7.3.4.2 Réalisation des étapes du scénario

La maquette de la fiche de frais doit être analysée afin de déterminer les vues nécessaires au déroulement du scénario de validation. Comme toujours en programmation, il est préférable de créer puis d'assembler des éléments simples (ici des vues) plutôt que de créer de gros éléments complexes.

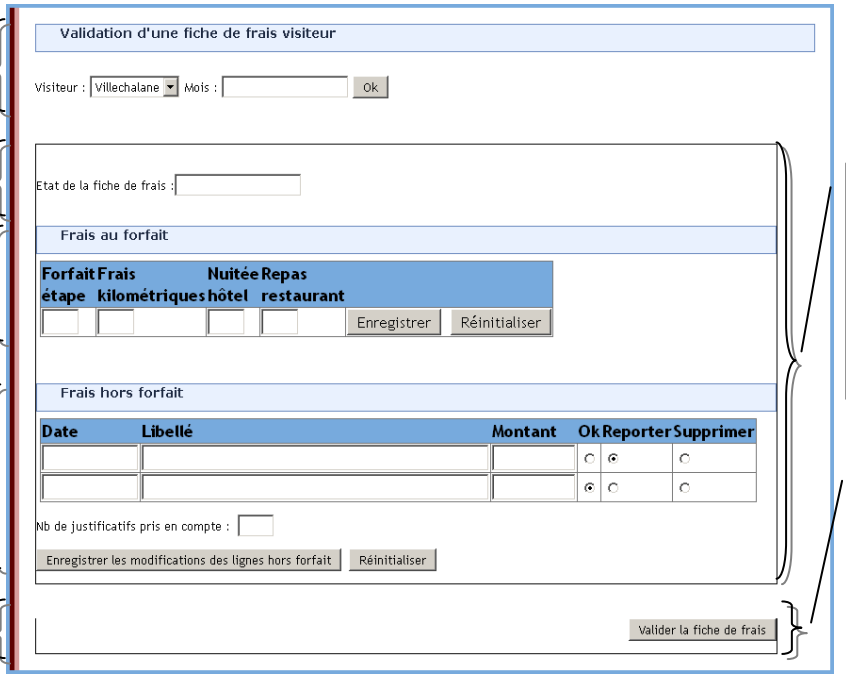
Vue n° 1 : le formulaire qui permet de choisir le visiteur dont on veut traiter la fiche.

Vue n° 2 : affichage de l'état de la fiche.

Vue n° 3 : affichage/modification des frais forfaitisés.

Vue n° 4 : affichage/modification des frais hors forfait.

Vue n° 5 : validation de la fiche.



Les 3 premières parties de la fiche de frais sont encadrées. Dans le code HTML une division est créée à cet effet.

La dernière partie de la fiche de frais est aussi contenue dans une division.

On s'intéresse ici aux 3 premières étapes du scénario.

7.3.4.2.1 Etape du choix d'un visiteur et affichage de la fiche de frais

Créez un nouveau contrôleur nommé `c_valideFrais.php`.

Modifiez le **contrôleur principal** de manière à inclure le contrôleur `c_valideFrais.php` : le cas d'utilisation à traiter se nomme `validerFicheFrais`.

Dans le contrôleur `c_valideFrais.php` ajoutez le code pour gérer le cas où `action` est égal à `choixInitialVisiteur` (vous pouvez copier-coller le code d'un autre contrôleur et le modifier).

7.3.4.2.2 Développement de la vue du choix d'un visiteur

Copiez et renommez le fichier `formValidFrais.htm` en `v_valideFraisChoixVisiteur.php`. Vous êtes en train de créer la vue n°1.

Il faut supprimer tout le code inutile et ajouter le remplissage de la liste déroulante des visiteurs et l'affichage du mois.

Remarque : si dans le cadre d'un projet antérieur vous aviez développé une bibliothèque de fonctions, ce serait une bonne idée de l'utiliser dans ce projet ...

