

**Compte rendu de la mission 1 :**  
**Développement de la partie**  
**comptable de l'application GSB de**  
**suivi des frais**

**Florian Martin – BTS SIO option SLAM**

**Année 2019-2020**

## Sommaire

I) Contexte.....	1
II) Architecture de l'application.....	2
III) Connexion des utilisateurs.....	3
IV) Valider les fiches de frais .....	9
V) Suivi du paiement des fiches de frais .....	22
VI) Renseigner une fiche de frais pour un visiteur .....	26
VII) Gestion de la base de données.....	28
VIII) Gestion de version du projet .....	29
IX)Déploiement de l'application sur le serveur .....	30
X) Documentation technique .....	31
XI) Tests unitaires .....	32

## I) Contexte

Galaxy Swiss Bourdin (GSB) est un laboratoire pharmaceutique. Le suivi des frais des visiteurs-salariés du laboratoire est actuellement géré de plusieurs façons. On souhaite uniformiser cette gestion.

Ainsi, une application doit être développée, destinée aux visiteurs et aux comptables de l'entreprise, permettant d'enregistrer tous les frais engagés (événementiel, déplacement, restauration...). L'application doit notamment permettre un suivi daté des opérations menées par le service comptable (validation des demandes de remboursement, mise en paiement, remboursement effectué). Pour ce qui est de l'interface destinée aux visiteurs, elle doit permettre à ces derniers de pouvoir saisir tous les frais qu'ils engendrent.

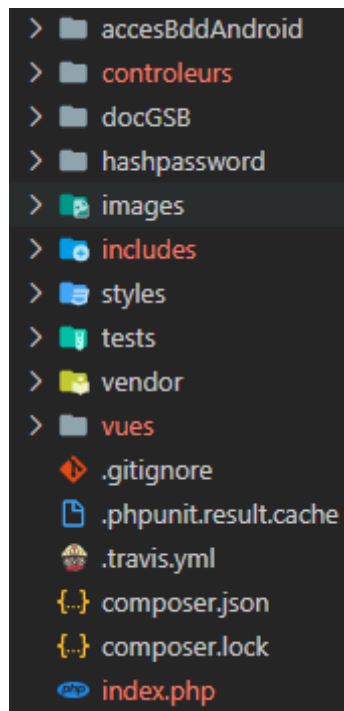
Une application existante permettait déjà aux visiteurs de renseigner les différents frais engendrés tels que les nuitées, les frais kilométriques et tout autres frais se rapportant à des frais engendrés par leur métier.

En revanche, la partie comptable permettant de suivre les frais, les validations des frais, le suivi des remboursements et autre n'était pas encore développé. Je me suis donc chargé de développer cette partie pour les comptables.

J'ai ainsi dû réaliser plusieurs tâches : coder la page de validation d'une fiche de frais accessible uniquement aux comptables, coder la page de suivi de paiement, gérer le refus ou le report de certains frais qui est uniquement disponible pour les comptables, sécuriser les mots de passe stockés dans la base de données, production d'une documentation technique.

Cette application a été développée en PHP, HTML, CSS et avec le langage SQL. Le développement a été effectué grâce à l'IDE Visual Studio Code. Concernant la gestion de version du projet, je me suis servi du logiciel de gestion de version Git et du service en ligne GitHub.

## II) Architecture de l'application



Dans le dossier « accesBddAndroid », on retrouve le fichier php qui permet à l'application Android d'interagir avec la base de données.

Dans le dossier « controleurs », on retrouve tous les contrôleurs de l'application.

Dans le dossier « docGSB », comme son nom l'indique, on retrouve la documentation technique de l'application.

Dans le dossier hashpassword, on retrouve le fichier permettant d'hasher les mots de passe des utilisateurs.

Dans le dossier « images », on retrouve les images de l'application.

Dans le dossier « includes » on retrouve les classes permettant d'interagir avec la base de données et les fonctions globales à l'application.

Dans le dossier « styles » on retrouve les classes css, notamment le fichier de styles css du framework bootstrap.

Dans le dossier « test » on retrouve les fichiers permettant d'effectuer les tests unitaires.

Dans le dossier « vues » on retrouve les fichiers permettant d'afficher les différentes vues de l'application.

Enfin, on retrouve le fichier « index.php » qui est le point d'entrée de l'application et le fichier qui redirige les différents traitements demandés par l'utilisateur.


### III) Connexion des utilisateurs


```
17 require_once 'includes/fct.inc.php';
18 require_once 'includes/class.pdgsb.inc.php';
19 use Pdgsb\Pdgsb;
20 session_start();
21 $pdo = Pdgsb::getPdgsb('Pdgsb');
22 $estConnecte = estConnecte();
23 $typeUtilisateur = typeUtilisateur();
24 $uc = filter_input(INPUT_GET, 'uc', FILTER_SANITIZE_STRING);
25 if ($uc && !$estConnecte) {
26     $uc = 'connexion';
27 } elseif (empty($uc)) {
28     $uc = 'accueil';
29 }
30 switch ($uc) {
31     case 'connexion':
32         include 'controleurs/c_connexion.php';
33         break;
34     case 'accueil':
35         include 'vues/v_entete.php';
36         include 'controleurs/c_accueil.php';
37         break;
38     case 'gererFrais':
39         include 'vues/v_entete.php';
40         include 'controleurs/c_gererFrais.php';
41         break;
42     case 'etatFrais':
43         include 'vues/v_entete.php';
44         include 'controleurs/c_etatFrais.php';
45         break;
46     case 'validerFrais':
47         include 'vues/v_entete.php';
48         include 'controleurs/c_validerFrais.php';
49         break;
50     case 'suivreFrais':
51         include 'vues/v_entete.php';
52         include 'controleurs/c_suivreFrais.php';
53         break;
54     case 'deconnexion':
55         include 'controleurs/c_deconnexion.php';
56         break;
57 }
58 require 'vues/v_pied.php';
59
```

Les 2 captures d'écrans ci-dessus représente le fichier index.php qui est la porte d'entrée pour chaque page demandée par l'utilisateur de l'application. En effet, elle permet, en fonction de l'URL demandée, d'appeler le contrôleur adéquat qui s'occupera des traitements à effectuer et des vues à afficher. Dans le code source qui m'avait été fourni, uniquement les cases « connexion », « accueil », « gererFrais », « etatFrais » et « deconnexion » étaient présents. J'ai ainsi implémenté les cases « validerFrais » et « suivreFrais » nécessaires pour le développement de la partie comptable.



