

# **GSB FRAIS-** **Documentation technique**



# PLAN

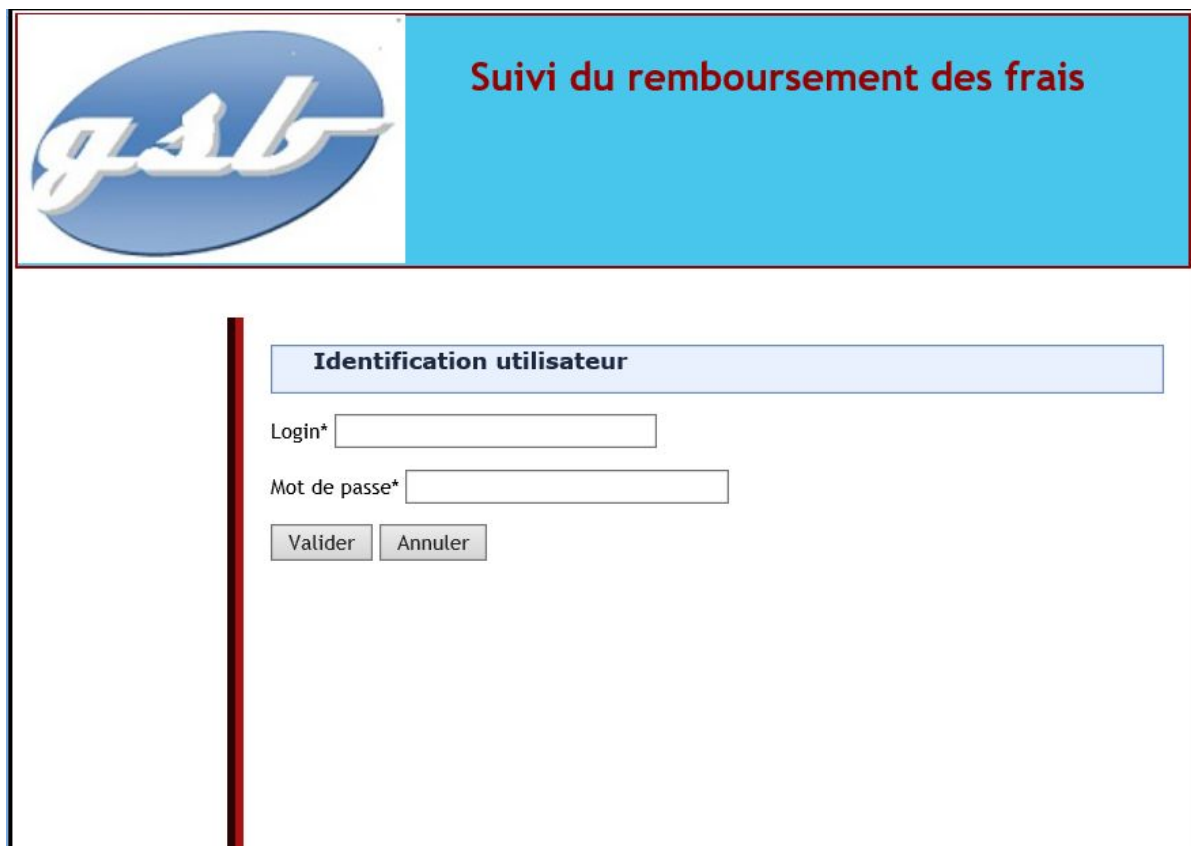
<b><u>1.Contexte de la situation professionnelle</u></b>	p3
<b><u>.Le besoin/les objectifs</u></b>	p3
<b><u>2.Environnement de développement</u></b>	p4/5
<b><u>3.Modélisation fonctionnelle</u></b>	p6/7
<b><u>4.IHM</u></b>	p8
<b><u>5.Architecture applicative</u></b>	p9-12
<b><u>6.Persistance des données</u></b>	p13/14
<b><u>7.Accès aux données (pdo singleton)</u></b>	p15
<b><u>8. Déploiement, mise en production de l'application</u></b>	p16
<b><u>9. Tests fonctionnels</u></b>	p16/17
<b><u>10.Conclusion, améliorations possibles</u></b>	p17

# 1.Contexte de la situation professionnelle :

## L'objectif:

Nous avons un client léger permettant la création et consultations de fiche de frais.

