

GSB Comptable Projet

Application de Gestion de frais



Sommaire Projet

I - Présentation	3
II – Le Menu	4
III – Index.PHP.....	5
IV – Clôture des fiches de frais.....	7
V – Validation des fiches de frais.....	9
VI – Paiement des fiches de frais.....	17
VII – Mot de passe quel algorithme ?.....	18
VIII - Documentation	21
IX - Hébergement	22

I – Présentation

Pour ce projet j'ai dû créer une application web pour le compte des laboratoires pharmaceutiques Galaxy Swiss Bourdin consistant à la gestion des notes de frais de salariés, destinées au comptable de l'entreprise.

Ceci comprenant 2 parties, l'une pour les employés pour venir remplir leurs notes de frais, et la deuxième pour la validation et paiement de ces fameuses notes. Je me suis donc occupé de la deuxième partie et ce, avec la contrainte du modèle MVC.

Pour cela j'ai utilisé du HTML/CSS pour le front end, et du Php pour le back-end. J'ai travaillé dans un premier temps en local sur un serveur Wamp puis fait la migration vers mon VPS. En ce qui concerne le développement aucun IDE n'était imposé donc ayant pour habitude de travailler avec Visual Studio Code, j'ai naturellement continué avec celui-ci.

Accès au code :

- Code source Github : https://github.com/G-alexis-Fr/gsb_comptable
- Accès au site-web :
 - Accès partie comptable :
 - Identifiant : alexis
 - Mot de passe : gsb
 - Lien : gsbcomptable.g-alexis.com
- Accès au site-web :
 - Accès partie visiteur :
 - Identifiant : dandre
 - Mot de passe : oppg5
 - Lien : gsbvisiteur.g-alexis.com
- Documentation PHP : <http://gsbdocumentation.g-alexis.com>

Contexte du projet : https://github.com/G-alexis-Fr/gsb_comptable/blob/master/GSB-AppliFrais-Fiche%20Descriptive.pdf

- Description de l'entreprise : https://github.com/G-alexis-Fr/gsb_comptable/blob/master/GSB-Organisation.pdf

- Projet GSB Mobile : https://github.com/G-alexis-Fr/gsb_comptable-android

II – Le Menu



Le menu étant le même d'une page à l'autre il était évident que celui-ci soit écrit dans un fichier à part.

Voici donc une partie du code :

```
if ($estConnecte) {  
    ?>  
    <div class="header">  
        <div class="row vertical-align">  
            <div class="col-md-4">  
                <h1>  
                      
                </h1>  
            </div>  
            <div class="col-md-8">  
                <ul class="nav nav-pills pull-right orange" role="tablist">  
                    <li <?php  
                        if (!$uc || $uc == 'accueil') {  
                            ?>  
                                class="active"  
                            <?php  
                                }  
                        ?>>  
                        <a href="index.php">  
                            <span class="glyphicon glyphicon-home"></span>  
                            Accueil  
                        </a>  
                    </li>  
                </ul>  
            </div>  
        </div>  
    }  
}
```

On remarquera l'utilisation du Framework Bootstrap pour rendre le site responsive ainsi que les icônes appelées **glyphicon**.

Une charte graphique nous avait été donnée pour que les sites visiteur et comptable aient le même visuel.

III – Index.php

La page **index.php** est le point de chute de notre site internet, c'est cette page qui gèrera les redirections vers les bonnes pages.

```
require_once 'includes/fct.inc.php';
require_once 'includes/class.pdogsdb.inc.php';

session_start();

$pdo = PdoGsb::getPdoGsb();

$estConnecte = estConnecte();
require 'vues/v_entete.php';

$uc = filter_input(INPUT_GET, 'uc', FILTER_SANITIZE_STRING);

if ($uc && !$estConnecte) {
    $uc = 'connexion';
} elseif (empty($uc)) {
    $uc = 'accueil';
}

switch ($uc) {
    case 'connexion':
        include 'controleurs/c_connexion.php';
        break;
    case 'accueil':
        include 'controleurs/c_accueil.php';
        break;
    case 'cloture':
        include 'controleurs/c_cloture.php';
        break;
    case 'validation':
        include 'controleurs/c_validation.php';
        break;
    case 'paiement':
        include 'controleurs/c_paiement.php';
        break;
    case 'deconnexion':
        include 'controleurs/c_deconnexion.php';
        break;
}
require 'vues/v_pied.php';
```

On remarque dans un premier l'appel des fichiers requis en l'occurrence la classe PDO et le fichier ou sont localisées les fonctions de mon programme. Puis nous lançons une session pour que le navigateur sache lorsqu'un utilisateur se connecte.