Identification utilisateur

 Login

 Mot de passe

Se connecter

```
26 case 'valideConnexion':  
27     $login = filter_input(INPUT_POST, 'login', FILTER_SANITIZE_STRING);  
28     $mdp = filter_input(INPUT_POST, 'mdp', FILTER_SANITIZE_STRING);  
29     $visiteur = $pdo->getInfosVisiteur($login, $mdp);  
30     $comptable = $pdo->getInfosComptable($login, $mdp);
```

Lors de la connexion, un contrôle est fait pour savoir si le couple login/mot de passe est correct. Le login et le mot de passe sont envoyés par le formulaire avec la méthode POST. Ainsi, on récupère ici le contenu des variables POST contenant le login et le mot de passe. J'ai implémenté la méthode `getInfosComptable` qui n'était pas présente dans le code source.

```

128 public function getInfosComptable($login, $mdp)
129 {
130     // Hâchge du mdp entré grâce à l'algorithme sha256
131     $mdpCrypte = hash("sha256", $mdp);
132     $requetePrepare = PdoGSB::$_monPdo->prepare(
133         'SELECT mdp '
134         . 'FROM comptable '
135         . 'WHERE login = :unLogin'
136     );
137     $requetePrepare->bindParam(':unLogin', $login, PDO::PARAM_STR);
138     $requetePrepare->execute();
139     $mdpStocke = $requetePrepare->fetch();
140
141     $requetePrepare = PdoGSB::$_monPdo->prepare(
142         'SELECT comptable.id AS id, comptable.nom AS nom, '
143         . 'comptable.prenom AS prenom '
144         . 'FROM comptable '
145         . 'WHERE comptable.login = :unLogin AND comptable.mdp = :unMdp'
146     );
147     $requetePrepare->bindParam(':unLogin', $login, PDO::PARAM_STR);
148     $requetePrepare->bindParam(':unMdp', $mdpCrypte, PDO::PARAM_STR);
149     $requetePrepare->execute();
150     $infosComptable = $requetePrepare->fetch();
151
152     /* Si la clé de hashage obtenue est identique à celle stockée en BDD, alors
153     * on retourne les infos du comptable
154     */
155     if ($mdpCrypte == $mdpStocke['mdp']) {
156         return $infosComptable;
157     } else {
158         return null;
159     }
160 }

```

Le contrôle du couple identifiant/mot de passe est réalisé dans la fonction « getInfosVisiteur » ou « getInfosComptable » en allant interroger la base de données avec un cryptage préalable du mot de passe. En effet, les mots de passe dans la base de données sont cryptés pour les sécuriser. Ainsi, si un pirate venait à se connecter à la base de données, ils ne pourraient pas se connecter à l'application étant donné que les mots de passes sont cryptés. Cette notion de cryptage n'était pas présente dans le code source, je l'ai donc implémenté. Le hashage est fait grâce à l'algorithme sha256 qui est un algorithme de sécurité fiable et reconnu.

```

31  /** Si l'utilisateur qui se connecte est un visiteur, on valorise les
32  * variable de session pour un visiteur, sinon on valorise pour un
33  * comptable. Si l'authentification échoue, on affiche un message d'erreur.
34  */
35  if (is_array($visiteur)) {
36      $id = $visiteur['id'];
37      $nom = $visiteur['nom'];
38      $prenom = $visiteur['prenom'];
39      connecter($id, $nom, $prenom, 'visiteur');
40  } elseif (is_array($comptable)) {
41      $id = $comptable['id'];
42      $nom = $comptable['nom'];
43      $prenom = $comptable['prenom'];
44      connecter($id, $nom, $prenom, 'comptable');

```

```

41  /**
42  * Enregistre dans une variable session les infos de l'utilisateur
43  *
44  * @param String $idUtilisateur ID du visiteur
45  * @param String $nom Nom du visiteur
46  * @param String $prenom Prénom du visiteur
47  * @param String $typeUtilisateur type d'utilisateur (visiteur ou comptable)
48  *
49  * @return null
50  */
51  function connecter($idUtilisateur, $nom, $prenom, $typeUtilisateur)
52  {
53      $_SESSION['idUtilisateur'] = $idUtilisateur;
54      $_SESSION['nom'] = $nom;
55      $_SESSION['prenom'] = $prenom;
56      $_SESSION['typeUtilisateur'] = $typeUtilisateur;
57  }
58

```

Si la requête retourne un résultat, alors on vérifie ensuite si l'utilisateur qui se connecte est un visiteur ou un comptable. En effet, en fonction du type d'utilisateur qui se connecte, les interfaces et fonctionnalités ne sont pas les mêmes. Si un des 2 types d'utilisateurs est valorisés, alors on appelle la fonction « connecter » qui s'occupe de valoriser des variables de sessions contenant l'id de l'utilisateur, son nom, son prénom et le type de l'utilisateur qui s'est connecté qui va servir pour des traitements futurs.



```

45     } else {
46         ajouterErreur('Login ou mot de passe incorrect');
47         include 'vues/v_entete.php';
48         include 'vues/v_erreurs.php';
49         include 'vues/v_connexion.php';
50     }
51     if (is_array($visiteur) | is_array($comptable)) {
52         header('Location: index.php');
53     }
54     break;

```


Si la connexion échoue, une vue d'erreur est affichée indiquant à l'utilisateur que le mot de passe ou l'identifiant est incorrect.



Login ou mot de passe incorrect

Identification utilisateur

 dandre

 .....

Se connecter

Si la connexion est réussie, on redirige l'utilisateur vers la page d'accueil.



[Accueil](#)
[✓ Valider les fiches de frais](#)
[€ Suivre le paiement des fiches de frais](#)
[Déconnexion](#)

Gestion des frais - Comptable : Françoise Goudet

Navigation



Valider les fiches de frais



Suivre le paiement des fiches de frais



## Gestion des frais - Visiteur : David Andre

### Navigation

[Renseigner la fiche de frais](#)[Afficher mes fiches de frais](#)

```
61 <span <?php if ($typeUtilisateur == 'visiteur') {  
62     ?>  
63     class="glyphicon glyphicon-list-alt" <?php } else {  
64         ?>  
65     class="glyphicon glyphicon-ok" <?php } ?></span>  
66     <br>  
67     <?php if ($typeUtilisateur == 'visiteur') {  
68         ?> Renseigner la fiche de frais  
69     <?php } else {  
70         ?> Valider les fiches de frais <?php  
71     } ?></a>  
72 <?php if ($typeUtilisateur == 'visiteur') { ?>  
73 <a href="index.php?uc-etatFrais&action=selectionnerMois"  
74     class="btn btn-primary btn-lg" role="button">  
75 <?php } else { ?>  
76 <a href="index.php?uc=suivreFrais"  
77     class="btn btn-primary-comptable btn-lg" role="button">  
78 <?php } ?>  
79 <span <?php if ($typeUtilisateur == 'visiteur') {  
80     ?>  
81     class="glyphicon glyphicon-list-alt" <?php } else {  
82         ?>  
83     class="glyphicon glyphicon-euro" <?php } ?></span>  
84     <br>  
85     <?php if ($typeUtilisateur == 'visiteur') {  
86         ?> Afficher mes fiches de frais  
87     <?php } else {  
88         ?> Suivre le paiement des fiches de frais <?php  
89     }  
90     ?></a>  
91
```

Voici la page d'accueil. On peut ainsi voir que la page affichée est fonction du type d'utilisateur, à savoir un visiteur commercial ou un comptable. La vue pour les visiteurs était déjà existante dans le code source. Je me suis donc occupé de développer la vue pour les comptables. Comme dans la capture d'écran ci-dessus, on peut voir que je me suis occupé d'écrire le code php qui, en fonction du type d'utilisateur connecté (récupéré grâce à la variable de session au préalable initialisée lors de la connexion) affiche les bons termes et les bons traitements au clic sur les boutons en fonction du type d'utilisateur.