C'est le contrôleur qui doit fournir les données. Il utilisera la première méthode de la classe `PdoGsb` que vous aviez développée pour appeler la procédure stockée qui liste les visiteurs.
Pour obtenir le mois concerné par la validation (sous la forme aaaamm) créez une fonction dans votre bibliothèque avec le code suivant :

```
    $laDate = new DateTime();
    $leMois = (int) ($laDate->format('m')) - 1;
    $lAnnee = (int) ($laDate->format('Y'));

    if ($leMois == 0) {
        $lAnnee--;
        $leMois = 12;
    }

    return (new DateTime((string) $lAnnee . '-' . (string) $leMois))->format('Ym');
```

Dans le contrôleur il faudra donc appeler pour `choixInitialVisiteur` les vues qui affichent :
- le menu de l'application,
- le formulaire de choix d'un visiteur.

Question : pourquoi ne pas appeler aussi les vues qui affiche l'entête et le pied de page dans le contrôleur `c_valideFrais.php` ?
Réponse : cherchez la solution du côté du contrôleur principal …

Testez l'application : en cliquant sur l'option de menu "Valider des fiches de frais" vous devriez obtenir le formulaire permettant de sélectionner un visiteur.

Passons à l'étape 4 du scénario.

7.3.4.2.3 Développement des autres vues composant une fiche de frais

L'utilisateur sélectionne un visiteur et soumet le formulaire. Il faut alors ré-afficher le formulaire et la fiche de frais complète :
- ajoutez à votre contrôleur l'action nommée `afficherFicheFraisSelectionnee`.

- modifiez l'identifiant du visiteur et le mois de la fiche de frais qui sont stockés dans les variables de session avec les valeurs qui ont été sélectionnées.

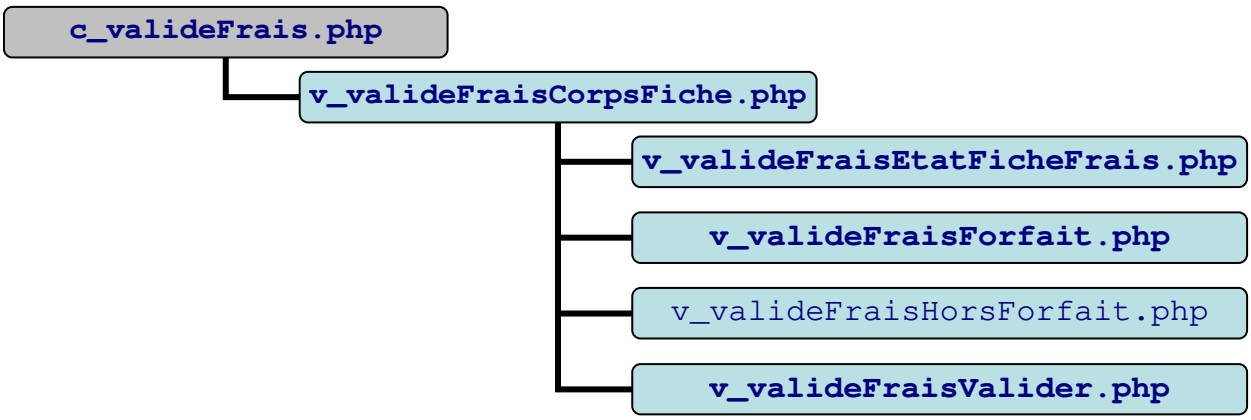
Remarque
Dans l'architecture MVC, c'est normalement le modèle qui s'occupe des variables de session. Mais ici le développement s'est effectué à partir d'un existant perfectible …

- Il faut ensuite instancier la fiche de frais qui a été sélectionnée.

- Il faut créer les vues 2, 3 et 5 qui composent la fiche de frais (la vue n° 4 des frais hors forfait sera créée ultérieurement).

Pour chaque vue procédez comme à l'étape précédente :
- ajoutez au contrôleur les instructions destinées à fournir les données à la vue. Pour les 3 vues, le contrôleur va devoir faire appel aux méthodes suivantes à développer dans la classe `FicheFrais` :
  - `getLibelleEtat()` qui retourne le libellé de l'état de la fiche ;
  - `getNbJustificatifs()` qui retourne le nombre de justificatifs ;
  - `getLesQuantitesDeFraisForfaitises()` qui retourne un tableau contenant les quantités pour chaque ligne de frais forfaitisé de la fiche de frais.
- Copiez et renommez le fichier `formValidFrais.htm` , puis supprimez le code inutile et ajoutez le code manquant à la vue.

