

Introduction

Nous allons développer l'application gsb-frais dans l'environnement Laravel. Cette application permet aux visiteurs médicaux de saisir les différents frais qu'ils ont eus dans le cadre de leurs missions.

Lister les fiches de frais

Pour réaliser cette fonctionnalité, nous aurons besoin des composants suivants :

Composant	Valeur	Rôle
Route	/getListeFrais	Prise en compte du clic sur l'item Lister du menu
Vue	listeFrais.blade.php	Vue listant tous les Frais du visiteur connecté
Classes métier	Frais	Stockage des données et requêtes
Contrôleur	FraisController	Pilotage de la demande de liste
Méthode métier	getFrais()	Recherche dans la base de données la liste des Frais du visiteur
Méthode contrôleur	getFraisVisiteur()	Affichage du formulaire et appel de la méthode métier geFrais()

Les données

Pour lister les Frais il faut qu'il y en ait dans la base et comme on n'a pas encore développé la fonctionnalité d'ajout, il faut rajouter manuellement deux ou trois lignes dans la table frais.



Ajouter quelques lignes de frais dans la table frais.

La route

La route getListeFrais de type get appelle la méthode getFraisVisiteur du contrôleur FraisController.



Ajouter la nouvelle route.

La classe métier Frais

```
namespace App\metier;
use Illuminate\Database\Eloquent\Model;
use DB;

class Frais extends Model {

    //On déclare la table Frais
    protected $table = 'frais';
    public $timestamps = false;
    private $id_frais;
    protected $fillable = [
        'id_frais',
        'id_etat',
        'anneemois',
        'id_visiteur',
        'nbjustificatifs',
        'datemodification',
        'montant_valide'
    ];

    public function __construct() {
        $this->id_frais = 0;
    }
}
```

Rien de particulier si ce n'est que l'on a ajouté la propriété privée `id_frais` et qu'on l'initialise à 0 dans le constructeur. On en aura besoin pour l'ajout d'un Frais.



Créer la classe métier Frais.

Le contrôleur FraisController

```
<?php

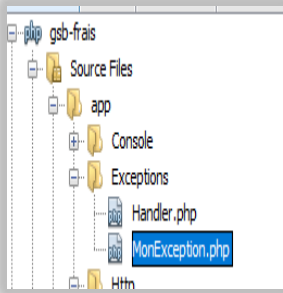
namespace App\Http\Controllers;

use Request;
use Illuminate\Support\Facades\Session;
use App\metier\Frais;
use Exception;
use App\Exceptions\MonException;

class FraisController extends Controller {
```

Il est quasi standard si ce n'est qu'il fait référence au package Session dont nous aurons besoin dans la méthode `getFraisVisiteur()`.

La gestion des exceptions est assurée par la classe personnalisée `MonException` qui se trouve dans :



```
<?php

namespace App\Exceptions;
use Exception;

class MonException extends Exception
{
    protected $message = 'Unknown exception'; // Message d'erreur
    private $string; // inconnu
    protected $code = 0; // Code d'erreur de l'utilisateur
    protected $file; // fichier source
    protected $line; // ligne de l'erreur
    private $trace; // inconnu

    public function __construct($message, $code = 0, Exception $previous = null)
    {
        if (!$message) {
            throw new $this('Unknown '. get_class($this));
        }
        parent::__construct($message, $code);
    }

    public function __toString()
    {
        return get_class($this) . " '{$this->message}' in {$this->file}({$this->line})\n"
            . "{$this->getTraceAsString()}";
    }
}
```



Créer le contrôleur `FraisController`.

La méthode `getFraisVisiteur()`

```
/**
 * Affiche la liste de toutes les fiches
 * de Frais du visiteur connecté
 * Si une erreur a été stockée dans la Session
 * on la récupère, on l'efface de la Session
 * et la passe au formulaire
 * @return type Vue listeFrais
 */
public function getFraisVisiteur() {
    try
    {
        $erreur = Session::get('erreur');
        Session::forget('erreur');
        $unFrais = new Frais();
        $id_visiteur = Session::get('id');
        // On récupère la liste de tous les mangas
        $mesFrais = $unFrais->getFrais($id_visiteur);
        // On affiche la liste de ces mangas
        return view('listeFrais', compact('mesFrais', 'erreur'));

        } catch (MonException $e) {
            $monErreur = $e->getMessage();
            return view('vues/pageErreur', compact('monErreur'));
        } catch (Exception $e) {
            $monErreur = $e->getMessage();
            return view('vues/pageErreur', compact('monErreur'));
        }
    }
}
```

