second état souhaité.
 *
 * @return Array un tableau associatif de clé un mois -aaaamm- et de valeurs
 *         l'année et le mois correspondant.
 */
public function getLesMoisEtat($idVisiteur, $etatA, $etatB)
{
    if ($etatB == null) {
        $requetePrepare = PdoGSB::$monPdo->prepare(
            'SELECT fichefrais.mois AS mois FROM fichefrais '
            . 'WHERE fichefrais.idvisiteur = :unIdVisiteur '
            . 'AND fichefrais.idetat = :unEtatA ORDER BY '
            . 'fichefrais.mois desc'
        );
        $requetePrepare->bindParam(':unEtatA', $etatA, PDO::PARAM_STR);
    } else {
        $requetePrepare = PdoGSB::$monPdo->prepare(
            'SELECT fichefrais.mois AS mois FROM fichefrais '
            . 'WHERE fichefrais.idvisiteur = :unIdVisiteur '
            . 'AND (fichefrais.idetat = :unEtatA '
            . 'OR fichefrais.idetat = :unEtatB) '
            . 'ORDER BY fichefrais.mois desc'
        );
        $requetePrepare->bindParam(':unEtatA', $etatA, PDO::PARAM_STR);
        $requetePrepare->bindParam(':unEtatB', $etatB, PDO::PARAM_STR);
    }
    $requetePrepare->bindParam(':unIdVisiteur', $idVisiteur, PDO::PARAM_STR);
    $requetePrepare->execute();
    $lesMois = array();
    while ($laLigne = $requetePrepare->fetch()) {
        $mois = $laLigne['mois'];
        $numAnnee = substr($mois, 0, 4);
        $numMois = substr($mois, 4, 2);
        $lesMois[] = array(
            'mois' => $mois,
            'numAnnee' => $numAnnee,
            'numMois' => $numMois
        );
    }
    return $lesMois;
}
```

Création de la méthode « formatMois » :

```php
/**
 * Réorganise dans une liste les dates de type -aaaamm- en type -mm / aaaa-.
 *
 * @param Array $liste liste de date à formater.
 * @return Array liste date formatées.
 */
function formatMois($liste)
{
    foreach ($liste as $value) {
        $listeFormate[] = $value['numMois'] . ' / ' . $value['numAnnee'];
    }
    return $listeFormate;
}
```

Création de la méthode « verificationIndex » :

```php
/**
 * Vérifie si l'index ne dépasse pas le nombre d'élement d'une liste.
 *
 * @param array $liste liste dont on doit vérifier le nombre d'élément
 * @param int $index index à vérifier
 *
 * @return int l'index si sa valeur ne dépasse pas le nombre d'élément
 * de la liste sinon retourne 0.
 */
function verificationIndex($liste, $index)
{
    if ($index >= count($liste)) {
        $index = 0;
    }
    return $index;
}
```

Création de la vue « v_choixVisiteurMois » :

```
<div class="row">
    <form method="post" id='form'
          action="index.php?uc=<?php if ($uc == 'gererFrais') { ?>gererFrais&action=saisirFrais<?php
          } else if ($uc == 'etatFrais') { ?>etatFrais&action=voirEtatFrais<?php } ?>"
          role="form">
        <label for='choixUtilisateur'>Choisir le visiteur: </label>
        <select name='indexListeNom' id='choixUtilisateur' onChange='envoi()'>
            <?php
                $i = 0;
                foreach($listeNomPrenomVisiteur['nomPrenom'] as
                    $nom=>$unNomPrenomVisiteur) {
                    if (($indexListeNom == $i)) {
                    echo '<option value=' . $i . ' selected >'
                    . $unNomPrenomVisiteur . '</option>';
                    } else {
                    echo '<option value=' . $i . '>' . $unNomPrenomVisiteur
                    . '</option>';
                    }
                    $i += 1;
                }
            ?>
        </select>
        <label for='choixMois'>Mois: </label>
        <select name='indexListeMois' id='choixMois' onChange='envoi()'>
            <?php
            $j = 0;
            foreach($listeNomPrenomVisiteur['mois'][$indexListeNom] as
                    $cle=>$value) {
                if ($indexListeMois == $j) {
                    echo '<option value=' . $j . ' selected >'
                    . $value . '</option>';
                } else {
                    echo '<option value=' . $j . '>' . $value
                    . '</option>';
                }
                $j += 1;
            }
            ?>
        </select>
    </form>
</div>
<?php
```

Modification de l'aiguilleur « index » :