- Pour plus de simplicité les 3 vues seront appelées par une vue unique nommée `v_valideFraisCorpsFiche.php` (voir le schéma ci-dessous). C'est cette vue que le contrôleur appellera pour afficher une fiche de frais. Créez cette vue.



- Testez votre application.

Remarque
Les deux cas d'utilisation et le schéma du cycle de traitement d'une fiche indiquent que seule une fiche dans l'état "clôturé" peut être validée. Ce point sera traité ultérieurement.

Passons aux étapes 5 et 6 du scénario.

7.3.4.2.4 Enregistrement des modifications apportées aux frais forfaitisés

Lorsqu'un formulaire est soumis, ses informations sont contrôlées, puis si aucune erreur n'est détectée elles peuvent être traitées. Le traitement peut consister à les enregistrer dans la source de données.

Vous allez procéder ainsi pour les frais forfaitisés, en utilisant la classe métier `FicheFrais`.

La fiche de frais doit d'abord être instanciée et initialisée sans frais forfaitisés. Ceux-ci seront ensuite ajoutés à partir des valeurs saisies par l'utilisateur. Ils pourront alors être contrôlés et serviront à mettre à jour la base de données.

- Pour cela il faut d'abord implémenter la méthode `ajouterUnFraisForfaitise()` de la classe `FicheFrais`. Pour initialiser la propriété `numFrais` (le n° de frais) la stratégie suivante a été retenue :

  une fiche de frais comporte systématiquement les 4 catégories de frais forfaitisés. Ainsi dans la base de données chaque fiche de frais comporte 4 lignes de frais forfaitisés qui seront numérotées de 1 à 4. Chacun de ces 4 numéros correspond à une catégorie de frais forfaitisé. Consultez dans la classe `FicheFrais` la propriété `tabNumLigneFraisForfaitise`.

- Implémentez la méthode privée `getNumLigneFraisForfaitise()` déjà déclarée et documentée dans la classe `FicheFrais`. Cette méthode utilisera bien sur la propriété `tabNumLigneFraisForfaitise`.

- Implémentez la méthode `ajouterUnFraisForfaitise()` déjà déclarée et documentée dans la classe `FicheFrais`.

- Ajoutez à votre contrôleur l'action nommée `enregModifFF`. Instanciez la fiche de frais et initialisez la sans les frais forfaitisés, puis ajoutez lui ensuite les frais forfaitisés saisis par l'utilisateur.

Le contrôle des informations saisies

Avant de mettre à jour les frais forfaitisés dans la base de données, il faut contrôler que les quantités saisies sont bien des entiers positifs.
Dans la bibliothèques `fct.inc.php` c'est le rôle de la fonction `lesQteFraisValides()`.

Dans l'architecture MVC le contrôle de données fait partie des classes métier. Ici c'est dans la classe `FicheFrais` qu'il devra être implémenté.

La méthode `controlerQtesFraisForfaitises()` est chargée de contrôler les quantités saisies. Pour cela elle doit utiliser la fonction `lesQteFraisValides()` déjà implémentée dans la bibliothèque et la méthode `getLesQuantitesDeFraisForfaitises()` déjà déclarée et documentée dans la classe `FicheFrais`.

- Implémentez ces deux méthodes.

Rappel
Le contrôle des valeurs saisies dans le formulaire peut aussi être implémenté en JavaScript.
Mais, il ne remplace pas celui effectué en PHP côté serveur http, car le JavaScript peut être désactivé dans le navigateur.

L'enregistrement des frais forfaitisés

L'opération de mise à jour des lignes de frais forfaitisés dans la base de données doit être réalisée en veillant à ce que l'intégrité des données soit maintenue, même en cas d'arrêt intempestif du SGBD.
Les mises à jour des lignes de frais forfaitisés doivent donc être traitées comme un tout.

Une solution : mettre en œuvre une transaction dans laquelle toutes les opérations de mise à jour seront effectués. En cas d'erreur la transaction sera annulée. Dans le cas contraire elle sera validée.

- Commencez par créer la procédure stockée nommée `SP_LIGNE_FF_MAJ` qui met à jour la quantité d'une ligne de frais forfaitisé. C'est cette procédure qui sera appelée dans la transaction.