Il nous faut lister les Frais du visiteur connecté, pour cela nous avons besoin de son id ce qui permettra de faire une requête sql avec une restriction sur l'id du visiteur. Comme par hasard nous disposons de cet id dans la variable de session 'id', il ne reste donc plus qu'à récupérer la valeur de cet id, c'est ce qui est fait à l'aide de la méthode get() de la classe Session. Ensuite il faudra appeler la méthode getFrais() de l'objet métier Frais en lui passant cet id.



Créer la méthode getFraisVisiteur().

La méthode getFrais()

```
/**
 * Lecture de toutes les fiches de Frais
 * d'un visiteur dont l'id est passé en paramètre
 * @param type $id_visiteur id du visiteur
 * @return type Collection de Frais
 */
public function getFrais($id_visiteur) {
    try {

        $lesFrais = DB::table('frais')
            ->Select()
            ->where('frais.id_visiteur', '=', $id_visiteur)
            ->get();
        return $lesFrais;
    }
    catch (QueryException $e) {
        throw new MonException($e->getMessage(), 5);
    }
}
```

Elle ne présente aucune difficulté.



Créer la méthode getFrais().

La vue listeFrais.blade.php

```
<table class="table table-bordered table-striped table-responsive">
  <thead>
    <tr>
      <th style="width:60%">Période</th>
      <th style="width:20%">Modifier</th>
      <th style="width:20%">Supprimer</th>
    </tr>
  </thead>
  @foreach($mesFrais as $unFrais)
    <tr>
      <td>{{ $unFrais->anneemois }} </td>
      <td style="text-align:center;"><a href="{{ url('/modifierFrais') }}/{{ $unFrais->id_frais }}">
        <span class="glyphicon glyphicon-pencil" data-toggle="tooltip" data-placement="top" title="Modifier">
      <td style="text-align:center;">
        <a class="glyphicon glyphicon-remove" data-toggle="tooltip" data-placement="top" title="Supprimer"
          onclick="javascript:if (confirm('Suppression confirmée ?'))
            { window.location = '{{ url('/supprimerFrais') }}/{{ $unFrais->id_frais }}'; }">
      </a>
    </td>
  </tr>
  @endforeach
</table>
@include('error')
```

Elle balaye la collection de Frais \$mesFrais que lui a fourni le contrôleur avec un foreach très classique. On remarquera que l'on utilise un script javascript pour demander confirmation d'une suppression.



Récupérer la vue listeFrais et la compléter.

L'url de l'item Lister



Ajouter l'url à l'item de menu Lister et tester.

GSB Frais	Lister	Ajouter	Se déconnecter
-----------	--------	---------	----------------

Liste des frais		
Période	Modifier	Supprimer
201607		
201608		
201609		

Modifier une fiche de Frais

Pour réaliser cette fonctionnalité, nous aurons besoin des composants suivants :

Composant	Valeur	Rôle
Routes	/modifierFrais	Prise en compte du clic sur le bouton Modifier de la liste des frais
	/validerFrais	Prise en compte du clic sur le bouton Valider du formulaire
Vue	formFrais.blade.php	Formulaire affichant la fiche de Frais sélectionnée et permettant sa modification
Classes métier	Frais	Stockage des données et requêtes
Contrôleur	FraisController	Pilotage de la demande de modification et de la validation
Méthodes métier	getById()	Lecture de la fiche de Frais
	updateFrais()	Enregistrement des modifications
Méthodes contrôleur	updateFrais()	Affichage du formulaire et appel de la méthode métier getById()
	validateFrais()	Récupération des données saisies et appel de la méthode métier updateFrais()

Les routes

Vous disposez de toutes les informations nécessaires pour construire les deux routes (tableau ci-dessus et vue listeFrais.blade.php). Un indice : une des deux routes dispose d'un paramètre !



Ajouter les deux routes nécessaires.