```
case 'gererFrais':
    if ($_SESSION['utilisateur'] == 'comptable') {
        include 'controleurs/c_comptable.php';
        if ($listeDeVisiteur != null) {
            include 'controleurs/c_gererFrais.php';
        }
    } else {
        include 'controleurs/c_gererFrais.php';
    }
    break;
```

Modification du contrôleur « c_gererFrais » :

```
if ($_SESSION['utilisateur'] == 'visiteur') {
    $idVisiteur = $_SESSION['idVisiteur'];
    $mois = getMois(date('d/m/Y'));
    $numAnnee = substr($mois, 0, 4);
    $numMois = substr($mois, 4, 2);
}else{
    $nbJustificatifs = $pdo->getNbJustificatifs($idVisiteur,$mois);
}
```

```
case 'validerMajFraisForfait':
    $lesFrais = filter_input(INPUT_POST, 'lesFrais', FILTER_DEFAULT, FILTER_FORCE_ARRAY);
    if (lesQteFraisValides($lesFrais)) {
        $pdo->majFraisForfait($idVisiteur, $mois, $lesFrais);
        if ($_SESSION['utilisateur'] == 'comptable') {
            ajouterReussite('Les frais ont été mis à jour.');
            include 'vues/v_succes.php';
        }
    } else {
        ajouterErreur('Les valeurs des frais doivent être numériques');
        include 'vues/v_erreurs.php';
    }
    break;
```

```
case 'majFraisHorsForfait':
    $lesFrais = filter_input(INPUT_POST, 'lesFrais', FILTER_DEFAULT, FILTER_FORCE_ARRAY);
    $indexId = rechercheBoutonUtilise($lesFrais['id']);
    if (is_array($lesFrais['id'])) {
    $idFrais = $lesFrais['id'][$indexId];
    $dateFrais = $lesFrais['date'][$indexId];
    $libelle = $lesFrais['libelle'][$indexId];
    $montant = $lesFrais['montant'][$indexId];
    } else {
        $idFrais = $lesFrais['id'];
        $dateFrais = $lesFrais['date'];
        $libelle = $lesFrais['libelle'];
        $montant = $lesFrais['montant'];
    }
    valideInfosFrais($dateFrais, $libelle, $montant);
    if (nbErreurs() != 0) {
        include 'vues/v_erreurs.php';
    } else {
        $pdo->majFraisHorsForfait(
            $idFrais,
            $dateFrais,
            $libelle,
            $montant
        );
        ajouterReussite('Les frais ont été mis à jour.');
        include 'vues/v_succes.php';
    }
    break;
```

```
case 'reporterFrais':
    $idFrais = filter_input(INPUT_GET, 'idFrais', FILTER_SANITIZE_STRING);
    $libelleAModifie = filter_input(INPUT_GET, 'libelle', FILTER_SANITIZE_STRING);
    $libelleChange = verificationLongueurChaine($libelleAModifie, 30);
    $moisSuivant = moisSuivant($mois);
    if ($pdo->estPremierFraisMois($idVisiteur,$moisSuivant)) {
        $pdo->creeNouvellesLignesFrais($idVisiteur, $moisSuivant);
    }
    $pdo->reporterFraisHorsForfait($idFrais, $libelleChange, $moisSuivant);
    break;
case 'validerFicheDeFrais':
    if (nbErreurs() != 0) {
        include 'vues/v_erreurs.php';
    } else {
        $pdo->majNbJustificatifs($idVisiteur, $mois, $nbJustificatifs);
        $pdo->majEtatFicheFrais($idVisiteur, $mois, 'VA');
        $indexListeNom = 0;
        $indexListeMois = 0;
    }
    break;
```