- Dans la classe `pdogsb` implémentez la méthode `setLesQuantitesFraisForfaitises()` qui est déjà déclarée et documentée. Cette méthode doit d'abord :
  - préparer l'exécution de la procédure stockée (voir la documentation de la methode `prepare()` de PDO).
  - Lier les paramètres à passer à la procédure stockée (voir `bindParam()` et `bindValue()`).

  - Dans un bloc `try{}` la transaction sera ensuite démarrée. Elle exécutera la procédure stockée autant de fois que nécessaire. Un `commit` sera ensuite exécuté.

  - Le bloc `catch{}` correspondant au `try` permettra de gérer les exceptions éventuelles : il utilisera le système de messages d'erreur de l'application pour décrire l'erreur (c'est le programme appelant qui se chargera d'afficher le message d'erreur), et annulera (`rollback`) la transaction.

Pour finir :
- implémentez la méthode `mettreAJourLesFraisForfaitises()` de la classe `FraisForfait` qui fait appel à `setLesQuantitesFraisForfaitises()`. Elle est déjà déclarée et documentée.

- Appelez cette méthode dans votre contrôleur lorsque les frais forfaitisés ne comportent pas d'erreur.

- Affichez un message indiquant que la mise à jour a été effectuée, ou affichez le message d'erreur créé lorsqu'une exception est apparue pendant la mise à jour.

Testez votre application. Peut-être que la fiche de frais que vous testez n'est pas dans l'état "Saisie clôturée". Nous résoudrons ce problème très bientôt …

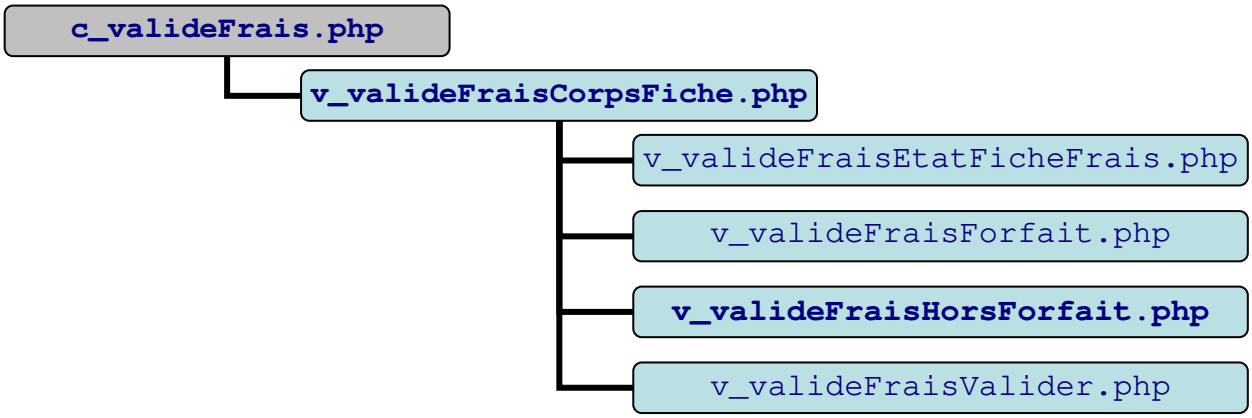
Traitement des frais hors forfait. Revenons à l'étape 4 du scénario.

7.3.4.2.5 Développement de la vue des frais hors forfait - Affichage

Il faut compléter la fiche de frais avec les frais hors forfait. Procédez avec la vue n°4 (voir page 18) comme pour les autres vues :