#### IV) Valider les fiches de frais

```
<a href="index.php?uc=validerFrais"
  <?php if ($typeUtilisateur == 'comptable') {
    ?>class="a-comptable" <?php |
  }
  ?>>
  <span class="glyphicon glyphicon-ok"></span>
  Valider les fiches de frais
</a>
```

Au clic sur le bouton ou sur l'onglet « valider les fiches de frais », on peut voir dans la capture du code php précédente que l'utilisateur est redirigé sur le lien « index.php?uc=validerFrais ». Ainsi, lors du clic, on appelle tout d'abord le fichier index.php qui s'occupe de récupérer les paramètres dans l'URL, notamment « uc= validerFrais » pour valoriser la variable uc :

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

Une fois la valeur récupérée, la variable \$uc vaut donc « validerFrais ». Il suffit ensuite d'appeler le bon contrôleur toujours dans le fichier index.php en passant dans le case « validerFrais » :

```

30  switch ($uc) {
31  case 'connexion':
32      include 'controleurs/c_connexion.php';
33      break;
34  case 'accueil':
35      include 'vues/v_entete.php';
36      include 'controleurs/c_accueil.php';
37      break;
38  case 'gererFrais':
39      include 'vues/v_entete.php';
40      include 'controleurs/c_gererFrais.php';
41      break;
42  case 'etatFrais':
43      include 'vues/v_entete.php';
44      include 'controleurs/c_etatFrais.php';
45      break;
46  case 'validerFrais':
47      include 'vues/v_entete.php';
48      include 'controleurs/c_validerFrais.php';
49      break;

```

Le contrôleur s'occupe alors d'afficher la sélection d'un visiteur et d'une fiche de frais :

gsb

[Accueil](#)
[✓ Valider les fiches de frais](#)
[🔄 Suivre le paiement des fiches de frais](#)
[Déconnexion](#)

Choisir le visiteur :  Mois :

```

105  require 'vues/v_listeVisiteur.php';

```

Une fois le visiteur et le mois sélectionné, au clic sur le bouton valider, on envoi en post le visiteur et le mois sélectionné. On récupère les valeurs dans le contrôleur :

```

18  <form
19      <?php if ($uc == 'validerFrais') {
20          ?> action="index.php?uc=validerFrais&action=afficherFrais"

```

```

20  /*
21  * On récupère l'id du visiteur sélectionné et le mois de la fiche sélectionnée.
22  * On stocke l'id et le mois dans une variable de session afin que ces 2 variables
23  * soient accessibles dans toutes les vues et les autres contrôleurs.
24  */
25  if (filter_input(
26      INPUT_POST,
27      'lstVisiteur',
28      FILTER_SANITIZE_STRING
29  )
30  ) {
31      $idVisiteurSelectionne = filter_input(
32          INPUT_POST,
33          'lstVisiteur',
34          FILTER_SANITIZE_STRING
35      );
36  }
37
38  if (filter_input(INPUT_POST, 'lstMois', FILTER_SANITIZE_STRING)) {
39      $moisFicheSelectionne = filter_input(
40          INPUT_POST,
41          'lstMois',
42          FILTER_SANITIZE_STRING
43      );
44  }

```

Dans le contrôleur validerFrais, on récupère également l'action :

```

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

```

Etant donné que l'action était « afficherFrais », on se retrouve dans le case « afficherFrais » qui s'occupe de récupérer tous les mois du visiteur répertoriés dans la base de données grâce à la méthode getLesMoisDisponibles. Si le mois sélectionné par ce comptable correspond à un des mois référencés pour le visiteur sélectionné, alors on passe le booléen ficheExistante à vrai. Si aucun mois ne correspond, alors on affiche la vue erreur qui affiche le message passé en paramètre de la fonction ajouterErreur.

```

106 switch($action) {
107 case 'afficherFrais':
108     $lesMoisDuVisiteur = $pdo->getLesMoisDisponibles($idVisiteurSelectionne);
109     foreach ($lesMoisDuVisiteur as $unMois) {
110         if ($moisFicheSelectionne == $unMois['mois']) {
111             $ficheExistante = true;
112         }
113     }
114     if (!$ficheExistante) {
115         ajouterErreur(
116             'Pas de fiche de frais pour ce visiteur ce mois,
117             veuillez en choisir une autre.'
118         );
119         include 'vues/v_erreurs.php';
120     }
121     break;

```

Enfin, on fini par afficher les frais pour le visiteur et le mois sélectionné. Les frais forfait et hors forfait sont récupérés dans la base de données par l'intermédiaire de requête SQL exécutées dans les méthodes `getLesFraisHorsForfait` et `getLesFraisForfait` :

```

241 /**
242  * Retourne sous forme d'un tableau associatif toutes les lignes de frais
243  * hors forfait concernées par les deux arguments.
244  * La boucle foreach ne peut être utilisée ici car on procède
245  * à une modification de la structure itérée - transformation du champ date-
246  *
247  * @param String $idVisiteur ID du visiteur
248  * @param String $mois      Mois sous la forme aaaamm
249  *
250  * @return tous les champs des lignes de frais hors forfait sous la forme
251  * d'un tableau associatif
252  */
253 public function getLesFraisHorsForfait($idVisiteur, $mois)
254 {
255     $requetePrepare = $pdoGsb::$monPdo->prepare([
256         'SELECT * FROM lignefraishorsforfait '
257         . 'WHERE lignefraishorsforfait.idvisiteur = :unIdVisiteur '
258         . 'AND lignefraishorsforfait.mois = :unMois'
259     ]);
260     $requetePrepare->bindParam(':unIdVisiteur', $idVisiteur, PDO::PARAM_STR);
261     $requetePrepare->bindParam(':unMois', $mois, PDO::PARAM_STR);
262     $requetePrepare->execute();
263     $lesLignes = $requetePrepare->fetchAll(PDO::FETCH_ASSOC);
264     for ($i = 0; $i < count($lesLignes); $i++) {
265         $date = $lesLignes[$i]['date'];
266         $lesLignes[$i]['date'] = dateAnglaisVersFrancais($date);
267     }
268     return $lesLignes;
269 }