La méthode updateFrais()

```
/**
 * Initialise le formulaire de Frais pour une modification
 * Lit la fiche de Frais
 * Initialise le titre de la vue
 * @param type $id Id de la fiche de Frais à modifier
 * @return type Vue formFrais
 */
public function updateFrais($id_frais) {
    try
    {
        $erreur = "";
        $unFrais = new Frais();
        $unFrais = $unFrais->getById($id_frais);
        $titreVue = "Modification d'une fiche de Frais";
        // Affiche le formulaire en lui fournissant les données à afficher
        return view('formFrais', compact('unFrais', 'titreVue', 'erreur'));
    } catch (MonException $e) {
        $monErreur = $e->getMessage();
        return view('vues/pageErreur', compact('monErreur'));
    } catch (Exception $e) {
        $monErreur = $e->getMessage();
        return view('vues/pageErreur', compact('monErreur'));
    }
}
```

La méthode ne présente pas de difficulté, on remarquera simplement que l'on passe au formulaire un objet \$unFrais et donc que les valeurs à afficher dans ce formulaire devront faire référence à cet objet. D'autre part si on passe la variable \$erreur initialisée à vide, cela signifie que l'on aura inclus la vue error dans le formulaire. Si une erreur plus conséquente survient (sql), on enverra le code de l'erreur à une page nommée pageErreur.



Saisir la méthode updateFrais

La méthode validateFrais()


```

/**
 * Enregistre un ajout ou une modification d'une fiche de Frais
 * Si id_frais > 0 c'est une modification, sinon
 * c'est un ajout. Id_frais est dans un INPUT HIDDEN
 * @return type route /getListeFrais
 */
public function validateFrais() {
    try
    {
        $id_frais = Request::input('id_frais');
        $anneemois = Request::input('anneemois');
        $nbjustificatifs = Request::input('nbjustificatifs');
        $unFrais = new Frais();
        if ($id_frais > 0) {
            $unFrais->updateFrais($id_frais, $anneemois, $nbjustificatifs);
        } else {
            $id_visiteur = Session::get('id');
            $unFrais->insertFrais($anneemois, $nbjustificatifs, $id_visiteur);
        }
        // Retourne à la liste des des Frais
        return redirect('/getListeFrais');
    } catch (MonException $e) {
        $monErreur = $e->getMessage();
        return view('vues/pageErreur', compact('monErreur'));
    } catch (Exception $e) {
        $monErreur = $e->getMessage();
        return view('vues/pageErreur', compact('monErreur'));
    }
}

```

On commence par récupérer l'id de la fiche de Frais qui avait été placé dans le champ INPUT HIDDEN, puis les valeurs saisies, on appelle la méthode métier updateFrais en lui passant les paramètres nécessaires et enfin on demande à réafficher la liste des frais à l'aide de l'instruction redirect().



Saisir la méthode validateFrais().

Le formulaire formFrais

Le formulaire est disponible dans les Ressources, il suffira de le compléter avec les informations prises dans les méthodes du contrôleur. On remarquera le champ INPUT HIDDEN dans lequel sera stocké l'id de la fiche de Frais (id_frais). Cet id sera récupéré lors de la validation et nous permettra selon qu'il sera > 0 ou non s'il s'agit de la validation d'une modification ou d'un ajout. Lorsqu'on clique sur le bouton Annuler on appelle un script JavaScript qui utilise la méthode window.location à laquelle on affecte l'url faisant référence à la route /getListeFrais.

Important : le montant validé n'est pas en saisi car ce n'est pas le visiteur qui saisit ce montant mais le comptable qui après avoir fait les vérifications nécessaires saisit ce montant, d'où son nom : montantvalide.

Note : on utilisera systématiquement la structure {{ \$variable or default }}, par exemple : {{ \$objet->nom or '' }} ou {{ \$objet->prix or 0 }}



Récupérer et compléter le formulaire.

La méthode métier updateFrais()

```
/**
 * Mise à jour d'une fiche de Frais
 * @param type $id_frais : id de la fiche de Frais à modifier
 * @param type $anneemois : période de la fiche de Frais à modifier
 * @param type $nbjustificatifs : Nb justificatifs de la fiche de Frais à modifier
 */
public function updateFrais($id_frais, $anneemois, $nbjustificatifs) {
    try {
        $dateJour = date("Y-m-d");
        DB::table('frais')->where('id_frais', '=', $id_frais)
            ->update(['anneemois' => $anneemois, 'nbjustificatifs' => $nbjustificatifs,
                'datemodification' => $dateJour]);
    }
    catch (QueryException $e) {
        throw new MonException($e->getMessage(), 5);
    }
}
```

Pas de difficulté particulière. Comme ce n'est pas le visiteur qui saisit la date de modification de la fiche, celle-ci est générée par la fonction PHP date() avec un format compatible avec celui des date dans MySQL.