En ce moment cette application ne permet qu'au utilisateur de créer leurs fiches et il est impossible pour les comptables de les valider. Nous devons ajouter un profil comptable pour qu'ils puisse agir en tant que tel sur l'application (validation,refus... des fiches de frais).



**gslb**

**Suivi du remboursement des frais**

**Identification utilisateur**

Login\*

Mot de passe\*

Valider Annuler

## **2.Environnement de développement:**

### **.Outil de développement:**

#### **A. NetBeans**



Nous allons utiliser "NetBeans" comme environnement de développement . NetBeans est un environnement de développement intégré IDE (Integrated Development Environment), placé en open source en 2000 par Sun sous licence CDDL (Common Development and Distribution License). NetBeans permet la prise en charge de divers langages tels que php, javascript, html et ces librairies (Jquery par exemple)..

#### **B. WampServer**



Nous avons également utilisé Wampserver. Il s'agit d'une plateforme de développement web que nous utilisons pour la base de données de l'application client.

## C. GitHub



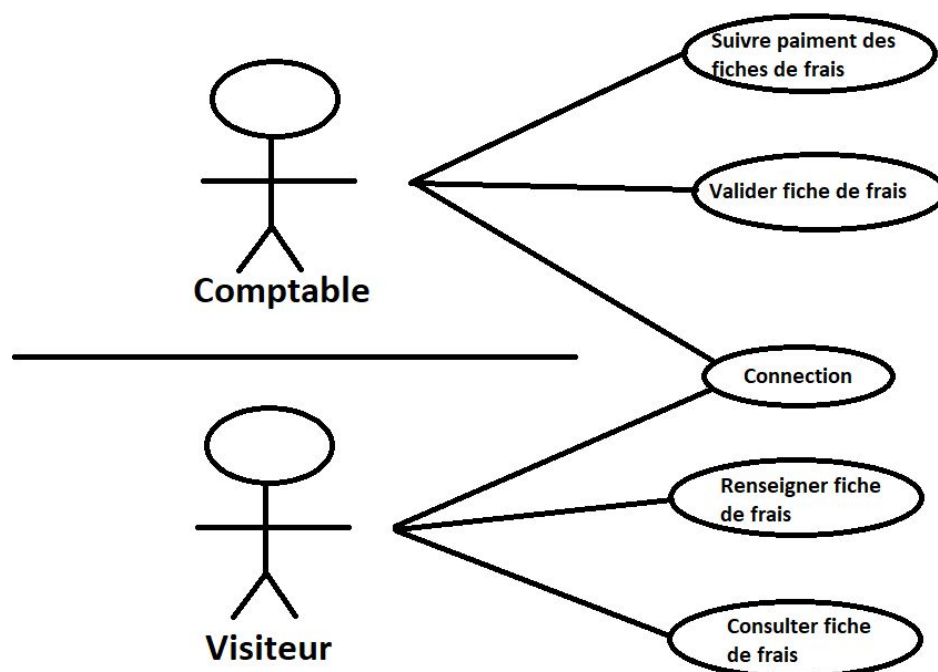
nous utiliserons Github comme outil de travail collaboratif.

GitHub est un service web d'hébergement, de gestion de développement de logiciels. Ce site assure un contrôle d'accès et de fonctionnalités destinées à la collaboration.

### 3.Modélisation fonctionnelle

#### cas d'utilisation pour comptable

Projet	Application web de gestion des frais
pré-condition	utilisateur enregistré et reconnu comme comptable
utilisation	saisir et confirmer une fiche de frais
scénario nominal	<p>1-visiteur sélectionne le menu "fiche de frais" dans le sommaire, sous le sous menu administration</p> <p>2-système retourne la liste des visiteur ayant une fiche de frais à confirmer</p> <p>3- utilisateur choisit un visiteur de la liste ainsi qu'un mois</p> <p>4-système retourne les notes de frais de visiteur pour le mois choisis (y compris hors forfait)</p> <p>5-visiteur valide les notes de frais ou refuse</p> <p>6-système enregistre le nouveau statut de la note de frais du visiteur</p>
Exception	<p>2- aucune note de frais n'a été remplis pour le moment, le système ne retourne rien</p> <p>3-le système affiche un message d'erreur si l'utilisateur tente de se sélectionner lui-même</p> <p>5-le système renvoi un message d'erreur si les note de frais sont déjà validé. SI les note de frais sont refusé le système demande confirmation</p>



