

BTS SIO SLAM

Session 2022

Ateliers de Professionnalisation



Galaxy Swiss Bourdin (GSB)

Application métier de gestion de frais

Sommaire

Conseils généraux	4
Partie 1 : Présentation du contexte	7
Partie 2 : Présentation de l'application	15
Partie 3 : Missions à réaliser	27
Partie 4 : Documents	37

Conseils généraux

Importance des AP

Les Ateliers de Professionnalisation ont pour but de mettre en œuvre les savoirs abordés dans les différents cours pour acquérir des compétences techniques globales mais aussi spécifiques à votre spécialisation (SLAM).

Au niveau de l'examen, les AP ont une grande importance :

- Ils doivent vous permettre de renseigner votre portefeuille de compétences, au même titre que vos expériences en stage, en vue de l'épreuve E4 "Support et mise à disposition de services informatiques".
- Même si ce n'est pas obligatoire, il est fortement conseillé de les utiliser pour la préparation de l'épreuve pratique E5 "Conception et développement d'applications".

L'AP présenté dans ce fascicule est basé sur l'un des contextes professionnels proposés par le Ministère pour le BTS SIO. Le fait de travailler sur ce contexte répond donc précisément aux attentes du Ministère pour l'examen.

Organisation du travail

Un AP a pour but de vous plonger dans un contexte professionnel, un peu comme vous le seriez dans le cadre d'un stage. Une application en partie créée est alors présentée.

À partir de cet existant, plusieurs missions vous sont confiées. Vous pouvez traiter ces missions en autonomie ou en collaboration avec un autre étudiant (si vous êtes en contact avec d'autres étudiants, le travail à distance est toujours possible et peut même s'avérer être une expérience enrichissante).

Que ce soit en autonomie ou en collaboration, utilisez un logiciel de gestion de versions de préférence sur un serveur distant, pour mémoriser l'évolution de votre travail.

Puisque vous partez d'un existant, vous devez récupérer des fichiers qui se trouvent sur la plateforme Moodle suivante : <https://moodle.gil83.fr>.

Un professeur est là pour répondre à vos questions et pour vous conseiller au cours de votre travail sur les AP, comme dans le cadre d'un stage.

Organisation du fascicule

Ce fascicule contient la présentation d'un AP.

Il se divise en quatre parties :

La partie 1 présente de façon détaillée le contexte de l'entreprise. C'est le genre d'information qui peut vous être communiqué lors d'un premier jour de stage. Vous devez commencer par lire attentivement cette partie et ne pas hésiter à prendre des notes, comme vous le feriez en stage, pour mémoriser les points essentiels qui pourront vous servir. En effet, le contexte est global et peut être exploité aussi bien par des étudiants de l'option SLAM que des étudiants de l'option SISR. Ce sont les missions qui sont ensuite différentes d'une option à l'autre. Ne pensez donc pas que le contexte se limite à présenter uniquement les aspects qui peuvent vous servir.

La partie 2 présente l'application existante qui est en partie créée. Prenez le temps, là aussi, de bien étudier les différents aspects présentés avant de vous lancer dans les missions : vous risqueriez de passer à côté d'informations importantes.

La partie 3 présente les missions à réaliser. Dans un premier temps, la configuration nécessaire est présentée. Ensuite, chaque mission est expliquée, un peu comme elle le serait dans le cadre d'un stage. À vous ensuite de gérer votre travail sur la mission confiée. Certaines missions sont plus complexes que d'autres, c'est normal : le niveau de complexité est précisé. Enfin, vous trouverez un tableau de correspondance entre les attentes du cahier des charges officiel et la portée de chaque mission par rapport à ces attentes.

La partie 4 présente les documents dont vous aurez besoin pour réaliser les différentes missions. Ces documents seront régulièrement sollicités.

N'oubliez pas qu'un professeur est là pour vous aider et répondre à vos questions, durant votre travail sur les AP.

Travail pour l'épreuve E5

Pour l'épreuve pratique E5 "Conception et développement d'applications", vous devez présenter deux contextes et plusieurs activités réalisées dans ces contextes. Au final, les activités présentées doivent répondre aux attentes du cahier des charges officiel publié par le Ministère (<https://www.education.gouv.fr/bo/18/Hebdo8/ESRS1801459N.htm>¹), mais les points qui vous concernent sont rappelés dans ce fascicule.