Si l'utilisateur est connecté alors nous le renvoyons vers **Accueil** sinon vers **Connexion**.

Vue Connexion :




Identification utilisateur

 Alexis



Se connecter


Vue Accueil :





[Accueil](#) [Clôture Automatique](#) [Validation](#) [Paiement](#) [Déconnexion](#)

Gestion des frais Comptable : comptable alexis

Navigation

 Clôture automatique

 Validation

 Paiement

Il y a :

- aucune fiche(s) à clôturer

- 26 fiche(s) à valider

- 28 fiche(s) à payer

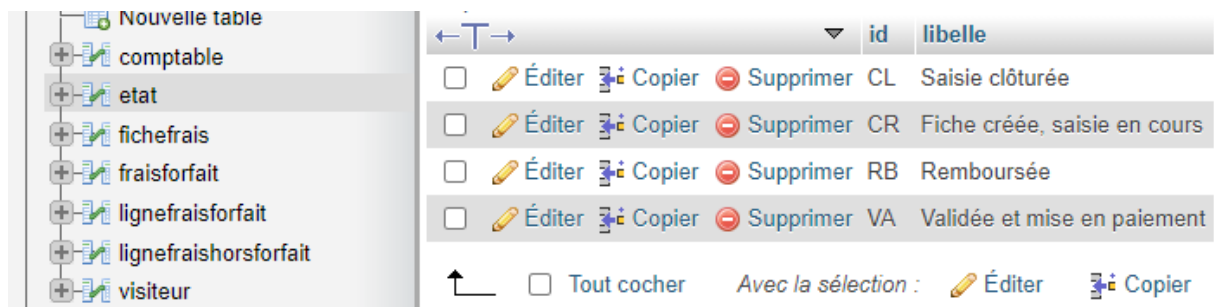
IV– Clôture des fiches de frais

Pour nous aider dans le remplissage de feuille de frais, nous avons à disposition des fonctions qui permettaient de créer des fiches de frais en cours de création et des fiches validées avec comme identifiants CR pour création et VA pour validée.

Cependant avant même de pouvoir manipuler ces fiches, il fallait pouvoir les clôturer.

J'ai donc dû créer une fonction permettant de récupérer dans la base de données les fiches ayant pour statut (idetat) 'CR'.

Vue des différents statut disponibles :



Le mois est requis en paramètre permettant une recherche ciblée.

```
public function getVisiteursPasClos($mois)
{
    $requetePrepare = PdoGSB::$monPdo->prepare(
        "SELECT idvisiteur FROM fichefrais WHERE idetat = 'CR' "
        . "AND mois = :unMois"
    );
    $requetePrepare->bindParam('unMois', $mois, PDO::PARAM_STR);
    $requetePrepare->execute();
    return $requetePrepare->fetchAll();
}
```

Nous avons un contrôleur qui lui permet de vérifier les fiches du mois précédent à clôturer et en fonction de si besoin elle clôture ou alors affiche un message disant qu'aucunes fiches ne doivent être clôturées.

```
$mois = getMois(date('d/m/Y'));
$moisPrec = getMoisPrec($mois);
$visiteurs = $pdo->getVisiteursPasClos($moisPrec);
$nbrFicheCloturer = count($visiteurs);

$numMoisPrec = substr($moisPrec, 4, 2);
$numAnnee = substr($moisPrec, 0, 4);
```

```

$action = filter_input(INPUT_GET, 'action', FILTER_SANITIZE_STRING);

// Si fiche en attente de clôture nous affichons message d'erreur sinon on clôture
switch ($action) {
    case 'cloturer':
        if ($nbrFicheCloturer === 0) {

            Erreur("Aucune fiche visiteur est en attente de clôture ! ");
            include 'vues/v_erreurs.php';
        } else {
            include 'vues/v_clotureAuto.php';
        }
        break;

    case 'succesCloture':
        $nbrCloture = count($visiteurs);
        foreach ($visiteurs as $visiteur) {
            $pdo->majEtatFicheFrais($visiteur['idvisiteur'], $moisPrec, 'CL');
            $pdo->creeNouvellesLignesFrais($visiteur['idvisiteur'], $mois);
        }
        include 'vues/v_termCloture.php';
        break;
    default:
        include 'vues/v_accueil.php';
}