```
default:
    ajouterErreur('Une erreur est survenue. Merci de contacter votre administrateur');
    include 'vues/v_erreurs.php';
    break;
}
if ($uc != 'Erreur') {
    $lesFraisHorsForfait = $pdo->getLesFraisHorsForfait($idVisiteur, $mois);
    $lesFraisForfait = $pdo->getLesFraisForfait($idVisiteur, $mois);
    require 'vues/v_listeFraisForfait.php';
    require 'vues/v_listeFraisHorsForfait.php';
}
```

Création de la méthode de classe « majNbJustificatifs » :

```php
/**
 * Met à jour le nombre de justificatifs de la table ficheFrais
 * pour le mois et le visiteur concerné
 *
 * @param String  $idVisiteur      ID du visiteur
 * @param String  $mois            Mois sous la forme aaaamm
 * @param Integer $nbJustificatifs Nombre de justificatifs
 *
 * @return null
 */
public function majNbJustificatifs($idVisiteur, $mois, $nbJustificatifs)
{
    $requetePrepare = PdoGsB::$monPdo->prepare(
        'UPDATE fichefrais '
        . 'SET nbjustificatifs = :unNbJustificatifs '
        . 'WHERE fichefrais.idvisiteur = :unIdVisiteur '
        . 'AND fichefrais.mois = :unMois'
    );
    $requetePrepare->bindParam(
        ':unNbJustificatifs',
        $nbJustificatifs,
        PDO::PARAM_INT
    );
    $requetePrepare->bindParam(':unIdVisiteur', $idVisiteur, PDO::PARAM_STR);
    $requetePrepare->bindParam(':unMois', $mois, PDO::PARAM_STR);
    $requetePrepare->execute();
}
```

Création de la méthode « ajouterReussite » :

```php
/**
 * Ajoute le libellé d'une réussite au tableau des réussites
 *
 * @param String $msg Libellé de la réussite
 *
 * @return null
 */
function ajouterReussite($msg)
{
    if (!isset($_REQUEST['reussites'])) {
        $_REQUEST['reussites'] = array();
    }
    $_REQUEST['reussites'][] = $msg;
}
```

Création de la méthode « moisSuivant » :

```php
/**
 * Recherche le mois suivant.
 *
 * @param String $mois le mois précédent.
 * @return string le mois suivant.
 */
function moisSuivant($mois)
{
    $numAnnee = intval(substr($mois,0,4));
    $numMois = intval(substr($mois, 5));
    if ($numMois + 1 > 12) {
        $moisSuivant = strval($numAnnee + 1) . '01';
    } else if (strlen($numMois + 1) < 2) {
        $moisSuivant = strval($numAnnee) . '0' . strval($numMois + 1);
    } else {
        $moisSuivant = strval($numAnnee) . strval($numMois + 1);
    }
    return $moisSuivant;
}
```

Création de la méthode « verificationLongueurChaine » :

```php
/**
 * Vérifie la longueur d'une chaine et tronque si celle-ci dépasse la longueur
 * voulu.
 *
 * @param String $chaine chaine à contrôler.
 * @param Integer $longueur longueur de la chaine souhaité.
 * @return String retourne la chaine.
 */
function verificationLongueurChaine($chaine, $longueur)
{
    if (strlen($chaine) > $longueur) {
        $chaine = substr($chaine, 0, $longueur);
    }
    return $chaine;
}
```

Création de la méthode « rechercheBoutonUtilise » :

```php
/**
 * Recherche le bouton sur lequel l'utilisateur a appuyé.
 *
 * @param Array $listeId liste d'id des frais hors forfait.
 *
 * @return Integer index de la liste correspondant au bouton appuyé.
 */
function rechercheBoutonUtilise($listeId)
{
    $i = 0;
    while (!isset($_POST[$listeId[$i]])) {
        $i++;
    }
    return $i;
}
```

Création de la vue « v_success » :

```php
<div class="alert alert-success" role="success">
    <?php
    foreach ($_REQUEST['reussites'] as $reussite) {
        echo '<p>' . htmlspecialchars($reussite) . '</p>';
    }
    ?>
</div>
```

Création de la méthode de classe « reporterFraisHorsForfait » :