Vous n'êtes pas obligé de présenter les travaux de l'AP pour l'épreuve E5. Cependant, cela vous est fortement conseillé car tout a été étudié pour répondre précisément aux exigences très strictes de cette épreuve.

Donc, une fois le travail sur ce fascicule terminé, vous pouvez sélectionner des missions que vous désirez retenir pour les présenter dans le cadre de l'épreuve E5. Vous devrez alors préciser si vous avez réalisé chaque mission en autonomie ou en collaboration avec un autre étudiant. Dans ce dernier cas, vous devrez préciser clairement les parties de la mission qui vous ont été confiées.

Attention, lors du choix des missions, contrôlez bien que vous respectez le cahier des charges officiel : à la fin de la partie 3 de ce fascicule, vous trouverez un tableau de correspondance qui vous aidera à contrôler cet aspect.

Travail pour l'épreuve E4

Pour l'épreuve orale E4 "Support et mise à disposition de services informatiques", vous devez présenter, entre autres, un portefeuille de compétences complet, accompagné d'un tableau de synthèse de ces compétences.

Le portefeuille de compétences doit être alimenté, à votre initiative, suite aux différentes expériences techniques. Les deux sources principales d'alimentation du portefeuille de compétences sont les stages et les AP. Vous pourrez aussi insérer des compétences personnelles ou d'autres acquises dans les travaux pratiques.

Logiciels utilisés

Vous allez utiliser plusieurs logiciels dans ce fascicule. Leurs téléchargements, installation et configuration ont déjà été abordés dans les différents cours que vous avez étudiés. Les consignes ne seront donc pas rappelées ici : vous devez commencer un travail d'autonomie à ce niveau-là.

Bon courage !

¹ Cahier des charges de l'épreuve du BTS SIO version 1 (la version 2022 n'est pas publiée à l'heure où ce document vous est distribué).

Partie 1

Présentation du contexte

Dans cette partie, le contexte de l'entreprise est présenté de façon détaillée, un peu comme ce serait le cas lors d'un premier jour de stage. Vous devez en faire une lecture attentive. Tous les aspects ne vous seront pas forcément utiles, cependant il est important que vous ayez cette vision globale du contexte professionnel. N'hésitez pas à prendre des notes pour récupérer les informations essentielles.

► Pré-requis

Aucun.

► Travail attendu en fin de partie

Avoir fait un résumé des informations présentées, en particulier tout ce qui peut être utile pour un étudiant de l'option SLAM.

► Contenu

1.	Description du laboratoire GSB	8
2.	Description du Système Informatique	9
3.	Organisation du réseau.....	10
4.	Salle serveur et connexion internet.....	11
5.	Domaine d'étude	12

1. Description du laboratoire GSB

Le secteur d'activité

L'industrie pharmaceutique est un secteur très lucratif dans lequel le mouvement de fusion acquisition est très fort. Les regroupements de laboratoires ces dernières années ont donné naissance à des entités gigantesques au sein desquelles le travail est longtemps resté organisé selon les anciennes structures.

Des déboires divers récents autour de médicaments ou molécules ayant entraîné des complications médicales ont fait s'élever des voix contre une partie de l'activité des laboratoires : la visite médicale, réputée être le lieu d'arrangements entre l'industrie et les praticiens, et tout du moins un terrain d'influence opaque.

L'entreprise

Le laboratoire Galaxy Swiss Bourdin (GSB) est issu de la fusion entre le géant américain Galaxy (spécialisé dans le secteur des maladies virales dont le SIDA et les hépatites) et le conglomérat européen Swiss Bourdin (travaillant sur des médicaments plus conventionnels), lui-même déjà union de trois petits laboratoires.

En 2009, les deux géants pharmaceutiques ont uni leurs forces pour créer un leader de ce secteur industriel. L'entité Galaxy Swiss Bourdin Europe a établi son siège administratif à Paris. Le siège social de la multinationale est situé à Philadelphie, Pennsylvanie, aux États-Unis.