```


V - Validation des fiches de frais



Sélectionner un visiteur et un mois :

Visiteurs :

Mois :

Valider

Effacer

Avant même de valider une fiche, faut d'abord la sélectionner en choisissant quel visiteur et que le mois de la fiche de frais.

Pour cela, il faut faire une requête avec comme paramètre, les fiches à valider et le mois concerné.

```
// Permet de retourner les infos dans listes déroulantes
$lesVisiteurs = $pdo->getLesVisiteursAValider();
if (!is_array($lesVisiteurs) || count($lesVisiteurs) === 0) {
    Erreur("Aucune fiche visiteur à valider !");
    include 'vues/v_erreurs.php';
} else {
    /**
     * Récupération des mois à valider pour tous les visiteurs
     * sélectionnés au dessus
     */
    foreach ($lesVisiteurs as $unVisiteur) {
        $lesMois[] = $unVisiteur['mois'];
    }

    // Suppression des doublons

    $lesMois = array_unique($lesMois);
    $visiteurASelectionner = $lesVisiteurs[0];
    $moisASelectionner = $lesMois[0];
    $lesVisiteurs = unique_multidim_array($lesVisiteurs, 'id');
}
include 'vues/v_selectVisiteurMois.php';
break;
```

Après avoir sélectionné un visiteur et un mois nous nous retrouvons donc sur la page suivante avec le détail des frais forfaitisés ainsi que ceux Hors Forfait. Et la possibilité de corriger ces frais.

Retour à la sélection

Fiche de frais du mois 02-2021 pour Villechalane Louis :

Etat : Saisie clôturée depuis le 20/03/2021

Montant frais forfait : 4 165,16 €

Montant frais hors forfait : 484,00 €

Total en cours : 4 649,16 €

Éléments forfaitisés

Forfait Etape

17

Frais Kilométrique

968

Nuitée Hôtel

19

Repas Restaurant

7

Corriger

Descriptif des éléments hors forfait

Nbre de justificatifs reçus = 6

Date	Libellé	Montant		
06/02/2021	REFUSE Repas avec praticien	47,00	Ce frais a été rejeté	Ce frais a été rejeté
12/02/2021	Voyage SNCF	103,00	Refuser ce frais	Report fiche mois suivant
05/02/2021	Frais vestimentaire/représentation	285,00	Refuser ce frais	Report fiche mois suivant
22/02/2021	Voyage SNCF	57,00	Refuser ce frais	Report fiche mois suivant
18/02/2021	Taxi	39,00	Refuser ce frais	Report fiche mois suivant

Corriger

Valider la fiche de Frais ✓

Avec l'identifiant de l'utilisateur ainsi que le mois il était possible de récupérer toutes les informations nécessaires pour remplir toutes les cases que vous voyez au-dessus.

J'ai par la suite utilisé des méthodes qui étaient disponible en début de projet.

```
// Récupération des informations pour la fiche de frais
    $lesFraisHorsForfait = $pdo-
>getLesFraisHorsForfait($idVisiteur, $leMois);
    $lesFraisForfait = $pdo->getLesFraisForfait($idVisiteur, $leMois);
    $etatFicheVisiteur = $pdo->getEtatFicheFrais($idVisiteur, $leMois);

    if (empty($etatFicheVisiteur)) {
        Erreur("Aucune fiche de frais disponible pour ce visiteur pour ce
mois");
        include 'vues/v_erreurs.php';
    } elseif ($etatFicheVisiteur != 'CL') {
        Erreur("La fiche de ce mois n'est pas à valider pour ce visiteur")
;
        include 'vues/v_erreurs.php';
    } else {
        $lesInfosFicheFrais = $pdo-
>getLesInfosFicheFrais($idVisiteur, $leMois);
        $montantForfait = $pdo->calculFraisForfait($idVisiteur, $leMois);
        $montantHorsForfait = $pdo->calculFraisHF($idVisiteur, $leMois);
        $libEtat = $lesInfosFicheFrais['libEtat'];
        $enCours = $montantForfait + $montantHorsForfait;
        $nbJustificatifs = $lesInfosFicheFrais['nbJustificatifs'];
        $dateModif = dateAnglaisVersFrancais($lesInfosFicheFrais['dateModi
f']);
    }

    include 'vues/v_validEntete.php';
    include 'vues/v_validForfait.php';
    include 'vues/v_validHorsForfait.php';
    include 'vues/v_validFiche.php';
    break;
```