```

```

293      /**
294       * Retourne sous forme d'un tableau associatif toutes les lignes de frais
295       * au forfait concernées par les deux arguments
296       *
297       * @param String $idVisiteur ID du visiteur
298       * @param String $mois      Mois sous la forme aaaamm
299       *
300       * @return l'id, le libelle et la quantité sous la forme d'un tableau
301       * associatif
302       */
303     public function getLesFraisForfait($idVisiteur, $mois)
304     {
305         $requetePrepare = PdoGSB::$monPdo->prepare(
306             'SELECT fraisforfait.id as idfrais, '
307             . 'fraisforfait.libelle as libelle, '
308             . 'lignefraisforfait.quantite as quantite '
309             . 'FROM lignefraisforfait '
310             . 'INNER JOIN fraisforfait '
311             . 'ON fraisforfait.id = lignefraisforfait.idfraisforfait '
312             . 'WHERE lignefraisforfait.idvisiteur = :unIdVisiteur '
313             . 'AND lignefraisforfait.mois = :unMois '
314             . 'ORDER BY lignefraisforfait.idfraisforfait'
315         );
316         $requetePrepare->bindParam(':unIdVisiteur', $idVisiteur, PDO::PARAM_STR);
317         $requetePrepare->bindParam(':unMois', $mois, PDO::PARAM_STR);
318         $requetePrepare->execute();
319         return $requetePrepare->fetchAll(PDO::FETCH_ASSOC);
320     }
321

```


```

240     if ($ficheExistante && !$estFicheValidee && $libEtat == 'CL') {
241         $nomEtPrenomVisiteur = $pdo->getNomEtPrenomVisiteur(
242             $idVisiteurSelectionne
243         );
244         $lesFraisHorsForfait = $pdo->getLesFraisHorsForfait(
245             $idVisiteurSelectionne,
246             $moisFicheSelectionne
247         );
248         $lesFraisForfait = $pdo->getLesFraisForfait(
249             $idVisiteurSelectionne,
250             $moisFicheSelectionne
251         );
252         include 'vues/v_listeFraisForfait.php';
253         include 'vues/v_listeFraisHorsForfait.php';

```

Comme on peut le voir sur la capture d'écran ci-dessus, on affiche la vue listeFraisForfait et listeFraisHorsForfait. Concernant la condition if, on affiche uniquement ces vues si la fiche sélectionnée pour le visiteur est existante, si la fiche n'a pas été validée et si le libellé de l'état de la fiche est à 'CL' (CL pour clôturée). En effet, les comptables peuvent valider une fiche uniquement si la fiche n'a pas déjà été validée et si le libellé de l'état est à 'CL'. Si le libellé est à 'CR' (CR pour créée) c'est que la fiche est encore en cours de saisie pour le mois courant et ne peut donc pas être validée, elle ne sera validable qu'à la fin du mois lorsque tous les frais auront été saisis. De même, si le libellé est à 'RB' (RB pour remboursée) c'est que la fiche a déjà été remboursée, elle a donc déjà été validée avant le remboursement.

Voici la vue permettant de valider une fiche pour un comptable :



[Accueil](#) [✓ Valider les fiches de frais](#) [Suivre le paiement des fiches de frais](#) [Déconnexion](#)

Choisir le visiteur : Villechalane Louis Mois : 12/2019 Valider

Valider la fiche de frais de Villechalane Louis

Eléments forfaitisés

Forfait Etape

3

Frais Kilométrique

1

Nuitée Hôtel

3

Repas Restaurant

3

Corriger Réinitialiser

Descriptif des éléments hors forfait

Date	Libellé	Montant	
04/12/2019	REFUSE Voyage SNCF	145.00	<button>Corriger</button> <button>Reporter</button> <button>Refuser</button>
18/12/2019	Frais vestimentaire/représen	260.00	<button>Corriger</button> <button>Reporter</button> <button>Refuser</button>
09/12/2019	Location équipement vidéo/	473.00	<button>Corriger</button> <button>Reporter</button> <button>Refuser</button>
06/12/2019	Location équipement vidéo/	303.00	<button>Corriger</button> <button>Reporter</button> <button>Refuser</button>
20/12/2019	Voyage SNCF	124.00	<button>Corriger</button> <button>Reporter</button> <button>Refuser</button>

Nombre de justificatifs : 0 Valider les justificatifs

Valider la fiche

Powered I



```

59 <tr>
60 <td id="td-utilisateur">
61 <input type="text"
62     id="form-control-comptable"
63     name="dateFrais-corrige"
64     class="form-control" id="text"
65     value="<?php echo $date?>">
66 </td>
67 <td id="td-utilisateur">
68 <input type="text"
69     id="form-control-comptable"
70     name="libelle-corrige"
71     class="form-control" id="text"
72     value="<?php echo $libelle?>">
73 </td>
74 <td id="td-utilisateur">
75 <input type="text"
76     id="form-control-comptable"
77     name="montant-corrige"
78     class="form-control" id="text"
79     value="<?php echo $montant?>">
80 </td>
81 <td id="td-utilisateur">
82 <button class="btn btn-success" name="corriger"
83     value="<?php echo $idFraisHorsForfait ?>"
84     type="submit">Corriger
85 </button>
86 <button class="btn btn-danger" name="reporter"
87     value="<?php echo $idFraisHorsForfait ?>"
88     type="submit">Reporter
89 </button>
90 <button class="btn btn-danger" name="refuser"
91     value="<?php echo $idFraisHorsForfait ?>"
92     type="submit">Refuser

```

Ainsi, le comptable connecté peut donc corriger les frais forfaits et hors forfaits en cliquant sur le bouton « corriger » après avoir modifier les inputs qu'ils souhaitent corriger. Également, pour les frais hors forfait, il peut reporter au mois suivant un frais en cliquant sur le bouton « reporter » de la ligne qu'il souhaite reporter par exemple si le justificatif n'a pas été fourni à temps. Enfin, il peut refuser un frais en cliquant sur le bouton « refuser » de la ligne correspondante. En cliquant sur le bouton « refuser », le mot « REFUSE » s'ajoute devant le libellé. C'est par exemple le cas sur la capture d'écran ci-dessus pour la première ligne du tableau, on peut voir « REFUSE Voyage SNCF ». Lorsque le comptable va valider cette fiche, ce frais ne sera pas pris en compte dans la somme à rembourser. Également, le comptable peut renseigner le nombre de justificatifs dans le champ correspondant et ainsi valider le nombre de justificatifs fourni pour cette fiche.

Voici quelques captures d'écran qui illustrent certains traitements effectués derrière les clics sur les boutons.

```
136      /* Si le frais est à refuser, on ajoute le texte 'REFUSE' devant
137      * le libellé du frais hors forfait afin de savoir qu'il a été refusé
138      * et qu'il ne sera pas pris en compte dans les remboursements.
139      * */
140      if ($traitementAEffectuer == 'refuser') {
141          if (substr($libelleFrais, 0, 6) != 'REFUSE') {
142              $libelleFrais = 'REFUSE ' . $libelleFrais;
143          }
144      }
145  }
```

```
150      if (($traitementAEffectuer == 'corriger' |
151          || $traitementAEffectuer == 'refuser'
152      )) {
153          $pdo->majFraisHorsForfait(
154              $idFraisHorsForfait,
155              $idVisiteurSelectionne,
156              $moisFicheSelectionne,
157              $libelleFrais,
158              $dateFrais,
159              $montantFrais
160          );
161          /* Si le frais est à reporter, on doit vérifier que la fiche
162          * dans laquelle on reporte le frais est bien créée. Si ce n'est
163          * pas le cas, on la crée puis on reporte le frais. On supprime également
164          * le frais de la fiche actuelle.
165          */
166      } elseif ($traitementAEffectuer == 'reporter') {
167          $mois = getMois(date('d/m/Y'));
168          if ($pdo->estPremierFraisMois($idVisiteurSelectionne, $mois)) {
169              $pdo->creeNouvellesLignesFrais(
170                  $idVisiteurSelectionne,
171                  $mois
172              );
173          }
174      }
```

Dans la capture ci-dessus, nous sommes dans le contrôleur `validerFrais`. Ici, la variable « `traitementAEffectuer` » est initialisée dans ce contrôleur en récupérant le contenu de la variable POST reçue. Si le traitement à effectuer est une correction ou un refus, alors on met à jour le frais hors forfaits dans la base de données avec les valeurs corrigées par le comptable. Également, on peut voir au préalable que si le traitement est de refuser un frais, on ajoute au libellé du frais hors forfait le mot `REFUSE`.