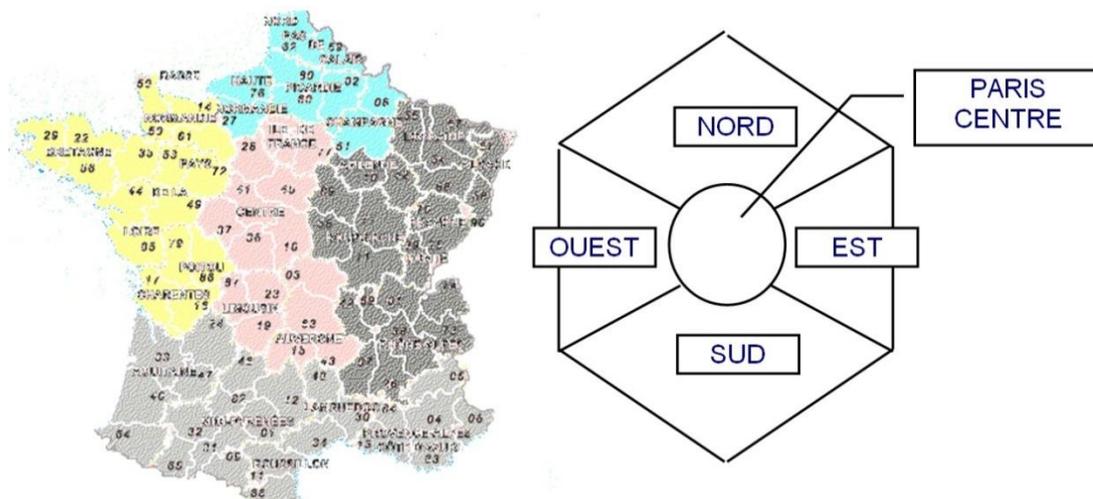
La France a été choisie comme témoin pour l'amélioration du suivi de l'activité de visite.

Réorganisation

Une conséquence de cette fusion, est la recherche d'une optimisation de l'activité du groupe ainsi constitué en réalisant des économies d'échelle dans la production et la distribution des médicaments (en passant par une nécessaire restructuration et vague de licenciement), tout en prenant le meilleur des deux laboratoires sur les produits concurrents.

L'entreprise compte 480 visiteurs médicaux en France métropolitaine (Corse comprise), et 60 dans les départements et territoires d'outre-mer. Les territoires sont répartis en 7 secteurs géographiques (Paris-Centre, Sud, Nord, Ouest, Est, DTOM Caraïbes-Amériques, DTOM Asie-Afrique).

Une vision partielle de cette organisation est présentée ci-dessous.



Après deux années de réorganisations internes, tant au niveau du personnel que du fonctionnement administratif, l'entreprise GSB souhaite moderniser l'activité de visite médicale.

2. Description du Système Informatique

Le système informatique

Sur le site parisien, toutes les fonctions administratives (gestion des ressources humaines, comptabilité, direction, commerciale, etc.) sont présentes. On trouve en outre un service labo-recherche, le service juridique et le service communication.

La salle serveur occupe le 6ème étage du bâtiment et les accès y sont restreints (étage accessible par ascenseur à l'aide d'une clé sécurisée, portes d'accès par escalier munies d'un lecteur de badge, sas d'entrée avec gardien présent 24h/24).

Les serveurs assurent les fonctions de base du réseau (DHCP, DNS, Annuaire et gestion centralisée des environnements) et les fonctions de communication (Intranet, Messagerie, Agenda partagé, etc.).

On trouve aussi de nombreuses applications métier (base d'information pharmaceutique, serveurs dédiés à la recherche, base de données des produits du laboratoire, base de données des licences d'exploitation pharmaceutique, etc.) et les fonctions plus génériques de toute entreprise (Progiciel de Gestion Intégré avec ses modules RH, GRC, etc.).

Un nombre croissant de serveurs est virtualisé.

Constitué autour de VLAN, le réseau segmente les services de manière à fluidifier le trafic.

Les données de l'entreprise sont considérées comme stratégiques et ne peuvent tolérer ni fuite, ni destruction. L'ensemble des informations est répliqué quotidiennement aux États-Unis par un lien dédié. Toutes les fonctions de redondances (RAID, alimentation, lien réseau redondant, Spanning-tree, clustering, etc.) sont mises en œuvre pour assurer une tolérance aux pannes maximale.

La gestion informatique

La DSI (Direction des Services Informatiques) est une entité importante de la structure Europe qui participe aux choix stratégiques.