Saisir la méthode métier updateFrais() et tester une modification.

Modification d'une fiche de Frais

Période :

Nb justificatifs :

Montant validé : 0.00

Ajouter une fiche de Frais

Pour réaliser cette fonctionnalité, nous aurons besoin des composants suivants :

GSB Frais

Composant	Valeur	Rôle
Routes	/ajouterFrais	Prise en compte du clic sur le bouton Ajouter du menu général
	/validerFrais	Prise en compte du clic sur le bouton Valider du formulaire
Vue	formFrais.blade.php	Formulaire affichant la fiche de Frais sélectionnée et permettant un ajout
Classes métier	Frais	Stockage des données et requêtes
Contrôleur	FraisController	Pilotage de la demande d'ajout et de la validation
Méthode métier	insertFrais()	Insertion de la fiche de Frais dans la Bdd
Méthodes contrôleur	addFrais()	Affichage du formulaire vide pour un ajout
	validateFrais()	Récupération des données saisies et appel de la méthode métier insertFrais()

Les routes

La route validerFrais ne change pas, elle est commune à l'ajout et la modification. La route d'ajout est simple, elle appelle la méthode addFrais() du contrôleur FraisController. Il faudra aussi faire quelque chose dans la page principale master.blade.php !

Le formulaire formFrais

Il est commun à la modification et l'ajout et ne change pas.

La méthode addFrais()

```
/**
 * Initialise le formulaire de Frais pour un ajout
 * @return type Vue formFrais
 */
public function addFrais() {
    try
    {
        $erreur = "";
        $unFrais = new Frais();
        $titreVue = "Ajout d'une fiche de Frais";
        // Affiche le formulaire en lui fournissant les données à afficher
        return view('formFrais', compact('unFrais', 'titreVue', 'erreur'));
    } catch (MonException $e) {
        $monErreur = $e->getMessage();
        return view('vues/pageErreur', compact('monErreur'));
    } catch (Exception $e) {
        $monErreur = $e->getMessage();
        return view('vues/pageErreur', compact('monErreur'));
    }
}
```

La seule chose qui est à noter, est cachée, en effet lorsqu'on fait un `new Frais()`, on invoque le constructeur, or dans ce constructeur, on initialise la propriété `id_frais` à 0 :

```
public function __construct() {  
    $this->id_frais = 0;  
}
```

On se rappelle que dans le formulaire il y a un champ INPUT HIDDEN qui contient cet id, donc quand on va demander un ajout ce champs contiendra la valeur 0 alors que lorsqu'on demandait une modification, ce champ contenait la valeur de l'id de la fiche à modifier (1, 22, 45 ...).

On verra plus loin que cette valeur 0 va nous être très utile.

La méthode validateFrais()

Comme elle est commune à la modification et à l'ajout, elle doit changer pour pouvoir faire la différence entre un update et un insert :

```
/**  
 * Enregistre un ajout ou une modification d'une fiche de Frais  
 * Si id_frais > 0 c'est une modification, sinon  
 * c'est un ajout. Id_frais est dans un INPUT HIDDEN  
 * @return type route /getListeFrais  
 */  
public function validateFrais() {  
    try  
    {  
        $id_frais = Request::input('id_frais');  
        $anneemois = Request::input('anneemois');  
        $nbjustificatifs = Request::input('nbjustificatifs');  
        $unFrais = new Frais();  
        if ($id_frais > 0) {  
            $unFrais->updateFrais($id_frais, $anneemois, $nbjustificatifs);  
        } else {  
            $id_visiteur = Session::get('id');  
            $unFrais->insertFrais($anneemois, $nbjustificatifs, $id_visiteur);  
        }  
        // Retourne à la liste des des Frais  
        return redirect('/getListeFrais');  
    } catch (MonException $e) {  
        $monErreur = $e->getMessage();  
        return view('vues/pageErreur', compact('monErreur'));  
    } catch (Exception $e) {  
        $monErreur = $e->getMessage();  
        return view('vues/pageErreur', compact('monErreur'));  
    }  
}
```

Si l'`id_frais` est égal à 0, c'est qu'il s'agit d'un ajout, sinon c'est qu'il s'agit d'une modification. Dans le cas d'un ajout il faut récupérer dans la Session l'id du visiteur à qui appartient la fiche de Frais que l'on vient de saisir.