```php
/**
 * Modifie le libellé et le mois d'un frais hors forfait.
 *
 * @param String $idFrais id du frais.
 * @param String $libelle libellé du frais
 * @param String $mois    le nouveau mois du frais.
 *
 * @return null
 */
public function reporterFraisHorsForfait($idFrais, $libelle, $mois)
{
    $requetePrepare = PdoGSB::$monPdo->prepare(
        'UPDATE lignefraishorsforfait '
        . 'SET lignefraishorsforfait.libelle = :unLibelle, '
        . 'mois = :unMois WHERE lignefraishorsforfait.id = :unIdFrais'
    );
    $requetePrepare->bindParam(':unIdFrais', $idFrais, PDO::PARAM_STR);
    $requetePrepare->bindParam(':unLibelle', $libelle, PDO::PARAM_STR);
    $requetePrepare->bindParam(':unMois', $mois, PDO::PARAM_STR);
    $requetePrepare->execute();
}
```

Création de la méthode « majEtatFicheFrais » :

```php
/**
 * Modifie l'état et la date de modification d'une fiche de frais.
 * Modifie le champ idEtat et met la date de modif à aujourd'hui.
 *
 * @param String $idVisiteur ID du visiteur
 * @param String $mois       Mois sous la forme aaaamm
 * @param String $etat       Nouvel état de la fiche de frais
 *
 * @return null
 */
public function majEtatFicheFrais($idVisiteur, $mois, $etat)
{
    if ($etat == 'VA') {
        $this->calculMontantValide($idVisiteur,$mois);
    }
    $requetePrepare = PdoGSB::$monPdo->prepare(
        'UPDATE ficheFrais '
        . 'SET idetat = :unEtat, datemodif = now() '
        . 'WHERE fichefrais.idvisiteur = :unIdVisiteur '
        . 'AND fichefrais.mois = :unMois'
    );
    $requetePrepare->bindParam(':unEtat', $etat, PDO::PARAM_STR);
    $requetePrepare->bindParam(':unIdVisiteur', $idVisiteur, PDO::PARAM_STR);
    $requetePrepare->bindParam(':unMois', $mois, PDO::PARAM_STR);
    $requetePrepare->execute();
}
```

Création de la méthode de classe « supprimerFraisHorsForfait » :

```php
/**
 * Supprime le frais hors forfait dont l'id est passé en argument
 *
 * @param String $idFrais ID du frais
 *
 * @return null
 */
public function supprimerFraisHorsForfait($idFrais)
{
    $requetePrepare = PdoGSB::$monPdo->prepare(
        'DELETE FROM lignefraishorsforfait '
        . 'WHERE lignefraishorsforfait.id = :unIdFrais'
    );
    $requetePrepare->bindParam(':unIdFrais', $idFrais, PDO::PARAM_STR);
    $requetePrepare->execute();
}
```

Ajout du fichier Javascript « fonctionDynamique » et de la méthode « envois » qui permet l'actualisation de la page au changement de sélection dans la liste déroulante.

```javascript
/**
 * Procédure qui permet d'envoyer un formulaire.
 *
 * @return null
 */
function envoi() {
    var formulaire = document.getElementById('form');
    formulaire.submit();
}
```

Ajout de l'appel de la méthode envois à la fin de la vue « v_choixVisiteurMois ». Pour actualiser lorsque plus aucune fiche n'existe dans l'état voulu.