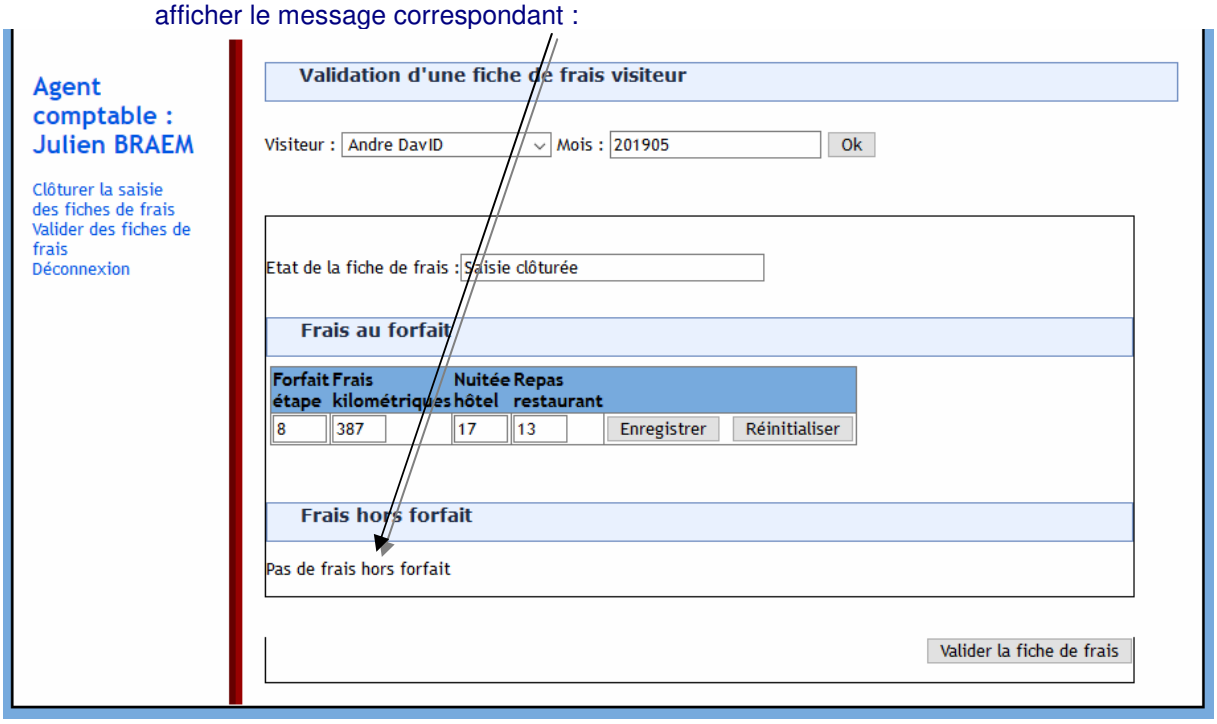
- ajoutez au contrôleur les instructions destinées à fournir les données à la vue. Pour les lignes de frais le contrôleur va devoir faire appel à la méthode `getLesInfosFraisHorsForfait()` à développer dans la classe `FicheFrais` (elle est déjà documentée).
- Copiez et renommez le fichier `formValidFrais.htm`, puis supprimez le code inutile et ajoutez le code manquant à la vue.

- Complétez la vue `v_valideFraisCorpsFiche.php`. Elle doit appeler la vue des frais hors forfait qui manquait jusqu'à présent :



D'après la modélisation, le nombre de lignes de frais hors forfait est variable d'une fiche à l'autre. A vous de voir comment regrouper au mieux les contrôles de saisie.

Attention ! Il faut aussi tenir compte du cas où il n'y a aucun frais hors forfait pour la fiche de frais. La vue devra tester le nombre d'éléments du tableau qui lui est fourni, et s'il n'y en a aucun afficher le message correspondant :



Passons aux étapes 7, 8, et 7.a, 8.a, 8.b du scénario.

7.3.4.2.6 Enregistrement des modifications apportées aux frais hors forfait

Comme pour les frais forfaitisés le contrôle et l'enregistrement des frais hors forfait doit s'effectuer via la classe métier `FicheFrais`.

Par contre, la mise à jour des frais hors forfait tient en une action pour chaque ligne : un report ou une suppression logique (ou une acceptation qui revient à laisser le frais en l'état).

La fiche de frais doit d'abord être instanciée et initialisée sans frais hors forfait. Ceux-ci seront ensuite ajoutés à partir des valeurs soumises par l'utilisateur. L'action choisie par l'utilisateur pour chaque ligne devra être intégrée à chaque objet frais. Le nombre de justificatifs pris en compte sera corrigé avec la valeur saisie par l'utilisateur. Il pourra alors être contrôlé, et toutes ces informations serviront à mettre à jour la base de données.