Effectuer l'ensemble des modifications nécessaires et tester l'ajout d'une fiche de frais.

Supprimer une fiche de Frais

La suppression d'une fiche en soi n'a rien de difficile, si ce n'est que l'on a ici affaire à une contrainte d'intégrité référentielle entre la table frais et la table fraishorsforfait :

```
alter table fraishorsforfait
  add constraint fk_fraishorsforfait_frais foreign key (id_frais)
  references frais (id_frais);
```

Cela implique que si l'on tente de supprimer une fiche de Frais qui a des Frais Hors Forfait qui lui sont rattachés, la base de données va générer une erreur fatale et "planter" notre application. Nous allons d'abord mettre en évidence ce "plantage", puis ensuite voir comment y remédier.

Les données

Dans un premier temps comme nous n'avons pas développé la saisie des frais hors forfait, nous allons rajouter manuellement des données dans la table fraishorsforfait.



Saisir deux ou trois lignes dans la table fraishorsforfait en les rattachant à des fiche de Frais (id_frais) existentes.

GSB Frais

Pour réaliser la suppression d'une fiche de Frais, nous aurons besoin des composants suivants :

Composant	Valeur	Rôle
Route	/supprimerFrais	Prise en compte du clic sur le bouton Supprimer de la liste de Frais
Vue	listeFrais.blade.php	Liste des Frais du visiteur connecté
Classes métier	Frais	Stockage des données et requêtes
Contrôleur	FraisController	Pilotage de la demande de suppression
Méthode métier	deleteFrais()	Suppression de la fiche de Frais dans la Bdd
Méthode contrôleur	supprimeFrais()	Appel à la méthode métier deleteFrais()

La route

La route supprimerFrais admet le paramètre id_frais et appelle la méthode supprimeFrais du contrôleur.

La vue listeFrais

Elle ne change pas, c'est le script JavaScript qui appellera la route /supprimerFrais.

La méthode métier deleteFrais

```
/**
 * Suppression d'une fiche de Frais
 * @param type $id_frais : Id de la fiche de frais à supprimer
 */
public function deleteFrais($id_frais) {
    try
    {
        DB::table('frais')->where('id_frais', '=', $id_frais)->delete();
    }
    catch (QueryException $e) {
        throw new MonException($e->getMessage(), 5);
    }
}
```

La méthode supprimeFrais

C'est elle la plus importante, mais pour le moment elle reste banale :

```
public function deleteFrais($id_frais) {
    $unFrais = new Frais();
    return redirect('/getListeFrais');
}
```



Effectuer les modifications nécessaires et tenter la suppression d'une fiche de Frais ayant des Frais Hors Forfait. Vous devriez obtenir quelque chose de proche de la copie d'écran ci-dessous :

Whoops, looks like something went wrong.

2/2 QueryException in Connection.php line 729:

```
SQLSTATE[23000]: Integrity constraint violation: 1451 Cannot delete or update a parent row: a foreign key constraint fails (`gsb`.`fraishorsforfait`, CONSTRAINT `fk_fraishorsforfait_frais` FOREIGN KEY (`id_frais`) REFERENCES `frais` (`id_frais`)) (SQL: delete from `frais` where `id_frais` = 1)
```

1. in Connection.php line 729
2. at Connection->runQueryCallback('delete from `frais` where `id_frais` = ?', array('1'), object(Closure)) in Connection.php line 685
3. at Connection->run('delete from `frais` where `id_frais` = ?', array('1'), object(Closure)) in Connection.php line 483
4. at Connection->affectingStatement('delete from `frais` where `id_frais` = ?', array('1')) in Connection.php line 438

Notre application est bel et bien plantée, en ce sens qu'elle n'est plus active, l'utilisateur ne peut plus l'utiliser !

Et ça c'est inacceptable, n'oubliez jamais en tant que futur développeur que l'on pourra à la grande rigueur vous "pardonner" qu'une application ne fasse pas exactement ce qu'elle était censée faire, mais on ne vous "pardonnera" JAMAIS qu'elle "plante" !