```php
if (($action == 'validerFicheDeFrais' && nbErreurs()== 0) ||
        $action == 'majEtatRembourse' || $action == 'majEtatMisePaiement') {
?><script> envoi(); </script>
<?php
}
```

Modification de la vue « v_listeFraisForfait » :

```
<div class="row">

    <h2 <?php if ($_SESSION['utilisateur']=="comptable") {
        echo "class=comptable";
        }
        ?>>
        <?php if ($_SESSION['utilisateur']== "visiteur") { ?>
        Renseigner ma fiche de frais du mois
        <?php echo $numMois . '-' . $numAnnee;
        } else {
        ?>
        Valider la fiche de frais
        <?php
        }
        ?>
    </h2>
```

```
<button class="btn btn-success" type="submit">
    <?php if ($_SESSION['utilisateur'] == 'visiteur') { ?>
        Ajouter
    <?php } else if ($_SESSION['utilisateur'] == 'comptable') { ?>
        Corriger
    <?php } ?>
</button>
    <?php if ($_SESSION['utilisateur'] == 'visiteur') { ?>
        <button class="btn btn-danger" type="reset">
        Effacer
    <?php } else if ($_SESSION['utilisateur'] == 'comptable') { ?>
        <button class="btn btn-danger" onclick='envoi()' type="button">
        Réinitialiser
    <?php } ?>
</button>
```

Modification de la vue « v_listeFraisHorsForfait » :

```
<div class="row">
    <div class="panel panel-info tableau-<?php
    echo $_SESSION['utilisateur']?>">
        <div class="panel-heading">Descriptif des éléments hors forfait</div>
        <form action="index.php?uc=gererFrais&action=majFraisHorsForfait"
            method="post" role="form">
            <?php if ($_SESSION['utilisateur'] == 'comptable') { ?>
                <input type="hidden" name="indexListeNom" value="<?php echo $indexListeNom ?>">
                <input type="hidden" name="indexListeMois" value="<?php echo $indexListeMois ?>">
            <?php
            }
            ?>
```

```php
foreach ($lesFraisHorsForfait as $unFraisHorsForfait) {
    $libelle = htmlspecialchars($unFraisHorsForfait['libelle']);
    $date = $unFraisHorsForfait['date'];
    $montant = $unFraisHorsForfait['montant'];
    $id = $unFraisHorsForfait['id'];
    if ($_SESSION['utilisateur'] == 'comptable') {
    ?>
    <tbody>
        <tr>
            <input type="hidden" name="lesFrais[id]"
                   value="<?php echo $id; ?>">
        <td>
            <input type="text" name="lesFrais[date]" size="5"
                   class="form-control" maxlength="10"
                   value="<?php echo $date; ?>"></td>
        <td>
            <input type="text" name="lesFrais[libelle]" size="20"
                   class="form-control" maxlength="30"
                   value="<?php echo $libelle; ?>"></td>
        <td>
            <input type="text" name="lesFrais[montant]" size="5"
                   class="form-control" maxlength="6"
                   value="<?php echo $montant; ?>"></td>
        <td>
            <button class="btn btn-success" type="submit" name="<?php echo $id
                ?>">Corriger</button>
            <button class="btn btn-danger" onclick="envoi()" type="button">Réinitialiser</button>
            <a href="index.php?uc=gererFrais&action=reporterFrais&idFrais=<?php echo $id;
            ?>&libelle=REFUSER : <?php echo $libelle;?>&indexListeNom=<?php echo $indexListeNom;
            ?>&indexListeMois=<?php echo $indexListeMois;
            ?>"
            onclick="return confirm('Voulez-vous vraiment reporter ce frais?');">Reporter ce frais</a></td>
        </tr>
        <?php
```

```php
<div class="row">
    <form method="post"
          action="index.php?uc=gererFrais&action=validerFicheDeFrais"
          role="form">
        <input type="hidden" name="indexListeNom" value="<?php echo $indexListeNom ?>">
        <input type="hidden" name="indexListeMois" value="<?php echo $indexListeMois ?>">
        <label for="nbJustificatif">Nombre de justificatifs :</label>
        <input type="text" id="nbJustificatifs"
                           name="nbJustificatifs"
                           size="1" maxlength="2"
                           value="<?php echo $nbJustificatifs ?>"
                           class="form-group">
        <button class="btn btn-success" type="submit">Valider la fiche</button>
        <button class="btn btn-danger" onclick="envoi()" type="button">Réinitialiser</button>
    </form>
</div>
```

Création de la méthode « valideJustificatifs » :