En revanche, comme l'explique le commentaire dans la capture d'écran, si l'action demandée est un report, alors on reporte le frais hors forfait sur la fiche du mois suivant en s'assurant bien évidemment que la fiche du mois suivant existe, auquel cas on l'a créée dans le cas contraire.

```

709  /**
710   * Met à jour un frais hors forfait pour un visiteur et un mois donné
711   * à partir des informations fournies en paramètre
712   *
713   * @param String $idFraisHorsForfait ID du frais hors forfait
714   * @param String $idVisiteur        ID du visiteur
715   * @param String $mois              Mois sous la forme aaaamm
716   * @param String $libelle           Libellé du frais
717   * @param String $date              Date du frais au format français jj//mm/aaaa
718   * @param Float  $montant           Montant du frais
719   *
720   * @return null
721   */
722  public function majFraisHorsForfait(
723      $idFraisHorsForfait,
724      $idVisiteur,
725      $mois,
726      $libelle,
727      $date,
728      $montant
729  ) {
730      $dateFr = dateFrancaisVersAnglais($date);
731      $requetePrepare = PdoGSB::$_monPdo->prepare(
732          'UPDATE lignefraishorsforfait '
733          . 'SET libelle = :unLibelle, date = :uneDate, '
734          . 'montant = :unMontant '
735          . 'WHERE id = :unId AND idVisiteur = :unIdVisiteur AND '
736          . 'mois = :unMois'
737      );
738      $requetePrepare->bindParam(':unIdVisiteur', $idVisiteur, PDO::PARAM_STR);
739      $requetePrepare->bindParam(':unMois', $mois, PDO::PARAM_STR);
740      $requetePrepare->bindParam(':unLibelle', $libelle, PDO::PARAM_STR);
741      $requetePrepare->bindParam(':uneDate', $dateFr, PDO::PARAM_STR);
742      $requetePrepare->bindParam(':unMontant', $montant, PDO::PARAM_STR);
743      $requetePrepare->bindParam(':unId', $idFraisHorsForfait, PDO::PARAM_INT);
744      $requetePrepare->execute();
745  }
746

```

Cette capture montre comment la mise à jour d'un frais hors forfait est effectuée. On vient mettre à jour le frais hors forfait correspondant à l'id du frais hors forfait passé en paramètre.

```

221  /**
222   * Vérifie la validité des trois arguments : la date, le libellé du frais
223   * et le montant
224   *
225   * Des message d'erreurs sont ajoutés au tableau des erreurs
226   *
227   * @param String $dateFrais Date des frais
228   * @param String $libelle   Libellé des frais
229   * @param Float  $montant   Montant des frais
230   *
231   * @return null
232   */
233  function valideInfosFrais($dateFrais, $libelle, $montant)
234  {
235      if ($dateFrais == '') {
236          ajouterErreur('Le champ date ne doit pas être vide');
237      } else {
238          if (!estDatevalide($dateFrais)) {
239              ajouterErreur('Date invalide');
240          } else {
241              if (estDateDepassee($dateFrais)) {
242                  ajouterErreur(
243                      "date d'enregistrement du frais dépassé, plus de 1 an"
244                  );
245              }
246          }
247      }
248      if ($libelle == '') {
249          ajouterErreur('Le champ description ne peut pas être vide');
250      }
251      if ($montant == '') {
252          ajouterErreur('Le champ montant ne peut pas être vide');
253      } elseif (!is_numeric($montant)) {
254          ajouterErreur('Le champ montant doit être numérique');
255      }
256  }
257

```

Dans le contrôleur validerFrais, avant la correction, le report ou le refus de frais, on appelle la méthode valideInfosFrais (dont le contenu est détaillé dans la capture ci-dessus) pour vérifier que les valeurs entrées par le comptable sont des valeurs valides (si le montant est un nombre par exemple ou si un champ n'est pas vide).

```

206     case 'validerFiche':
207         $pdo->validerLaFiche(
208             $idVisiteurSelectionne,
209             $idComptable,
210             $moisFicheSelectionne
211         );
212         $montantValide = $pdo->getMontantValideHorsFraisRefuses(
213             $idVisiteurSelectionne,
214             $moisFicheSelectionne
215         );
216         $pdo->majMontantValide(
217             $idVisiteurSelectionne,
218             $moisFicheSelectionne,
219             $montantValide
220         );
221         $estFicheValidee = true;
222         break;
223     }

```

Enfin, au clic sur le bouton « Valider la fiche », on se retrouve dans le case « validerFiche », toujours dans le contrôleur « validerFrais ». Ici, on appelle la méthode validerLaFiche dont voici le contenu et qui s'occupe de mettre la fiche à l'état validée, d'affecter le comptable qui a effectué la validation à la fiche et d'ajouter la date de modification de la fiche :

```

747  /**
748  * Permet de valider la fiche pour un visiteur donné et un mois donné
749  * en modifiant la date de modification de la fiche à celle du jour actuel et
750  * en affectant le comptable qui a effectué la validation à la fiche
751  *
752  * @param String $idVisiteur ID du visiteur
753  * @param String $idComptable ID du comptable
754  * @param String $mois Mois sous la forme aaaamm
755  *
756  * @return null
757  */
758  public function validerLaFiche($idVisiteur, $idComptable, $mois)
759  {
760      $requetePrepare = PdoGSB::$_monPdo->prepare(
761          'UPDATE fichefrais '
762          . 'SET idcomptable = :unIdComptable, '
763          . "idetat = 'VA', "
764          . 'datemodif = now() '
765          . 'WHERE idvisiteur = :unIdVisiteur AND '
766          . 'mois = :unMois'
767      );
768      $requetePrepare->bindParam(':unIdComptable', $idComptable, PDO::PARAM_STR);
769      $requetePrepare->bindParam(':unIdVisiteur', $idVisiteur, PDO::PARAM_STR);
770      $requetePrepare->bindParam(':unMois', $mois, PDO::PARAM_STR);
771      $requetePrepare->execute();
772  }