Il m'aura tout de même fallu ajouter une méthode de calcul de frais et ce simplement en multipliant les frais de de la table lignefraisforfait ainsi que ceux de la table fraisforfait.

```
public function calculFraisForfait($idVisiteur, $mois)
{
    $requetePrepare = PdoGsb::$monPdo->prepare(
        "SELECT montant, quantite "
        . "FROM lignefraisforfait "
        . "JOIN fraisforfait "
        . "ON lignefraisforfait.idfraisforfait = fraisforfait.id "
        . "WHERE lignefraisforfait.idvisiteur = :unVisiteur "
        . "AND lignefraisforfait.mois = :unMois"
    );
}
```

```

        $requetePrepare-
>bindParam(':unVisiteur', $idVisiteur, PDO::PARAM_STR);
        $requetePrepare->bindParam(':unMois', $mois, PDO::PARAM_STR);
        $requetePrepare->execute();
        $totalForfait = 0.00;
        while ($donnees = $requetePrepare->fetch()) {
            $totalForfait = $totalForfait + floatval($donnees['montant']) * fl
            oatval($donnees['quantite']);
        }
        return $totalForfait;
    }
}

```

Il nous était demandé de pouvoir refuser les frais Hors Forfait donc une méthode simple permettant de vérifier si le libellé est REFUSE s'il l'est alors return TRUE sinon FALSE et ceci permettra par la suite le calcul total des Hors Forfait.

Voici donc la méthode de refus :

```

public function horsForfaitRefuse($idFrais)
{
    $requetePrepare = PdoGsb::$monPdo->prepare(
        "SELECT libelle "
        . "FROM lignefraishorsforfait "
        . "WHERE id= :idFrais"
    );
    $requetePrepare->bindParam(':idFrais', $idFrais, PDO::PARAM_STR);
    $requetePrepare->execute();
    $libelle = $requetePrepare->fetch();
    $libelleDebut = substr($libelle['libelle'], 0, 6);
    if ($libelleDebut == 'REFUSE') {
        return true;
    } else {
        return false;
    }
}
}

```

Et ici nous additionnons simplement les Hors Forfait qui ont été VALIDES (J'entends par là, le fait que le libellé n'a pas été change à REFUSE)

```
public function calculFraishf($idVisiteur, $mois)
{
    $requetePrepare = PdoGsb::$monPdo->prepare(
        "SELECT id, montant "
        . "FROM lignefraishorsforfait "
        . "WHERE idvisiteur = :unVisiteur "
        . "AND mois = :unMois"
    );
    $requetePrepare-
>bindParam(':unVisiteur', $idVisiteur, PDO::PARAM_STR);
    $requetePrepare->bindParam(':unMois', $mois, PDO::PARAM_STR);
    $requetePrepare->execute();
    $totalHorsForfait = 0.00;
    while ($donnees = $requetePrepare->fetch()) {
        if (!($this->horsForfaitRefuse($donnees['id']))) {
            $totalHorsForfait = $totalHorsForfait + floatval($donnees['mon
tant']);
        }
    }
    return $totalHorsForfait;
}
```

Mentionné plus haut, nous avons pour consigne de pouvoir corriger les frais si une personne s'était trompée par exemple. Nous avons comme contrainte que les saisies entrées par l'utilisateur soient de type entier.

