16/11/2016



Documentation Application GSB FRAIS

Laetitia MONSERRAT / Victor DIANA / ROMAIN STEFANELLI

ACTIVITES DU REFERENTIEL MISES EN OEUVRE

A1.1.1 Analyse du cahier des charges d'un service à produire

A1.4.1 Participation à un projet

A4.1.2 Conception ou adaptation de l'interface utilisateur d'une solution applicative

A4.1.7 Développement, utilisation ou adaptation de composants logiciels

A4.1.9 Rédaction d'une documentation technique

A5.1.1 Mise en place d'une gestion de configuration

Sommaire

[I/ Contexte de la situation professionnelle 3](#_Toc467069813)

[A/ Le besoin et les objectifs 3](#_Toc467069814)

[B/ L’existant 3](#_Toc467069815)

[II/ Environnement de développement 3](#_Toc467069816)

[Langages : 3](#_Toc467069817)

[Programmes 3](#_Toc467069818)

[Nature du projet : 3](#_Toc467069819)

[Méthodologie choisie : 3](#_Toc467069820)

[III/ Modélisation fonctionnelle 3](#_Toc467069821)

[Méthodologie mise en œuvre pour l’implémentation du code : 3](#_Toc467069822)

[Cas d’utilisation du comptable : 4](#_Toc467069823)

[IV/ IHM 4](#_Toc467069824)

[V/ Architecture applicative 4](#_Toc467069825)

[VI/ Persistance des données 4](#_Toc467069826)

[A/ Schéma de données (MCD) 4](#_Toc467069827)

[B/ Extrait du script SQL de création des tables (contraintes d’intégrité référentielle) 4](#_Toc467069828)

[VII/ Accès aux données 4](#_Toc467069829)

[A/ PDO 4](#_Toc467069830)

[B/ Design Pattern Singleton 5](#_Toc467069831)

[VIII/ Déploiement, mise en production de l’application 5](#_Toc467069832)

[IX/ Tests fonctionnels 5](#_Toc467069833)

[X/ Difficultés rencontrées 5](#_Toc467069834)

[XI/ Améliorations possibles 5](#_Toc467069835)

[XII/ Conclusion 5](#_Toc467069836)

# I/ Contexte de la situation professionnelle

## A/ Le besoin et les objectifs

## B/ L’existant

# II/ Environnement de développement

Pour notre application nous avons utilisé :

## Langages :

- CSS  
- HTML  
- PHP

Programmes :  
- GitHub pour créer un dépôt distant disponible en ligne pour tous les membres du projet  
- l’IDE NetBeans pour créer le code source  
- Wampserver afin de pouvoir visualiser l’application.  
- JMerise pour modéliser les bases de données

Nature du projet :  
**(développement d’un composant, d’un client lourd, d’une bibliothèque de classes…)**  
-   
-

Méthodologie choisie :  
**(implémentation par les cas d'utilisation…)**  
-   
-

# III/ Modélisation fonctionnelle

## Méthodologie mise en œuvre pour l’implémentation du code :

**-(L’expression des besoins et la modélisation fonctionnelle en UML par les cas d'utilisation (voir exemple du GAB réalisé ensemble...))**

Cas d’utilisation du comptable :  
**-(une nouvelle version de l'application web avec les nouvelles fonctionnalités opérationnelles (le comptable accède à l'application pour consulter les fiches de frais et valider les notes de frais).)**

**-(Diagramme UML et description textuelle des cas d’utilisation du comptable)**

# IV/ IHM

L’Intéraction entre Humain et Machine est l’ensemble des dispositifs matériels et logiciels permettant à un utilisateur de communiquer avec un système informatique.

**- (Maquette des écrans pour le rôle comptable (+ exemple de code significatif d’une Vue))**

# V/ Architecture applicative

**- (Screen avec explication de l’intérêt de cette architecture, la répartition des rôles entre Modèle, Vue et Contrôleur (cf ressources pdf fournies, reprendre et incorporer les schémas pour expliquer les interactions entre Modèle, Vue et Contrôleur). S’appuyer également sur la ressource fournie ArchitectureApllicative\_gsb.pdf et donner un extrait de code source d’un contrôleur.)**

# VI/ Persistance des données

## A/ Schéma de données (MCD)

## B/ Extrait du script SQL de création des tables (contraintes d’intégrité référentielle)

# VII/ Accès aux données

## A/ PDO

Php Data Object, est une extension PHP qui permet d'utiliser une base de données en programmant avec un style orienté objet. Il s'agit une couche d'abstraction des fonctions d'accès aux bases de données. Ce qui signifie que les fonctions d'accès seront universelles. PDO ne fournis pas une abstraction de base de données.

PDO permet l'utilisation simultanée ou alternée de plusieurs Systèmes de Gestion de Bases de Données avec le même code PHP.

PDO comporte trois classes :

- La classe PDO, qui représente une connexion à la base de données.

- La classe PDOStatement, qui représente d'une part une requête préparée et d'autre part le jeu de résultats de la requête une fois qu'elle est exécutée. Cette classe offre des méthodes de parcours, de comptage et d'informations ;

- La classe PDOException, qui représente une erreur émise par PDO.

**-(mise en oeuvre dans le Modèle (classe pdogsb pour l’accès aux données (diagramme de classe, fournir le code des 2 ou 3 méthodes les plus significatives))**

## B/ Design Pattern Singleton

Nous avons utilisé une version de PDO implémentant le design pattern Singleton. Ce dernier permet de récupérer notre objet PDO à la demande (lorsque nous voulons faire une requête) en se connectant à la base de données lors de son premier appel. Ainsi, la connexion à la Base De Données est gérée de façon transparente et se fait uniquement en cas de besoin.

# VIII/ Déploiement, mise en production de l’application

-(VM Windows server / Apache / MySQL)

# IX/ Tests fonctionnels

-(Recette de l’application, voir fiche type de rapport de tests fonctionnels)

# X/ Difficultés rencontrées

# XI/ Améliorations possibles

# XII/ Conclusion