```

Également, dans ce case, on s'occupe de mettre à jour le montant validé pour la fiche, sans prendre en compte les libellés commençant par « REFUSE ». Cela s'effectue dans les deux méthodes suivantes :

```

783 public function getMontantValideHorsFraisRefuses($idVisiteur, $mois)
784 {
785     $requetePrepare = PdoGSB::$_monPdo->prepare(
786         "SELECT SUM(quantite * montant) AS 'montant valide' "
787         . 'FROM fraisforfait JOIN lignefraisforfait '
788         . 'ON idfraisforfait = id '
789         . 'WHERE idvisiteur = :unIdVisiteur AND '
790         . 'mois = :unMois'
791     );
792     $requetePrepare->bindParam(':unIdVisiteur', $idVisiteur, PDO::PARAM_STR);
793     $requetePrepare->bindParam(':unMois', $mois, PDO::PARAM_STR);
794     $requetePrepare->execute();
795
796     $montant = $requetePrepare->fetch();
797     $montantFraisForfait = $montant['montant valide'];
798
799     $requetePrepare = PdoGSB::$_monPdo->prepare(
800         'SELECT montant, libelle '
801         . 'FROM lignefraishorsforfait '
802         . 'WHERE idvisiteur = :unIdVisiteur AND '
803         . 'mois = :unMois'
804     );
805     $requetePrepare->bindParam(':unIdVisiteur', $idVisiteur, PDO::PARAM_STR);
806     $requetePrepare->bindParam(':unMois', $mois, PDO::PARAM_STR);
807     $requetePrepare->execute();
808
809     $fraisHorsForfait = $requetePrepare->fetchAll();
810     $montantFraisHorsForfait = 0;
811     foreach ($fraisHorsForfait as $unFraisHorsForfait) {
812         // Il ne faut pas prendre en compte les frais hors forfait refusés
813         if (substr($unFraisHorsForfait['libelle'], 0, 6) != 'REFUSE') {
814             $montantFraisHorsForfait+= $unFraisHorsForfait['montant'];
815         }
816     }
817     $ficheMontantValide = $montantFraisForfait + $montantFraisHorsForfait;
818     return $ficheMontantValide;
819 }

```

```

821  /**
822  * Met à jour le montant validé d'une fiche pour un visiteur donné
823  * et un mois donné
824  *
825  * @param String $idVisiteur id du visiteur
826  * @param String $mois mois sous la forme aaaamm
827  * @param String $montantValide montant validé pour la fiche
828  *
829  * @return null
830  */
831  public function majMontantValide($idVisiteur, $mois, $montantValide)
832  {
833      $requetePrepare = PdoGSB::$_monPdo->prepare(
834          'UPDATE fichefrais '
835          . 'SET montantvalide = :unMontantValide '
836          . 'WHERE idvisiteur = :unIdVisiteur AND '
837          . 'mois = :unMois'
838      );
839      $requetePrepare->bindParam(
840          'unMontantValide',
841          $montantValide,
842          PDO::PARAM_STR
843      );
844      $requetePrepare->bindParam('unIdVisiteur', $idVisiteur, PDO::PARAM_STR);
845      $requetePrepare->bindParam('unMois', $mois, PDO::PARAM_STR);
846      $requetePrepare->execute();
847  }
848  }
849

```

## V) Suivi du paiement des fiches de frais

J'ai implémenté une interface pour les comptables leurs permettant de suivre le paiement des fiches de frais des visiteurs. Voici l'interface :





Choisir le visiteur : Andre David ▼

Mois : 11/2019 ▼

Valider

## Fiche de frais du mois 11-2019 :

**Etat :** Saisie clôturée depuis le 04/12/2019  
**Montant validé :** 1365.08

## Éléments forfaitisés

Forfait Etape	Frais Kilométrique	Nuitée Hôtel	Repas Restaurant
0	329	8	2

## Descriptif des éléments hors forfait - 0 justificatifs reçus

Date	Libellé	Montant
04/11/2019	Voyage SNCF	87.00
27/11/2019	Taxi	65.00
11/11/2019	Traiteur, alimentation, boisson	27.00

Powered by

Ainsi, pour le visiteur et le mois sélectionné, on retrouve l'état de la fiche, le montant validé pour la fiche ainsi que l'ensemble des frais forfait et hors forfait.

Voici ci-dessous des captures d'écran de la vue qui affiche les valeurs :

```

18 <div class="panel panel-primary">
19   <div class="panel-heading">Fiche de frais du mois
20   <?php echo $numMois . '-' . $numAnnee ?> : </div>
21   <div class="panel-body">
22     <strong><u>Etat :</u></strong> <?php echo $libEtat ?>
23     depuis le <?php echo $dateModif ?> <br>
24     <strong><u>Montant validé :</u></strong> <?php echo $montantValide ?>
25   </div>
26 </div>
27 <div class="panel panel-info">
28   <div class="panel-heading">Eléments forfaitisés</div>
29   <table class="table table-bordered table-responsive">
30     <tr>
31       <?php
32       foreach ($lesFraisForfait as $unFraisForfait) {
33         $libelle = $unFraisForfait['libelle']; ?>
34         <th> <?php echo htmlspecialchars($libelle) ?></th>
35       <?php
36     }
37   ?>
38 </tr>
39   <tr>
40     <?php
41     foreach ($lesFraisForfait as $unFraisForfait) {
42       $quantite = $unFraisForfait['quantite']; ?>
43       <td class="qteForfait"><?php echo htmlspecialchars($quantite) ?></td>
44     <?php
45   }
46   ?>
47 </tr>
48 </table>

```

```

50 <div class="panel panel-info">
51   <div class="panel-heading">Descriptif des éléments hors forfait -
52   <?php echo htmlspecialchars($nbJustificatifs) ?> justificatifs reçus</div>
53   <table class="table table-bordered table-responsive">
54     <tr>
55       <th class="date">Date</th>
56       <th class="libelle">Libellé</th>
57       <th class="montant">Montant</th>
58     </tr>
59     <?php
60     foreach ($lesFraisHorsForfait as $unFraisHorsForfait) {
61       $date = htmlspecialchars($unFraisHorsForfait['date']);
62       $libelle = htmlspecialchars($unFraisHorsForfait['libelle']);
63       $montant = htmlspecialchars($unFraisHorsForfait['montant']); ?>
64       <tr>
65         <td><?php echo $date ?></td>
66         <td><?php echo $libelle ?></td>
67         <td><?php echo $montant ?></td>
68       </tr>
69     <?php
70   }
71   ?>
72 </table>
73 </div>

```

Ces 2 captures d'écran ci-dessus s'occupent d'afficher les différents frais récupérés dans la base de données en fonction du visiteur et du mois au préalable sélectionnés.