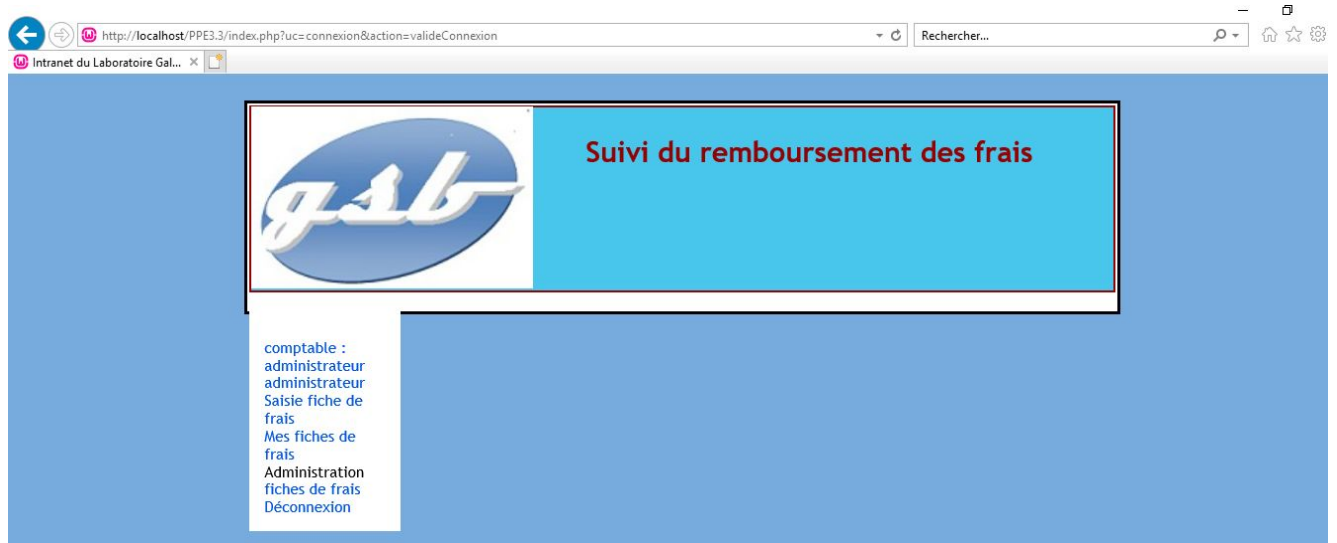
## Le fonctionnement.

Lorsque l'utilisateur se connecte à l'application son statut (visiteur ou comptable) est immédiatement vérifié et gardé en mémoire.

Pour un comptable de nouveaux choix apparaissent dans le menu sous "administration", c'est ici que se trouvent les opérations administratives (tel que valider les notes de frais pour le comptable).

Ci-dessous un extrait du code permettant cette action ainsi qu'un aperçu comme end user.

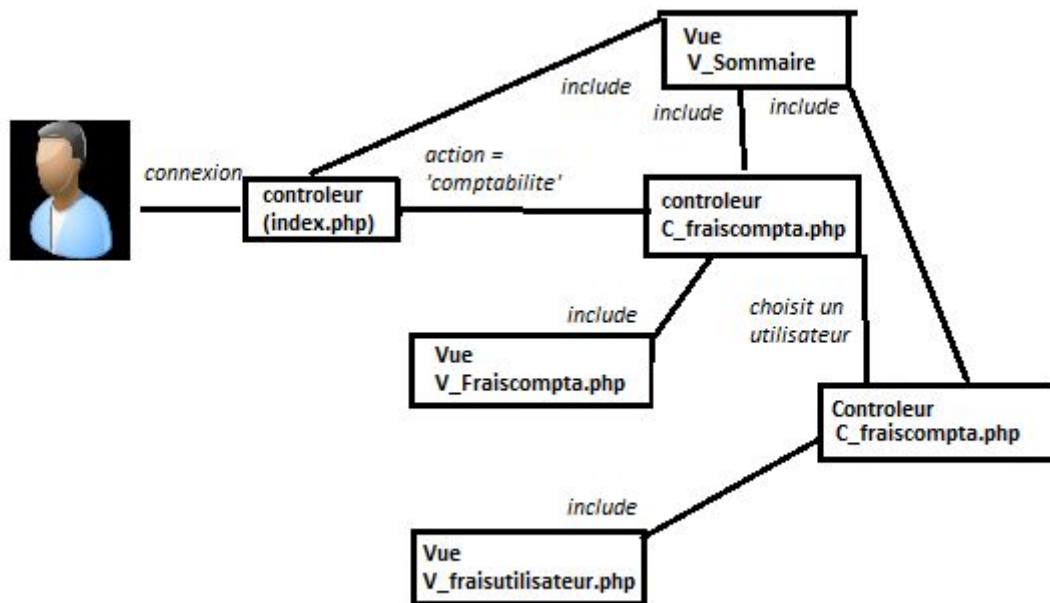
```
<?php if($_SESSION['type_utilisateur']=='comptable'){
echo "<li style='color:black;' class='smenu'> <span >Administration</span> </li>
<li class='smenu'>
<a href='index.php?uc=comptabilite&action=rien' title='Consultation de fiches de frais'> fiches de f:
</li>";
} ?>
```