Dans la classe `FraisHorsForfait`:
- Modifiez le constructeur : il doit aussi initialiser la propriété `action`. Le nouveau paramètre doit avoir la valeur '`O`' par défaut (il peut prendre une valeur parmi O, R ou S). Ainsi le constructeur pourra être appelé sans ou avec ce paramètre, comme s'il avait plusieurs signatures (cf. le manuel de PHP : `https://www.php.net/manual/fr/functions.arguments.php#functions.arguments.default`).
- Implémentez l'accesseur `getAction()`.
- Modifiez la méthode `getLesInfosFraisHorsForfait()` : la propriété `action` doit être intégrée aux données retournées. Profitez-en pour mettre à jour la documentation de cette méthode.

Ensuite implémentez dans la classe `FicheFrais` :
- la méthode `ajouterUnFraisHorsForfait()` déjà déclarée et documentée.
- La méthode `setNbJustificatifs()`.
- La méthode `initAvecInfosBDDSansFHF()`, qui sera appelée après le constructeur pour poursuivre l'initialisation :
  - de l'état de la fiche (identifiant et libellé) et du nombre de justificatifs ;
  - des frais forfaitisés ;
  - mais pas des frais hors forfait.

Tester votre application, elle devrait continuer de fonctionner sans problème car les modifications que vous avez apportées aux méthodes existantes leur ont ajouté de nouvelles siignatures, mais sans modifier les signatures existantes.

- Ajoutez à votre contrôleur l'action nommée `enregModifFHF`. Instanciez la fiche de frais et initialisez-la sans les frais hors forfait, puis ajoutez lui ensuite les frais hors forfait saisis par l'utilisateur.

- Implémentez la méthode `controlerNbJustificatifs()` de la classe `FicheFrais` : elle doit contrôler si le nombre de justificatifs est bien un entier positif et retourner un booléen …

Etapes 7, 8 et 8.b du scénario

Remarques

L'enregistrement des modification apportée aux frais hors forfait implique 3 opérations :
- la suppression d'une ligne de frais. Elle est répétée autant de fois que de lignes à supprimer.
- Le report d'une ligne de frais. Elle est répétée autant de fois que de lignes à reporter.
- la mise à jour du nombre de justificatifs pris en compte. Elle est effectuée une fois.

Toutes ces opérations doivent former un tout afin que l'intégrité des données soit toujours respectée. Donc, lorsqu'une fiche de frais est modifiée, soit toutes ces opérations sont inscrites dans la base de données, soit aucune ne l'est. Un état intermédiaire n'est pas envisageable.

Comme pour les frais forfaitisés, ici aussi la solution est de mettre en œuvre une transaction qui sera pilotée par le client.

7.3.4.2.7 La suppression des frais hors forfait

- Créez la procédure stockée (`SP_LIGNE_FHF_SUPPRIME`) qui supprime une ligne de frais hors forfait selon les modalités décrites à l'étape 8. Veillez à ce qu'une ligne ne puisse pas être supprimée plusieurs fois. Implémentez ensuite la méthode du modèle qui appelle cette procédure.

- Dans la classe `pdogsb` implémentez la méthode `setLesFraisHorsForfait()` qui est déjà déclarée et documentée. Cette méthode doit d'abord :
  - préparer l'exécution de la procédure stockée (voir la documentation de la methode `prepare()` de PDO).
  - Lier les paramètres à passer à la procédure stockée (voir `bindParam()` et `bindValue()`).

  - Dans un bloc `try{}` la transaction sera ensuite démarrée. Elle exécutera la procédure stockée autant de fois que nécessaire. Un `commit` sera ensuite exécuté.

  - Le bloc `catch{}` correspondant au `try` permettra de gérer les exceptions éventuelles : il utilisera le système de messages d'erreur de l'application pour décrire l'erreur (c'est le programme appelant qui se chargera d'afficher le message d'erreur), et annulera (`rollback`) la transaction.

- Implémentez dans la classe `FicheFrais` la méthode `mettreAJourLesFraisHorsForfait()` qui fait appel à `setLesFraisHorsForfait()`. Pour l'instant il s'agit uniquement des suppressions. Les autres opérations seront traités à l'étape suivante.

- Complétez le contrôleur : si le nombre de justificatifs est correct on peut enregistrer dans la base de données les suppressions demandées par l'utilisateur. Si le nombre de justificatifs n'est pas correct, utilisez le système de message d'erreur de l'application pour informer l'utilisateur, et réaffichez la fiche de frais.