C'est également dans cette vue que l'on va pouvoir valider le remboursement d'une fiche, c'est-à-dire passer l'état d'une fiche à « remboursée » :

25/12/2019	Achat de matériel de papeterie	46.00
24/12/2019	Taxi	72.00
17/12/2019	Traiteur, alimentation, boisson	57.00
12/12/2019	Voyage SNCF	134.00
28/12/2019	Location équipement vidéo/sonore	716.00
24/12/2019	Taxi	67.00
12/12/2019	Location salle conférence	610.00

Valider le remboursement de la fiche

Power

Cette fois-ci, nous sommes dans le contrôleur suivreFrais et dans le case « rembourserFrais » :

```

124 case 'rembourserFrais':
125     $pdo->majEtatFicheFrais($idVisiteurSelectionne, $moisFicheSelectionne, 'RB');
126     break;
127 }
128

```

On appelle la méthode majEtatFicheFrais qui va passer la fiche à l'état « remboursée » et également mettre à jour la date du remboursement:

```

685 /**
686  * Modifie l'état et la date de modification d'une fiche de frais.
687  * Modifie le champ idEtat et met la date de modif à aujourd'hui.
688  *
689  * @param String $idVisiteur ID du visiteur
690  * @param String $mois      Mois sous la forme aaaamm
691  * @param String $etat      Nouvel état de la fiche de frais
692  *
693  * @return null
694  */
695 public function majEtatFicheFrais($idVisiteur, $mois, $etat)
696 {
697     $requetePrepare = PdoGSB::$_monPdo->prepare(
698         'UPDATE ficheFrais '
699         . 'SET idetat = :unEtat, datemodif = now() '
700         . 'WHERE fichefrais.idvisiteur = :unIdVisiteur '
701         . 'AND fichefrais.mois = :unMois'
702     );
703     $requetePrepare->bindParam(':unEtat', $etat, PDO::PARAM_STR);
704     $requetePrepare->bindParam(':unIdVisiteur', $idVisiteur, PDO::PARAM_STR);
705     $requetePrepare->bindParam(':unMois', $mois, PDO::PARAM_STR);
706     $requetePrepare->execute();
707 }
708

```

## VI) Renseigner une fiche de frais pour un visiteur

Voici, ci-dessous les interfaces et quelques explications du code source qui m'avait été fournies et les fonctionnalités déjà présentes :



## Renseigner ma fiche de frais du mois 03-2020

### Éléments forfaitisés

Forfait Etape

Frais Kilométrique

Nuitée Hôtel

Repas Restaurant

[Ajouter](#)[Effacer](#)

Descriptif des éléments hors forfait

Date	Libellé	Montant	
------	---------	---------	--

### Nouvel élément hors forfait

Date (jj/mm/aaaa):

Libellé

Montant :

€

[Ajouter](#)[Effacer](#)

Powered b

C'est dans l'onglet « Renseigner la fiche de frais » qu'un visiteur vient entrer les différents frais pour le mois en cours. Il ne peut modifier les frais que du mois en cours.

Il peut également, via l'onglet « Afficher mes fiches de frais », consulter ses différentes fiches de frais renseignées auparavant :



## Mes fiches de frais

Sélectionner un mois :

Mois :

12/2019

Valider

Effacer

Fiche de frais du mois 12-2019 :

**Etat :** Validée et mise en paiement depuis le 08/01/2020

**Montant validé :** 4535.65

Éléments forfaitisés

Forfait Etape	Frais Kilométrique	Nuitée Hôtel	Repas Restaurant
5	569	9	14

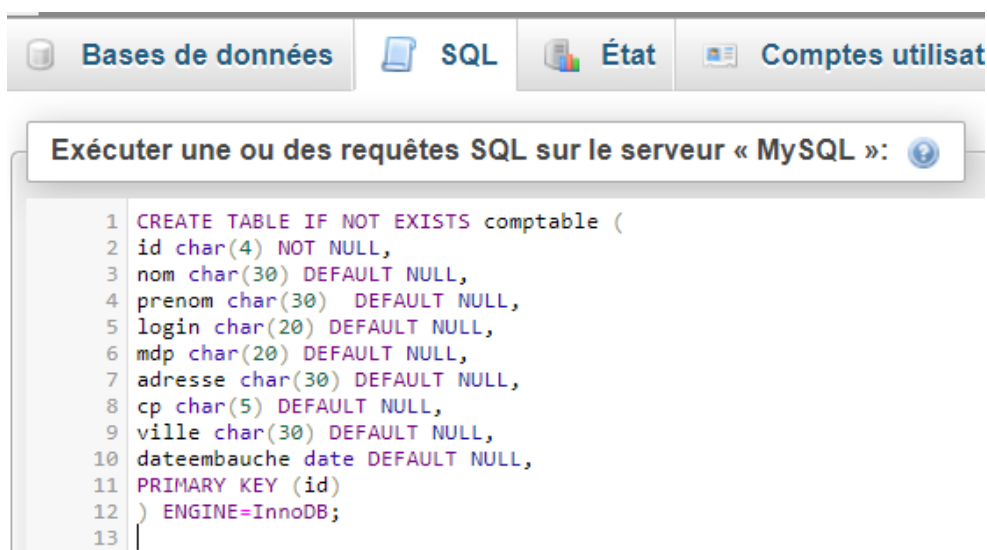
Descriptif des éléments hors forfait - 14 justificatifs reçus

Date	Libellé	Montant
10/12/2019	REFUSE Voyage SNCF	135.00

Power

## VII) Gestion de la base de données

Dans cette application, il était fondamental de gérer les comptes. Pour cela, il était également nécessaire d'avoir une table « Comptable » dans la base de données. Cette table n'était pas présente dans la base de données existantes qui m'a été fournie. Je me suis donc occupé de créer cette table « Comptable » sous phpmyadmin avec le script SQL suivant :



```
1 INSERT INTO comptable
2 VALUES ('c001', 'Goudet', 'Françoise', 'fgoudet', 'bcjh7', '12 rue des tulipes', '75001', 'Paris', '2006-03-22')
3 |
```

Également, il a fallu ajouter la colonne idComptable à la table fichefrais pour savoir quel comptable valide une fiche :

```
1 ALTER TABLE fichefrais
2 ADD COLUMN idComptable char(4),
3 ADD CONSTRAINT fk_comptable FOREIGN KEY (idComptable)
4 REFERENCES comptable (id)
5 |
```

## VIII) Gestion de version du projet

Ce projet a été réalisé avec Git qui permet la gestion du versionning de l'application, en association avec le service en ligne GitHub. Ainsi, cela m'a permis de gérer les différentes versions de l'application, de pouvoir faire des retours en arrière en cas d'erreurs et de surtout sécuriser mon projet ou cas où j'aurais eu des problèmes de matériel par exemple.

Florian935 / GSB

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Actions Projects 0 Wiki Security Insights Settings

Création du repo pour la mission 1 du PPE. Edit

Manage topics

107 commits 12 branches 0 packages 0 releases 1 contributor

Branch: development New pull request Create new file Upload files Find file Clone or download

Florian935 Merge pull request #36 from Florian935/fichierAccesBddPourAppliAndroid Latest commit d3bd487 yesterday

.vscode	Implémentation du code permettant de detecter le type d'utilisateur q...	4 months ago
accesBddAndroid	Merge pull request #36 from Florian935/fichierAccesBddPourAppliAndroid	yesterday
controleurs	Ajout de commentaires permettant une meilleure compréhension de certa...	2 months ago
docGSB	Génération de la documentation technique du projet	2 months ago
hashpassword	Implémentation du hashage des mdp dans la BDD grâce à l'algorithme sh...	2 months ago
images	initialisation d'un repository GitHub pour versionner mon code.	5 months ago
includes	WIP - résolution du problème de connexion refusée sur le serveur dist...	2 months ago
styles	Travail sur le design et le wording	2 months ago
tests	Resolve conflict	2 months ago







## IX) Déploiement de l'application sur le serveur

Le déploiement sur le serveur a été réalisé avec le service en ligne DeployBot qui permet, lorsqu'un push est réalisé sur la branche master de mon repos en ligne, d'envoyer en un simple clic l'application sur le serveur.