lorsque l'utilisateur clique sur "fiche de frais" l'index le reconduit à 'c\_fraisconmpta.php'.

À partir de cette page il pourra choisir un utilisateur ayant fait sa note de frais afin d'accéder aux frais de celui-ci.

nous pouvons résumer cela par un schéma.





## **4.Interactions homme-machine, IHM**

IHM définit les moyens et outils mis en œuvre afin qu'un utilisateur puisse contrôler et communiquer avec une machine. Les ingénieurs en ce domaine étudient la façon dont les humains interagissent avec les ordinateurs ou entre eux à l'aide d'ordinateur. Ils étudient aussi la façon de concevoir des systèmes ergonomiques et faciles à utiliser.



Dans notre situation nous communiquons avec l'application (visiteur ou comptable) en lui demandant de créer, modifier, valider des notes de frais.

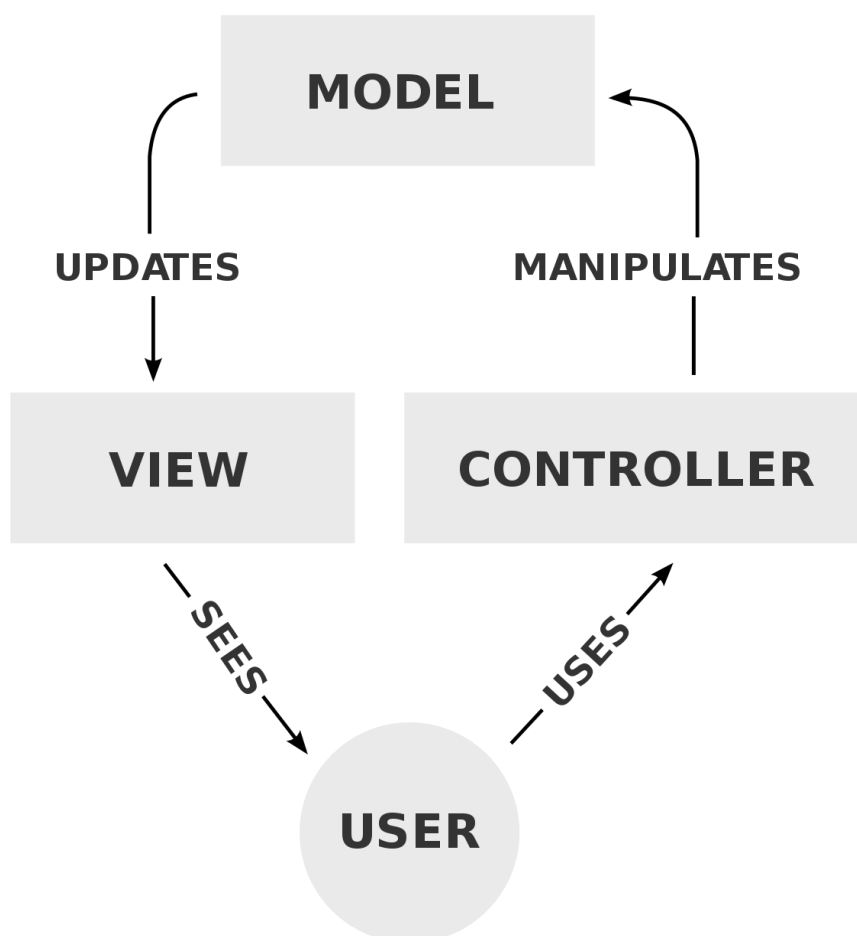
L'application a été modélisée pour afficher les réponses de l'ordinateur de façon claire et facile.