```php
/**
 * Valide que le nombre de justificatifs n'est pas nul et bien numérique.
 *
 * @param Integer $justificatif
 *
 * @return null
 */
function valideJustificatifs($justificatif)
{
    if (!is_numeric($justificatif)) {
        ajouterErreur('Le champ nombre de justificatif doit être numérique.');
    }
}
```

Création de la méthode de classe « majNbJustificatifs » :

```php
/**
 * Met à jour le nombre de justificatifs de la table ficheFrais
 * pour le mois et le visiteur concerné
 *
 * @param String  $idVisiteur      ID du visiteur
 * @param String  $mois            Mois sous la forme aaaamm
 * @param Integer $nbJustificatifs Nombre de justificatifs
 *
 * @return null
 */
public function majNbJustificatifs($idVisiteur, $mois, $nbJustificatifs)
{
    $requetePrepare = PdoGsB::$monPdo->prepare(
        'UPDATE fichefrais '
        . 'SET nbjustificatifs = :unNbJustificatifs '
        . 'WHERE fichefrais.idvisiteur = :unIdVisiteur '
        . 'AND fichefrais.mois = :unMois'
    );
    $requetePrepare->bindParam(
        ':unNbJustificatifs',
        $nbJustificatifs,
        PDO::PARAM_INT
    );
    $requetePrepare->bindParam(':unIdVisiteur', $idVisiteur, PDO::PARAM_STR);
    $requetePrepare->bindParam(':unMois', $mois, PDO::PARAM_STR);
    $requetePrepare->execute();
}
```

Modification de l'aiguilleur « index » :

```
case 'etatFrais':
    if ($_SESSION['utilisateur'] == 'comptable') {
        include 'controleurs/c_comptable.php';
        if ($listeDeVisiteur != null) {
            include 'controleurs/c_etatFrais.php';
        }
    } else {
        include 'controleurs/c_etatFrais.php';
    }
    break;
```

Modification du contrôleur « c_etatFrais » :

```
if ($_SESSION['utilisateur'] == 'visiteur') {
    $action = filter_input(INPUT_GET, 'action', FILTER_SANITIZE_STRING);
    $idVisiteur = $_SESSION['idVisiteur'];
} else {
    if ($action == 'selectionnerMois') {
        $action = 'voirEtatFrais';
    }
}
```

```
case 'voirEtatFrais':
    if ($_SESSION['utilisateur'] == 'visiteur') {
        $leMois = filter_input(INPUT_POST, 'lstMois', FILTER_SANITIZE_STRING);
        $lesMois = $pdo->getLesMoisDisponibles($idVisiteur);
        $moisASelectionner = $leMois;
        include 'vues/v_listeMois.php';
    } else if ($_SESSION['utilisateur'] == 'comptable') {
        $leMois = $mois;
    }
    $lesFraisHorsForfait = $pdo->getLesFraisHorsForfait($idVisiteur, $leMois);
    $lesFraisForfait = $pdo->getLesFraisForfait($idVisiteur, $leMois);
    $lesInfosFicheFrais = $pdo->getLesInfosFicheFrais($idVisiteur, $leMois);
    $numAnnee = substr($leMois, 0, 4);
    $numMois = substr($leMois, 4, 2);
    $idEtat = $lesInfosFicheFrais['idEtat'];
    $libEtat = $lesInfosFicheFrais['libEtat'];
    $montantValide = $lesInfosFicheFrais['montantValide'];
    $nbJustificatifs = $lesInfosFicheFrais['nbJustificatifs'];
    $dateModif = dateAnglaisVersFrancais($lesInfosFicheFrais['dateModif']);
    include 'vues/v_etatFrais.php';
    break;
case 'majEtatMisePaiement':
    $pdo->majEtatFicheFrais($idVisiteur, $mois, 'PM');
    $indexListeNom = 0;
    $indexListeMois = 0;
    break;
case 'majEtatRembourse':
    $pdo->majEtatFicheFrais($idVisiteur, $mois, 'RB');
    $indexListeNom = 0;
    $indexListeMois = 0;
    break;
}
```

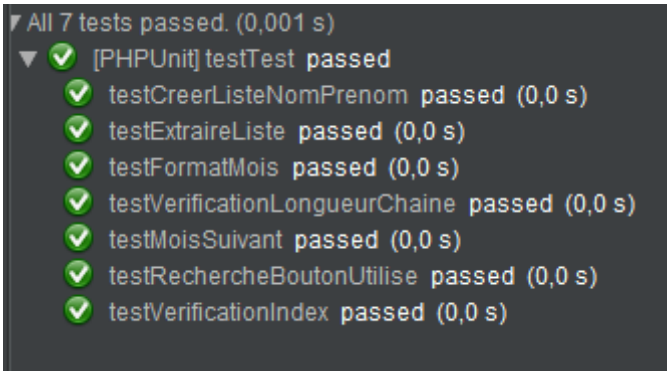
Modification de la vue « v_etatFrais » :