Au préalable, j'ai bien-sûr par exemple renseigné le hostname du serveur et le mot de passe du serveur afin de pouvoir associer mon compte DeployBot et mon dépôt GitHub au serveur pour effectuer le déploiement.

Voici un aperçu des déploiements que j'ai réalisé :



Tue, Jan 28 4:29 pm		GSB → <input type="radio"/> Production <b>e2ce2195:</b> Merge pull request #32 from Florian935/fichierAccesBddPourAppliAndroid Merge fichierAc...
Tue, Jan 28 4:04 pm		GSB → <input type="radio"/> Production <b>aa7db9c6:</b> Update serveurgsb.php
Tue, Jan 28 3:40 pm		GSB → <input type="radio"/> Production <b>4644cc2e:</b> Merge pull request #31 from Florian935/fichierAccesBddPourAppliAndroid Merge fichierAc...
Tue, Jan 28 3:19 pm		GSB → <input type="radio"/> Production <b>467f0546:</b> Merge pull request #30 from Florian935/fichierAccesBddPourAppliAndroid Merge fichierAc...
Mon, Jan 13 4:39 pm		GSB → <input type="radio"/> Production <b>560f46cb:</b> Merge pull request #17 from Florian935/validerFraisComptable Merge validerFraisCompta...
Thu, Jan 9 5:37 pm		GSB → <input type="radio"/> Production <b>9af09738:</b> Merge pull request #15 from Florian935/hachageDesMotsDePasse Merge hachageDesMot...

## X) Documentation technique

Voici la commande pour générer la documentation (en se plaçant au préalable à la racine du projet en ligne de commande) :

```
vendor\bin\phpdoc.bat -d "tests/gendatas" -t "./docGSB/gendatas" --template="responsive-twig"
```

Voici quelques explications :

-d (ou --directory): dossier à partir duquel on doit générer la documentation

-t: dossier dans lequel on veut générer la doc.

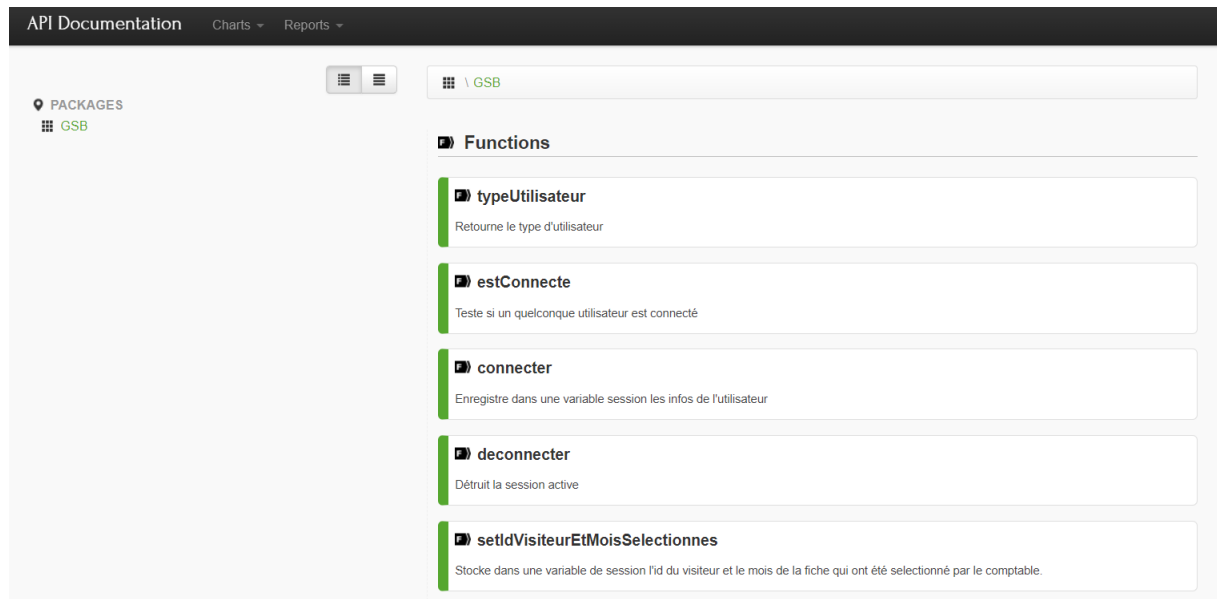
--template: template que l'on souhaite utiliser pour afficher la documentation

Ensuite, il suffit d'aller à l'URL de l'application avec le suffixe « /docGSB » pour voir la documentation générée.

Voici le lien vers la documentation en ligne de l'application :

<https://gsb-gestion-frais.000webhostapp.com/docGSB/>

Ci-dessous, un extrait de la documentation :



## XI) Tests unitaires

Des tests unitaires ont été écrit pour chaque fonction de l'application afin de tester les valeurs de retours et afin de savoir si les fonctions effectuent les bons traitements. Voici des extraits :

```

114     /**
115     * Teste que la fonction getInfosComptable retourne l'id du comptable
116     * associé au login et au mdp fourni en paramètre.
117     *
118     * @return null
119     */
120     public function testGetInfosComptableIdCorrect()
121     {
122         $comptable = PdoGsbTest::$_monPdoGsb->getInfosComptable('fgoudet', 'bcjh7');
123         $id = $comptable['id'];
124         $this->assertEquals('c001', $id);
125     }
126
127     /**
128     * Teste que la fonction getInfosComptable retourne le nom du comptable
129     * associé au login et au mdp fourni en paramètre.
130     *
131     * @return null
132     */
133     public function testGetInfosComptableNomCorrect()
134     {
135         $comptable = PdoGsbTest::$_monPdoGsb->getInfosComptable('fgoudet', 'bcjh7');
136         $nom = $comptable['nom'];
137         $this->assertEquals('Goudet', $nom);
138     }

```

Les tests de l'application peuvent être lancés avec la commande suivante :

```

C:\wamp64\www\GSB>vendor\bin\phpunit ./tests/unitTest

Warning: The use statement with non-compound name 'PDO' has no effect in C:\wamp64\www\GSB\tests\unitTest\class.pdogsb.incTest.php on line 4
PHPUnit 8.4.3 by Sebastian Bergmann and contributors.

.....                                         49 / 49 (100%)

Time: 1.25 seconds, Memory: 6.00 MB

OK (49 tests, 71 assertions)

C:\wamp64\www\GSB>

```

Ainsi, on peut voir ici que tous les tests sont passés.