## 5.Architecture Applicative

Le modèle que nous suivons, le modèle MVC, est extrêmement utile s'il y a nécessité d'améliorer le code utilisé ou de régler une erreur. En effet il répartit en trois modules coopératifs toutes les capacités que l'on souhaite donner au programme, ce qui facilite la maintenance par le programmeur.

Pour résumer les interactions entre le modèle, la vue et le contrôleur.

La vue affiche les données et gère l'aspect de l'application. Elle reçoit ses données du modèle qui les demande à la base de données. Et les fonction du modèle sont réclamé par les contrôleurs qui sont enclenché par l'utilisateur. le graphique ci dessous le démontre.



L'intérêt du MVC baisse cependant pour les projets de petites envergure, tel que des sites simples. En effet le modèle augmente la complexité de l'architecture du programme et sépare en trois modules ce qui pourrait être fait dans un seul. En prenant l'exemple d'un site simple le modèle MVC change une page web codée dans un module en une page web codée en trois modules, ceci pour chaque pag

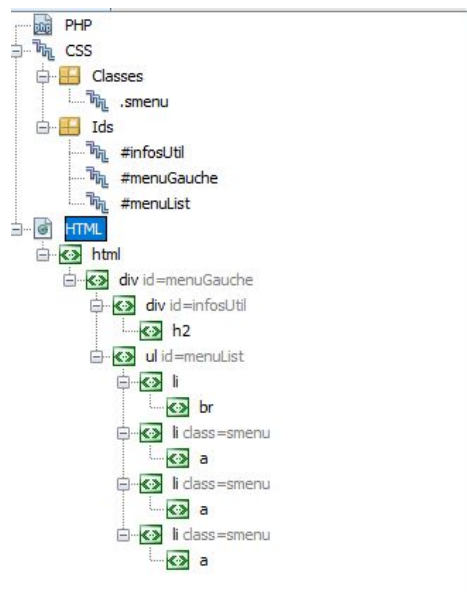
## Rôle de la vue

Le rôle de la vue est considérablement visuel. En effet elle s'occupe de l'affichage du contenu sur l'écran (caractères , dimensions, boutons, données...). Elle doit récupérer les informations provenant du modèle.

Par exemple ci-dessous la vue V\_Sommaire affiche le sommaire permettant le déplacement sur l'application. Cette vue est l'une des nombreuses utilisés.



Voici sa structure.



## Rôle du Contrôleur

Le contrôleur gère les interactions avec l'utilisateur ainsi que les traitements devant être effectués pour une action donnée. Il va utiliser les données du modèle et les traiter en fonction de l'action de l'utilisateur avant de les envoyer à la vue afin qu'elle les affiche.

Voici un extrait du contrôleur C\_etatsfrais

```
<?php
include("vues/v_sommaire.php");
$action = $_REQUEST['action'];
$idVisiteur = $_SESSION['idVisiteur'];
switch($action){
    case 'selectionnerMois':{
        $lesMois=$pdo->getLesMoisDisponibles($idVisiteur);
        // Afin de sélectionner par défaut le dernier mois dans la zone de liste
        // on demande toutes les clés, et on prend la première,
        // les mois étant triés décroissants
        $lesCles = array_keys( $lesMois );
        $moisASelectionner = $lesCles[0];
        include("vues/v_listeMois.php");
        break;
    }
}
```

Ce contrôleur inclut la vue sommaire afin de permettre le déplacement et permet grâce à un switch d'effectuer diverses action selon les actions de l'utilisateur.

Dans cet extrait cela affiche le mois actuel dans le dropdown de la vue V\_listemois et permet de choisir un autre mois si possible. (on sélectionne le mois dans le dropdown ci-dessous).