Une solution

Globalement la solution consiste à "trapper" l'erreur fatale à l'aide d'un gestionnaire d'erreurs. Le framework Laravel met à disposition toute une panoplie d'outils très performants mais aussi assez complexe à mettre en œuvre. Aussi nous allons utiliser une solution à la fois simple et commune à tous les langages, les Exceptions. Le principe est relativement simple on utilise la structure try .. catch ... finally qui permet de lancer des opérations à risque et de capturer l'éventuelle erreur fatale empêchant ainsi l'application de "planter". Voici le nouveau code de la méthode `supprimeFrais()` :

```
/**
 * Supression d'une fiche de frais sur son Id
 * Si la fiche de Frais a des lignes, cela produira une erreur
 * à cause de la contrainte d'intégrité référentielle, on
 * stockera dans la Session le message d'erreur qui sera
 * repris et affiché lors du réaffichage de la liste
 * @param type $id_frais Id de la fiche de Frais à supprimer
 * @return type Vue getListeFrais ou getErrors
 */
public function supprimeFrais($id_frais) {
    $unFrais = new Frais();
    try {
        $unFrais->deleteFrais($id_frais);
    } catch (Exception $ex) {
        Session::put('erreur', $ex->getMessage());
    } finally {
        return redirect('/getListeFrais');
    }
}
```

On tente d'appeler la méthode métier deleteFrais() si tout se passe bien on passe par le bloc finally qui demande le réaffichage de la liste des Frais, par contre si une erreur fatale se produit, on rentre dans le bloc catch dans lequel on place le message d'erreur produit par l'Exception \$ex dans une variable de Session, puis comme précédemment on passe par le bloc finally qui fera réafficher la liste des Frais.

Il nous faut maintenant modifier la méthode getFraisVisiteur() pour tenir compte de la possibilité d'une erreur fatale :


```
/**
 * Affiche la liste de toutes les fiches
 * de Frais du visiteur connecté
 * Si une erreur a été stockée dans la Session
 * on la récupère, on l'efface de la Session
 * et la passe au formulaire
 * @return type Vue listeFrais
 */
public function getFraisVisiteur() {
    try
    {
        $erreur = Session::get('erreur');
        Session::forget('erreur');
        $unFrais = new Frais();
        $id_visiteur = Session::get('id');
        // On récupère la liste de tous les frais
        $mesFrais = $unFrais->getFrais($id_visiteur);
        // On affiche la liste de ces frais
        return view('listeFrais', compact('mesFrais', 'erreur'));

        } catch (MonException $e) {
            $monErreur = $e->getMessage();
            return view('vues/pageErreur', compact('monErreur'));
        } catch (Exception $e) {
            $monErreur = $e->getMessage();
            return view('vues/pageErreur', compact('monErreur'));
        }
    }
}
```

La méthode de classe `Session::get('erreur')` récupère le contenu de la variable de session 'erreur' si celle-ci est vide ou n'existe pas la variable `$erreur` contiendra une chaîne vide ou sera nulle sinon elle contiendra le message d'erreur. Une fois que l'on a récupéré l'erreur, il faut effacer la variable de session erreur, pour ne pas réafficher cette erreur éternellement, ce qui est fait avec la méthode `forget()`.

La vue `error.blade.php` qui est incluse dans la vue `listeFrais.blade.php` teste le contenu de la variable `$erreur` et l'affiche s'il n'est pas vide.









Important : pour que cela fonctionne, il faut faire référence au package PHP Exception, donc en haut de la classe `FraisController`, on ajoutera la ligne :

```
use Request;
use Illuminate\Support\Facades\Session;
use App\metier\Frais;
use Exception;
```



Effectuer les modifications nécessaires et tester une suppression.

Liste des frais

Période	Modifier	Supprimer
201607		
201608		
201609		
201610		

! SQLSTATE[23000]: Integrity constraint violation: 1451 Cannot delete or update a parent row: a foreign key constraint fails (`gsb`.`fraishorsforfait`, CONSTRAINT `fk_fraishorsforfait_frais` FOREIGN KEY (`id_frais`) REFERENCES `frais` (`id_frais`)) (SQL: delete from `frais` where `id_frais` = 1)









Evidemment dans une application aboutie, on n'afficherait pas le message d'erreur de la base de données, mais un message plus compréhensible par l'utilisateur. Il faudrait analyser le contenu du message et si l'on trouve tel ou tel expression, on produirait tel ou tel message. C'est faisable, mais nous emmènerait trop loin ici, nous on se contentera d'afficher le message suivant :

Impossible de supprimer une fiche ayant des Frais Hors Forfait !



Effectuer la modification et tenter une suppression.