Pour cela j'ai utilisé l'attribut 'number' avec comme minimum 0 et un step de 1 pour chaque input.

```
$idfrais = $unFrais['idfrais'];
$libelle = htmlspecialchars($unFrais['libelle']);
$quantite = $unFrais['quantite']; ?>
<div class="form-group">
    <label for="idfrais"><?php echo $libelle; ?></label>
    <input type="number"
        class="form-control"
        id="idfrais"
        name="lesFrais[<?php echo $idfrais; ?>]"
        min="0" step="1"
        value="<?php echo $quantite; ?>">
```

Date	Libellé	Montant		
06/02/2021	REFUSE Repas avec praticien	47.00	Ce frais a été rejeté	Ce frais a été rejeté
12/02/2021	Voyage SNCF	<input type="text" value="103,00"/>	Refuser ce frais	Report fiche mois suivant
05/02/2021	Frais vestimentaire/représentation	<input type="text" value="285,00"/>	Refuser ce frais	Report fiche mois suivant
22/02/2021	Voyage SNCF	<input type="text" value="57,00"/>	Refuser ce frais	Report fiche mois suivant
18/02/2021	Taxi	<input type="text" value="39,00"/>	Refuser ce frais	Report fiche mois suivant

En ce qui concerne donc le refus, si le comptable clique sur Refuser ce frais alors un une alerte de type **confirm** apparaît :

localhost indique

Voulez-vous refuser ce frais ?

Ceci vous retour un booléen si Ok True, Annuler False et en fonction de ça, nous appellerons la même fonction qu’auparavant pour changer le libellé à REFUSE. (Fin de la page 12)

Puis nous effectuerons un contrôle SI True ou non, voici une partie du code :

```
foreach ($lesFraisHorsForfait as $unFraisHorsForfait) {
    $libelle = checkInput($unFraisHorsForfait['libelle']);
    $date = $unFraisHorsForfait['date'];
    $montant = $unFraisHorsForfait['montant'];
    $id = $unFraisHorsForfait['id'];
    $horsForfaitRefuseFrais = $pdo->horsForfaitRefuse($id);
}

<tr>
    <td> <?php echo $date ?></td>
    <td> <?php echo $libelle ?></td>
    <td><?php if ($horsForfaitRefuseFrais) {
        echo $montant;
    } else { ?>
        <input type="number" step="0.01" name="horsForfait[<?php echo $id; ?>]" value="<?php echo $montant ?>">
    }</td>
</tr>
<?php }>

<td><?php if ($horsForfaitRefuseFrais) { ?>
    <p>Ce frais est rejeté</p>
    <?php } else {
    ?>
    <a href="index.php?uc=validation&action=supprimerFrais&idVisiteur=<?php echo $idVisiteur; ?>
    &leMois=<?php echo $leMois; ?>&idFrais=<?php echo $id; ?>"
    onclick="return confirm('Voulez-vous refuser ce frais ?');">
        Refuser ce frais</a>
    }</td>
</tr>
<td><?php if ($horsForfaitRefuseFrais) { ?>
    <p>Ce frais est rejeté</p>
    <?php } else {
    ?>
    <a href="index.php?uc=validation&action=reporterFrais&idVisiteur=<?php echo $idVisiteur; ?>
    &leMois=<?php echo $leMois; ?>&idFrais=<?php echo $id; ?>"
    onclick="return confirm('Voulez-vous reporter ce frais ?');">
        Report mois suivant</a>
    }</td>
</tr>
```

Quant au contrôleur, il faut lui ajouter le libelle REFUSE voici la partie du contrôleur qui appelle la fonction **refuserFraisHF**.

```
case 'supprimerFrais':

    $idFrais = filter_input(INPUT_GET, 'idFrais', FILTER_SANITIZE_STRING);
    $pdo->refuserFraisHF($idFrais);
    header('Location:index.php?uc=validation&action=valider&idVisiteur=' .
    $idVisiteur . '&leMois=' . $leMois);
    include 'vues/v_validation.php';
    break;
```