```
<?php switch ($idEtat) {
    case 'VA':?>
        <form action="index.php?uc=etatFrais&action=majEtatMisePaiement"
              method="post" role="form">
            <input type="hidden" name="indexListeNom" value="<?php echo $indexListeNom ?>">
            <input type="hidden" name="indexListeMois" value="<?php echo $indexListeMois ?>">
            <button class="btn btn-success" type="submit">Fiche Mise en paiement</button>
        </form>
    <?php break;
    case 'PM': ?>
        <form action="index.php?uc=etatFrais&action=majEtatRembourse"
              method="post" role="form">
            <input type="hidden" name="indexListeNom" value="<?php echo $indexListeNom ?>">
            <input type="hidden" name="indexListeMois" value="<?php echo $indexListeMois ?>">
            <button class="btn btn-success" type="submit">Fiche remboursée</button>
        </form>
    <?php break;
    default:
    break;
}
?>
```

5. Tests unitaires

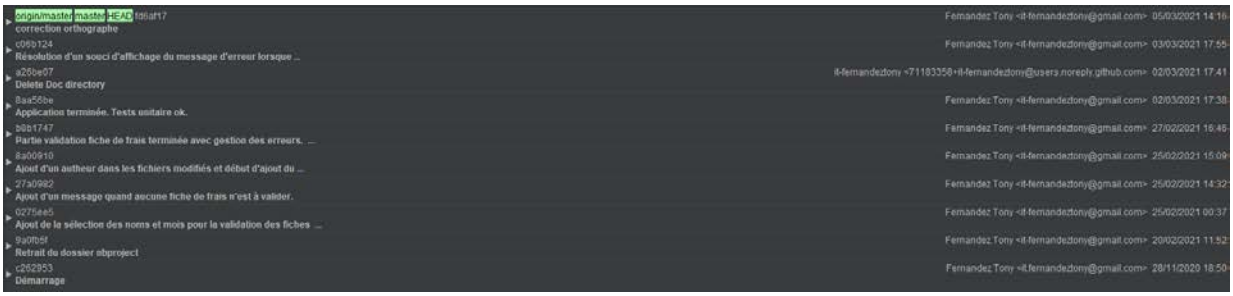
Deux fichiers ont été créés pour ces tests :

- testFonction.php
- testFonctionTest.php



6. Gestion de version

Git :



Github :

https://github.com/it-fernandeztony/GSB-project.git

7. Compétences

A1.1.1 , Analyse du cahier des charges d'un service à produire.

A1.1.3 , Étude des exigences liées à la qualité attendue d'un service.

A1.4.2 , Évaluation des indicateurs de suivi d'un projet et justification des écarts.

A1.4.3 , Gestion des ressources.

A4.1.1 , Proposition d'une solution applicative.

A4.1.2 , Conception ou adaptation de l'interface utilisateur d'une solution applicative.

A4.1.3 , Conception ou adaptation d'une base de données.

A4.1.6 , Gestion d'environnements de développement et de test.

A4.1.9 , Rédaction d'une documentation technique.

A4.1.10 , Rédaction d'une documentation d'utilisation.

A5.2.1 , Exploitation des référentiels, normes et standards adoptés par le prestataire.

A5.2.3 , Repérage des compléments de formation ou d'auto-formation ...