Liste des frais

Période	Modifier	Supprimer
201607		
201608		
201609		
201610		

! Impossible de supprimer une fiche ayant des Frais Hors Forfait !

Les Frais Hors Forfait

Les visiteurs ont droits à deux types de frais : les Frais forfaitaires, ils ont un tarif fixé à l'avance (kms, repas, hôtel ...) et les frais dont le coût ne peut être déterminé à l'avance (location d'une salle de conférence, prestation d'un intervenant ...), ce sont les frais hors forfait.

Pour le moment nous ne gèrerons que les frais hors forfait. La liste des Frais Hors Forfait est accessible par le bouton Frais Hors Forfait du formulaire formFrais.

The screenshot shows a web form titled "Modification d'une fiche de Frais". It contains three input fields: "Période" with the value "201607", "Nb justificatifs" with the value "4" and a small up/down arrow icon, and "Montant validé" with the value "0.00". Below these fields are three buttons: a blue button with a checkmark and the text "Valider", a blue button with an 'X' and the text "Annuler", and a blue button with a list icon and the text "Frais hors forfait".

On a cependant un petit problème ce bouton est aussi présent sur le formulaire lors d'un Ajout d'une fiche de Frais, or on ne peut pas ajouter des frais hors forfait à une fiche de Frais qui n'existe pas !

Il faut donc faire disparaître ce bouton (en fait ne pas faire apparaître) lorsque le formulaire formFrais est utilisé pour un ajout.



Effectuez la modification demandée.

Deux indices : le champ INPUT HIDDEN id_frais peut vous être très utile et la vue error fournit un bon exemple.

Lister les Frais Hors Forfait

Produire la liste des frais hors forfait est très proche de ce que nous avons fait pour la liste des Frais. Vous disposez donc de l'ensemble des ressources (la vue est dans les Ressources) et du savoir-faire. Pour autant il ne faut pas vous lancer sans quelques préparatifs, voici quelques conseils :

- Utilisez le tableau des composants nécessaires que nous avons systématiquement utilisé pour effectuer nos développements. Ce sera un guide précieux, il vous permettra d'abord de réfléchir à ce dont vous aurez besoin pour réaliser votre objectif, puis vous servira de "check list" au fur et à mesure de l'avancée des travaux.

Composant	Valeur	Rôle
Route		
Vue		
...

- Connectez-vous au site exemple : <http://gsb/gsb-frais>, vous aurez une idée plus précise de ce qu'il faut faire.

- La liste des frais hors forfait est celle d'une fiche de frais, il vous faudra donc gérer l'id_frais de la même manière que pour les Frais. De plus chaque fois que vous ferez une redirection pour réafficher cette liste, il faudra utiliser cet id, exemple :

```
return redirect('/getListeProducts/'.$id_product);
```

- Consultez bien le script de création de la table fraishorsforfait avant de réaliser la classe FraisHorsForfait, vous y remarquerez qu'il y a une clé étrangère id_frais qui pointe vers la clé primaire id_frais de la fiche de Frais.
- Le calcul du montant total des frais hors forfait se fait dans le contrôleur, en utilisant une boucle foreach et on passe la valeur par une variable à la vue.
- Le bouton Annuler appellera la route /modifierFrais.
- Le lien qui permettra de supprimer un FHF devra passer deux paramètres, voici un exemple :

```
<a href="{{ url('/supprimeLigneCde') }}" />{{ $uneLigne->id_commande }} / {{ $uneLigne->id_article }}</a>
```



Réalisez votre tableau des composants, montrez-le à l'un des enseignants qui vous donnera ou non le feu vert pour démarrer la programmation.

Ajouter un Frais Hors Forfait

Même démarche que pour la liste.

Conseils :

- Comme pour la liste, l'ajout nécessite de conserver mémoire de l'id de la fiche de Frais dont dépend le FHF (Frais Hors Forfait), cet id sera donc à passer en paramètre sur le lien du bouton Ajout.

Modifier un Frais Hors Forfait

Même démarche que pour la liste.

Conseils :

- Même conseil que pour l'ajout.

Supprimer un Frais Hors Forfait

Même démarche que pour la liste.

Conseils :

- Supprimer un Frais Hors Forfait ne pose pas le même genre de problème qu'une fiche de Frais, mais vous devrez quand même utiliser le gestionnaire d'Exception car c'est une bonne habitude à prendre et vous en aurez besoin pour les CCF.