Et donc la fonction **refuserFraisHF** en question :

```
public function refuserFraisHF($idFrais)
{
    $requetePrepares = PdoGsb::$monPdo->prepare(
        "SELECT libelle "
        . "FROM lignefraishorsforfait "
        . "WHERE id= :idFrais"
    );
    $requetePrepares->bindParam(':idFrais', $idFrais, PDO::PARAM_STR);
    $requetePrepares->execute();
    $libelleOriginal = $requetePrepares->fetch();







    $libelleModifie = 'REFUSE ' . $libelleOriginal['libelle'];
    $requeteModifie = PdoGsb::$monPdo->prepare(
        "UPDATE lignefraishorsforfait "
        . "set libelle = :nouveauLibelle "
        . "WHERE id = :idFrais"
    );
    $requeteModifie->bindParam(':nouveauLibelle', $libelleModifie,
    PDO::PARAM_STR);
    $requeteModifie->bindParam(':idFrais', $idFrais, PDO::PARAM_STR);
    $requeteModifie->execute();
}
```


Puis vient le moment de la validation de la fiche de frais, une fois les modifications apportées il suffit de cliquer sur **Valider la fiche** pour déclencher le contrôleur qui se chargera de faire une petite addition des frais forfait et ceux hors forfait et mettre à jour la base de données qui passera le statut de cette fiche à VA.

```
case 'succesValidation':  
    $montantForfait = $pdo->calculFraisForfait($idVisiteur, $leMois);  
    $montantHorsForfait = $pdo->calculFraisHF($idVisiteur, $leMois);  
    $enCours = $montantForfait + $montantHorsForfait;  
    $pdo->valideFrais($idVisiteur, $leMois, $enCours);  
    $pdo->majEtatFicheFrais($idVisiteur, $leMois, 'VA');  
    include 'vues/v_termValidation.php';  
    header('Refresh:5 ; URL=index.php?uc=validation&action=selectionVisiteurMois');  
    break;
```


VI - Paiement des fiches de frais

Fiches de Frais Validées, en attentes de remboursement :

Nom	Prénom	Mois de la fiche	Montant	Date de validation	Voir le détail	Payer : <input type="checkbox"/>
Bioret	Luc	202101	2 286,54 €	01/02/2021	 Voir la fiche	<input type="checkbox"/>
Bunisset	Francis	202101	1 715,82 €	01/02/2021	 Voir la fiche	<input type="checkbox"/>
Daburon	François	202101	2 291,70 €	01/02/2021	 Voir la fiche	<input type="checkbox"/>
Debelle	Michel	202101	3 429,18 €	01/02/2021	 Voir la fiche	<input type="checkbox"/>
Debroise	Michel	202101	2 889,32 €	01/02/2021	 Voir la fiche	<input type="checkbox"/>
Tusseau	Louis	202102	1 849,04 €	21/03/2021	 Voir la fiche	<input type="checkbox"/>

Payer 

Donc les fiches ont été clôturées puis validées il reste donc maintenant à les payer qui est la dernière étape du processus.

Voici le contrôleur en charge du paiement, il reçoit en action 'payer' qui déclenchera ce switch case :

```
case 'payer':  
    // Récupération des cases cochées et MAJ des données payées  
    $ficheRembourser = $pdo->getFichesVisiteursAPayer();  
    $virementMontant = 0;  
    foreach ($_POST['aPayer'] as $virement) {  
        $posId = strpos($virement, '&');  
        $idVisiteur = substr($virement, 0, $posId);  
        $moisPaye = substr($virement, $posId + 1);  
        $montantAPayer = getLeMontantAPayer($ficheRembourser, $idVisiteur,  
            $moisPaye);  
        $virementMontant += $montantAPayer;  
        $pdo->majEtatFicheFrais($idVisiteur, $moisPaye, 'RB');  
    }  
    include 'vues/v_termPaiement.php';  
    break;
```

En déclenchant ce switch case, ceci permet de passer la fiche au statut de Remboursée (RB) dans la base de données. Le comptable verra un message apparaître pour lui dire que la banque a bien été notifié d'un virement.

Virements sélectionnés pour un montant total de 2 286,54 € ont bien été notifiés à la banque !

VII – Mot de passe quel algorithme ?

Stocker des données sensibles en claires dans une base de données est quelques choses à proscrire, il faut impérativement crypter les données, car si une personne venait à s'introduire dans notre base alors il serait facile pour lui de naviguer d'un compte à l'autre ayant accès aux identifiants des utilisateurs.