Mois à sélectionner :

Mois : 12/2018 ▼
<input type="button" value="Valider"/> <input type="button" value="Effacer"/>

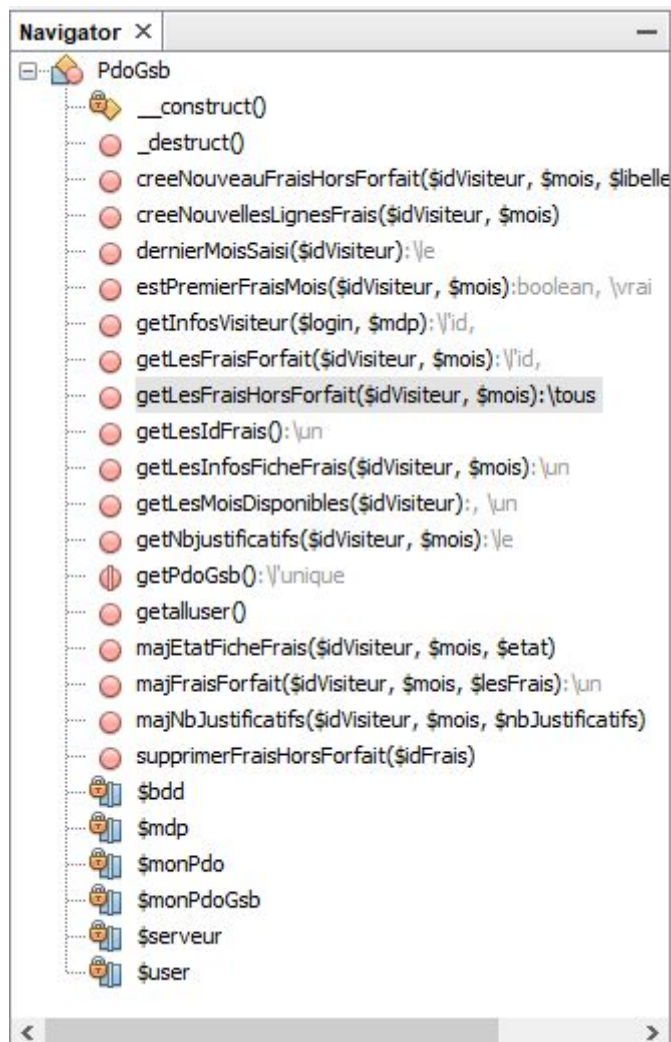
## Rôle du Modèle

Le modèle gère la logique métier régissant l'accès aux données dont il est responsable .

Par exemple le code ci-dessous récupère de la base de données tous les frais hors forfait d'un visiteur pendant un mois donnée.

```
public function getLesFraisHorsForfait($idVisiteur,$mois){
    $req = "select * from lignefraishorsforfait where lignefraishorsforfait.idvisiteur ='$idVisiteur'
    and lignefraishorsforfait.mois = '$mois' ";
    $res = PdoGsb::$monPdo->query($req);
    $lesLignes = $res->fetchAll();
    $nbLignes = count($lesLignes);
    for ($i=0; $i<$nbLignes; $i++){
        $date = $lesLignes[$i]['date'];
        $lesLignes[$i]['date'] = dateAnglaisVersFrancais($date);
    }
    return $lesLignes;
}
```