Etapes 7.a et 8.a du scénario.

7.3.4.2.8 Le report des frais hors forfait

Le report d'une ligne de frais hors forfait implique le calcul d'un nouveau numéro de ligne.

- Commencez par créer une fonction stockée nommée `F_NUM_NOUVELLE_LIGNE_FRAIS()` qui retourne le prochain numéro à utiliser pour créer un frais (ou une ligne de frais) pour la fiche dont l'identifiant visiteur et le mois sont passés en paramètre. Si aucune ligne de frais n'a encore été créée, 1 sera retourné. Si nécessaire consultez la documentation de `create function`.

- Créez ensuite la procédure stockée nommée `SP_LIGNE_FHF_REPORTE` qui reporte une ligne de frais selon les modalités décrites à l'étape 8.a du scénario.

- Dans la classe `pdogsb` complétez la méthode `setLesFraisHorsForfait()` : préparation et exécution de la procédure stockée de report des frais, afin que la méthode enregistre aussi les reports …

7.3.4.2.9 La mise à jour du nombre de justificatifs pris en compte

Bien que non décrite dans le scénario, cette dernière opération fait partie de la mise à jour des frais hors forfait (Cf. 2.1 Campagne de validation page 2 le paragraphe intitulé "Pour les frais hors forfait").

- La procédure stockée a créer s'appelle `SP_FICHE_NB_JPEC_MAJ`. Elle met à jour le nombre de justificatifs uniquement si celui-ci a été modifié.

- Implémentez la méthode de la classe `pdogsb` qui l'appelle.

- Complétez la méthode `setLesFraisHorsForfait()` de la classe `pdogsb` avec la mise à jour du nombre de justificatifs.

Testez votre application.

Etapes 9 et 10 du scénario

7.3.4.2.10 Développement de la vue de validation de la fiche de frais

Une fois que le contenu de la fiche de frais est satisfaisant, l'agent comptable peut valider la fiche.

Comme indiqué à l'étape 10 du cas d'utilisation, la validation consiste à mettre à jour la date de modification et l'état de la fiche. Mais il ne faut pas omettre la mise à jour du montant validé, dont le calcul doit être implémenté …

- Dans la classe `Frais` décommentez la ligne `abstract public function getMontant();` située à la fin de la définition de cette classe.

- Vous devez à présent implémenter 2 méthodes. Vous avez toutes les informations nécessaires pour cela. A vous de jouer …

- Implémentez ensuite dans la classe `FicheFrais` la méthode `calculerLeMontantValide()` qui calcule le montant total (des frais) validé. Utilisez la fonction `array_merge()` pour vous simplifier la vie ...

- Implémentez la méthode `valider()` de la classe `FicheFrais`. bien entendu il faut d'abord développer la procédure stockée `SP_FICHE_VALIDE` et la méthode correspondante dans la classe `PdoGsb`. C'est la procédure stockée déterminera la date de modification de la fiche (la date du jour donc).

- Compléter la vue `v_valideFraisValider.php`. Complétez uniquement le formulaire HTML de validation de la fiche. Ici l'attribut `action` de la balise "`<form>`" vaudra "`validerFicheFrais`".

- Complétez le controleur `c_valideFrais.php`. Il faut traiter l'action "`validerFicheFrais`". Une fois la fiche validée elle est réaffichée, mais il est toujours possible de la modifier. Nous corrigerons cela ultérieurement.

Testez votre application.

Les finitions

Les finitions suivantes seront réalisées grâce à des traitements côté client, donc développées en JavaScript.

Le fait de cliquer sur le bouton "Valider la fiche de frais" devrait entrainer préalablement à la validation de la fiche de frais :
- soit l'affichage d'un message d'erreur indiquant que des modifications des lignes de frais n'ont pas encore été enregistrées, si cela est le cas ;
- soit l'affichage d'un message demandant confirmation.

Travail à faire
- Réalisez tous les ajouts et modifications nécessaires pour implémenter cette fonctionnalité côté client.
- L'affichage d'un message de confirmation est aussi nécessaire pour l'enregistrement des frais forfaitisés et hors forfait. On doit aussi éviter d'enregistrer les données qui n'ont pas été modifiées. Procédez comme pour l'enregistrement de la validation d'une fiche de frais.