Il existe plusieurs façons de crypter un mot de passe ou du moins des données sensibles sur une base de données. Je me suis rendu sur le site de la documentation de PHP et plus précisément sur ce lien :

<https://www.php.net/manual/fr/faq.passwords.php>

En lisant donc cette documentation on se rend vite compte que certains algorithmes destinés à être rapides sont maintenant dépassés, je veux parler par exemple de MD5, SHA1, SHA256.

Je me suis tourné dans un premier temps vers l'algorithme par défaut **PASSWORD_DEFAULT** puis en lisant un peu plus, il s'avère que celui-ci est amené à évoluer et se mettre à jour seul avec les mises à jour de PHP et donc risque de problème dans le temps si je ne rempli pas la taille de ma table correctement.

En continuant de lire la documentation, il nous est recommandé d'utiliser **BCRYPT()** avec l'algorithme Blowfish.

Quote de la documentation php :

« Une autre option est la fonction [crypt\(\)](#), qui supporte différents algorithmes de hachage en PHP 5.3 et suivants. Lors de l'utilisation de cette fonction, vous avez la garantie que l'algorithme sélectionné est disponible, sachant que PHP contient une implémentation native de chaque algorithme supporté, dans le cas où un ou plusieurs algorithmes n'est pas supporté par votre système.

L'algorithme suggéré à utiliser pour le hachage de mots de passe est Blowfish, qui est aussi l'algorithme par défaut de l'API de hachage de mots de passe, sachant qu'il est significativement plus gourmand en calcul que MD5 ou SHA1, mais plus évolutif. »

Il ne nous était pas demandé de créer un formulaire pour les utilisateurs ou comptable de s'enregistrer.

J'ai donc rentré manuellement dans la base de données mes comptables avec un mot de passe en clair, et a la première connexion celui-ci sera crypte car mon script vérifiera que le mdp commence par **\$2y2\$**, si ce n'est pas le cas alors le celui-ci sera crypté. Je ne me suis pas occupé des identifiants visiteurs, eux ne sont pas cryptés.

Quand je rentre mes identifiants comptable, le contrôleur **c_connexion.php** est appelé et rentre dans le switch case et le cas '**valideConnexion**' se lance.

```
case 'valideConnexion':
    $mDPasseDP = $pdo->getMDPComptables();

    foreach ($mDPasseDP as $comptable) {
        $mDPasse = $comptable['mdp'];
        $id = $comptable['id'];
        $hashage = substr($mDPasse, 0, 4);
        if ($hashage !== '$2y$') {
            $pdo->setMDPHashComptables($id, $mDPasse);
        }
    }

    // Récupération des identifiants saisis par l'utilisateur
    $login = checkInput($_POST['login']);
    $mdp = checkInput($_POST['mdp']);

    // Récupération des informations du comptable connecté
    $comptable = $pdo->getInfosComptable($login);

    // Si aucun comptable avec ce login alors nous affichons un message
    erreur
    if (!is_array($comptable)) {
        Erreur('Login ou mot de passe incorrect');
        include 'vues/v_erreurs.php';
        include 'vues/v_connexion.php';
    } else {
        // Sinon, vérification du mdp hashé
        $mdpValid = password_verify($mdp, $comptable['mdp']);
        // Si mdp est valide, création du cookie de session
        if ($mdpValid) {
            $id = $comptable['id'];
            $nom = $comptable['nom'];
            $prenom = $comptable['prenom'];
            connecter($id, $nom, $prenom);
            header('Location: index.php');
        } else {
            // Si les identifiants sont incorrects alors nous redirigeons
            la personne
            // et affichons un message d'erreur
            Erreur('Login ou mot de passe incorrect');
            include 'vues/v_erreurs.php';
            include 'vues/v_connexion.php';
        }
    }
    break;
```

On remarque que plusieurs fonctions sont appelées, mais surtout 2 nous intéressent. Je veux parler de :