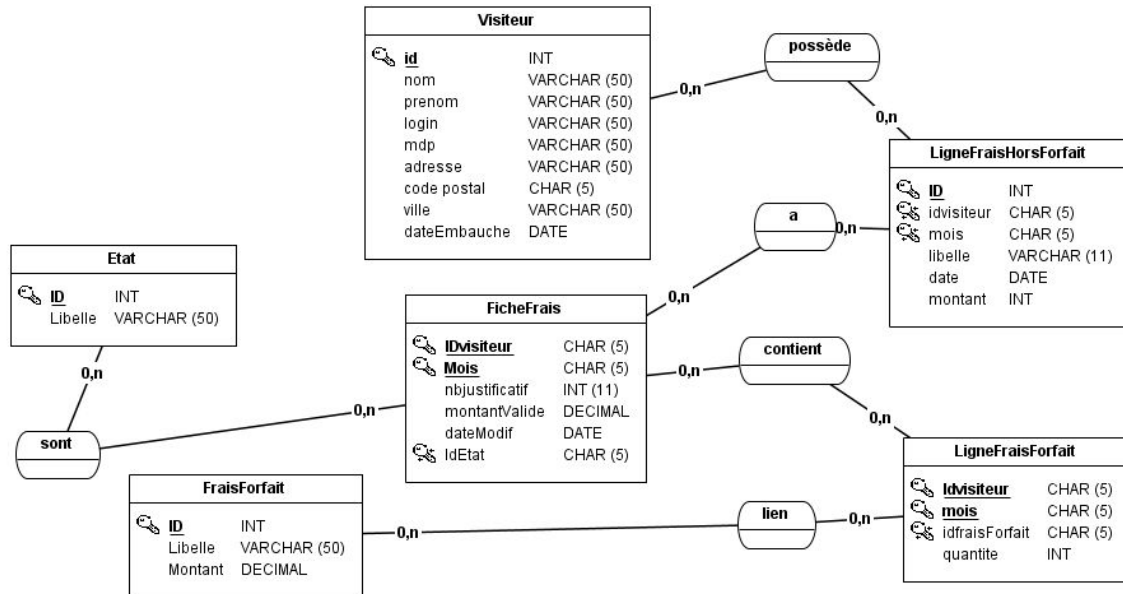
## architecture du modèle



## 6.Persistance des données

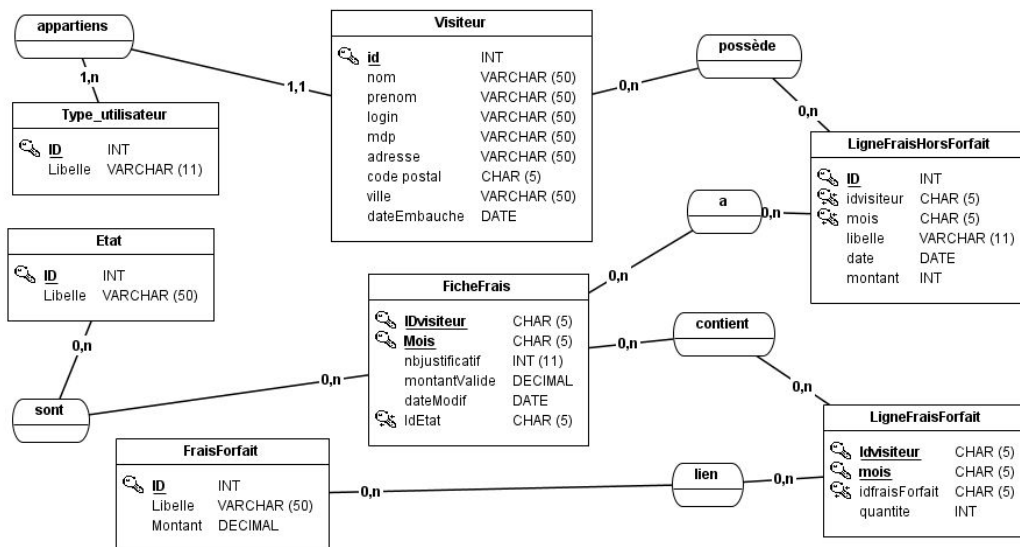
Nous avons modifié la base de données afin de subvenir à nos nouveaux besoins.

### Base de données original



### base de données adaptée

Nous avons rajouté une table type\_utilisateur afin d'authentifier le rang des visiteurs du site et de pouvoir déterminer quel droit doivent leurs être accordé.



Voici un extrait du code utilisé pour la création de la base de donnée

```
CREATE TABLE IF NOT EXISTS `LigneFraisHorsForfait` (
  `id` int(11) NOT NULL auto_increment,
  `idVisiteur` char(4) NOT NULL,
  `mois` char(6) NOT NULL,
  `libelle` varchar(100) DEFAULT NULL,
```

```
`date` date DEFAULT NULL,  
`montant` decimal(10,2) DEFAULT NULL,  
PRIMARY KEY (id),  
FOREIGN KEY (`idVisiteur`, `mois`) REFERENCES FicheFrais(`idVisiteur`, `mois`)  
) ENGINE=InnoDB;
```

## 7.Accès aux données, connexion

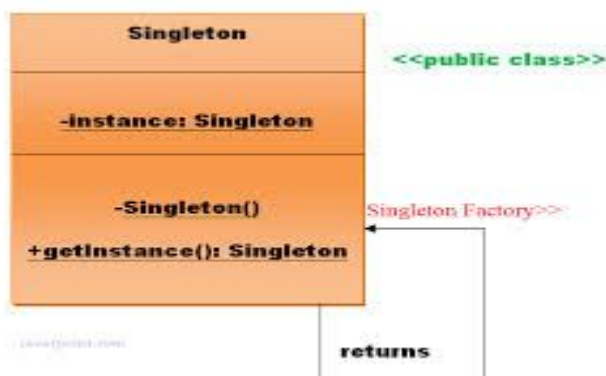
### Php Data Object, PDO



PDO est une extension définissant l'interface d'accès à une base de données en utilisant PHP.

Pour récupérer les données d'une base de donnée la procédure PDO consiste à prendre en une seule fois tous les enregistrements de la table sous forme d'une (étant un tableau à deux dimensions). Cela réduit le temps de traitement par rapport à la procédure classique consistant à récupérer les données ligne par ligne.

### Design Pattern Singleton, DPS



Le Design Pattern Singleton restreint l'instanciation d'une classe à un seul objet . Il est utilisé lorsqu'on a besoin exactement d'un objet pour coordonner des opérations dans un système.

Le singleton se crée en écrivant une classe contenant une méthode qui crée une instance privée s'il n'en existe pas encore. Sinon elle renvoie vers l'instance qui existe déjà.

## **8.Déploiement, mise en production de l'application**

la mise en production de la nouvelle version de l'application s'effectuera d'abord en modifiant la base de donnée afin d'ajouter la nouvelle table déterminant le statut de chaque utilisateur (en mettant par défaut celui de visiteur). Il faudra par la suite mettre à jour dit statuts pour les comptables.

Par la suite le code modifié de l'application sera uploadé en production.

## **9.Tests fonctionnels**

les tests fonctionnel sont des tests qui servent à vérifier automatiquement toutes les fonctionnalités de l'application. Par "toutes", on veut dire : les fonctionnalités qui étaient demandées dans le cahier des charges du projet.

Nous allons vérifier qu'un comptable peut se connecter et modifier ainsi que valider une fiche de frais avant de vérifier qu'un simple utilisateur ne peut pas faire de même.

N	Action	Résultat attendu	Réponse
1	le comptable saisit ses identifiants (login et mot de passe) et valide.	Le comptable se connecte. Son nom, prénom, statut (comptable) apparaît en haut à gauche.  En tant que comptable on peut voir un nouveau sous menu apparaître.	OK
2	le comptable sélectionne "fiche de frais"	le système fait apparaître la liste d'utilisateurs ayant une liste de frais en attente sur le mois actuel	OK
3	le comptable sélectionne un mois spécifique puis valide.	Le système ne fait apparaître que les utilisateurs ayant des notes de frais pour le mois choisis (valider ou non)	OK



4	le comptable sélectionne un utilisateur de la liste.	Les note de frais de l'utilisateur apparaissent pour le mois choisis (ou le mois actuel si aucun mois n'a été spécifié)	
5	le comptable change le nombre de justificatif et valide une note de frais	la note de frais est modifié et validé. le système retourne les nouvelles données de la fiche à l'écran	
6	Le comptable se déconnecte	le système déconnecte le comptable	OK
7	Saisir l'identifiant d'un utilisateur classique	utilisateur se connecte, il n'a pas de menu administratif de visible. Son nom, prénom, statut (visiteur) apparaît en haut à gauche	OK
8	se déconnecte	le système déconnecte l'utilisateur	OK

## **10.Conclusion, améliorations possibles**

Bien que l'application soit fonctionnelle de cette façon il est possible de l'améliorer.

### **Proposition:**

**-1**

sécurisé les mots de passe. Il est préférable d'établir un modèle de mot de passe que les utilisateurs devront suivre (par exemple le mot de passe doit contenir des caractères spéciaux et être long d'au moins 8 caractères). Il faut également s'assurer qu'ils ne soient pas visible au clair dans la base de données.

**-2:**

fin de session. En ce moment lorsque l'on revient en arrière après avoir quitté sa session nous sommes reconnecté à celle-ci (sans avoir entré de mot de passe ou login à nouveau !!), il faut changer cela par mesure de sécurité.

**-3:**

Etablir un tri des note de frais selon plusieurs option (id visiteur, nom visiteur, frais total,...)

**-4:**