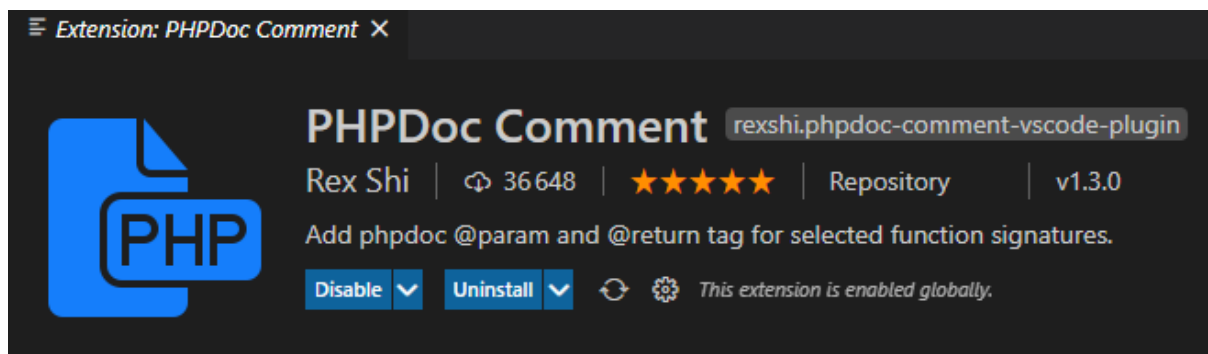
- `getMDPComptables();`
- `setMDPHashComptables()`

La première étant pour récupérer les mots de passes des comptables, et ensuite les passer dans la deuxième fonction qui va hacher le mot de passe si celui-ci ne l'est pas déjà.

VIII - Documentation

Tout au long du projet j'ai implémenté des commentaires a mes fonctions pour ensuite pouvoir générer une documentation claire pour les éventuels développeurs qui travaillerons sur ce projet.

J'ai ajouté ces commentaires avec l'aide d'une petite extension dans Visual Studio Code qui se nomme **phpDoc Comment** et j'ai également parcouru le site [Doxygen: Main Page](#) qui nous aide à comprendre comment faire de bon commentaires, pour une meilleure lecture de la documentation.



En seulement quelques clics, j'ai pu ensuite générer ma documentation au format html.

Cette documentation est visible à l'adresse suivante : <http://www.gsbdocumentation.g-alexis.com>

IX – Hébergement

J'ai transféré mes fichiers avec FileZilla mais j'aurai pu faire un git clone directement ayant git d'installé sur le serveur (Je suis sous Debian 10).

J'ai paramétré ce projet sous un sous-domaine, avec la configuration suivante :

```
VirtualHost *:80>
    ServerAdmin admin@your_domain.com
    DocumentRoot /var/www/gsb_comptable
    ServerName gsbcomptable.g-alexis.com

    <Directory /var/www/gsb_comptable>
        Options FollowSymlinks
        AllowOverride All
        Require all granted
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/gsb_comptable_error.log
    CustomLog ${APACHE_LOG_DIR}/gsb_comptable_access.log combined

</VirtualHost>
```

Puis activé le site avec la commande :

- 2ensite gsb_compta.conf
- Systemctl reload apache2

Puis ensuite, il a fallu créer une base de données et y importer la base que j'utilisais en local.

```
MariaDB [(none)]> create database gsb_restore;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> use gsb_restore;
Database changed
MariaDB [gsb_restore]> source /var/www/gsb_comptable/gsb_restore.sql;
Query OK, 0 rows affected (0.000 sec)
```

On vérifie que les tables ont bien été ajoutées avec la commande :

- show tables ;

```
MariaDB [gsb_restore]> show tables;
+-----+
| Tables_in_gsb_restore |
+-----+
| comptable              |
| etat                   |
| fichefrais             |
| fraisforfait           |
| lignefraisforfait      |
| lignefraishorsforfait  |
| visiteur               |
+-----+
7 rows in set (0.000 sec)

MariaDB [gsb_restore]>
```

Puis nous accordons les privilèges à cet utilisateur pour pouvoir se connecter à la base de données.

```
MariaDB [gsb_restore]> GRANT ALL PRIVILEGES ON *.* TO "userRoot"@"%" IDENTIFIED BY "gsb";  
Query OK, 0 rows affected (0.000 sec)  
  
MariaDB [gsb_restore]> FLUSH PRIVILEGES;  
Query OK, 0 rows affected (0.000 sec)
```