Pour Swiss-Bourdin, qui occupait le siège parisien avant la fusion, l'outil informatique et l'utilisation d'outils décisionnels pour améliorer la vision et la planification de l'activité ont toujours fait partie de la politique maison, en particulier pour ce qui concerne la partie recherche, production, communication et juridique.

La partie commerciale a été le parent pauvre de cette informatisation, les visiteurs étant vus comme des acteurs distants autonomes. La DSI a convaincu l'entreprise que l'intégration des données fournies par cette partie aura un impact important sur l'ensemble de l'activité.

L'équipement

L'informatique est fortement répandue sur le site. Chaque employé est équipé d'un poste fixe relié au système central. On dénombre ainsi plus de 350 équipements terminaux et un nombre de serveurs physiques conséquent (45 en 2012) sur lesquels tournent plus de 100 serveurs virtuels.

On trouve aussi des stations de travail plus puissantes dans la partie labo-recherche, et une multitude d'ordinateurs portables (personnels de direction, service informatique, services commerciaux, etc).

Les visiteurs médicaux reçoivent une indemnité bisannuelle pour s'équiper en informatique (politique Swiss-Bourdin) ou une dotation en équipement (politique Galaxy). Il n'y a pas à l'heure actuelle d'uniformisation des machines ni du mode de fonctionnement.

Chaque employé de l'entreprise a une adresse de messagerie de la forme nomUtilisateur@swiss-galaxy.com. Les anciennes adresses de chaque laboratoire ont été définitivement fermées au 1er janvier 2011.

3. Organisation du réseau

Répartition des services

Chaque étage dispose d'une baie de brassage qui le relie par une fibre à la baie centrale de la salle serveurs.

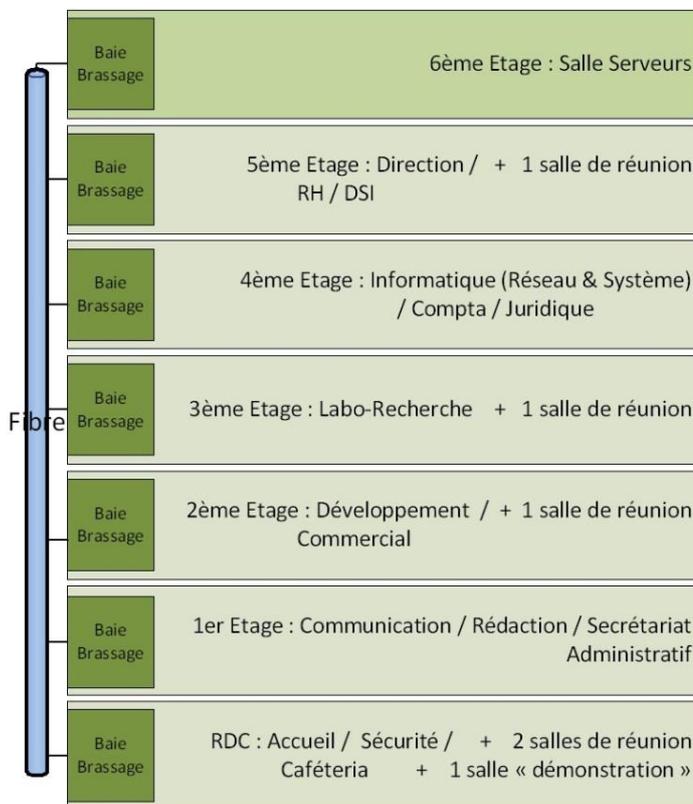
Toutes les salles de réunion sont équipées d'un point d'accès WiFi positionné par défaut dans le VLAN "Visiteurs" qui autorise uniquement un accès Internet.

Les portables connectés en WiFi à ce point d'accès reçoivent ainsi une adresse IP et n'ont, par conséquent accès qu'aux services DHCP et DNS.

Le point d'accès peut être configuré à la demande pour être raccordé à un VLAN présent au niveau de l'étage.

Chaque salle de réunion dispose d'un vidéoprojecteur, d'enceintes et d'un tableau numérique interactif.

La salle "Démonstration" est destinée à l'accueil des organismes de santé (AFSSAPS notamment) et des partenaires scientifiques. Elle dispose de paillasses et d'équipements de laboratoire, en plus d'une salle de réunion.



Segmentation

L'organisation des VLAN et de l'adressage IP est la suivante :

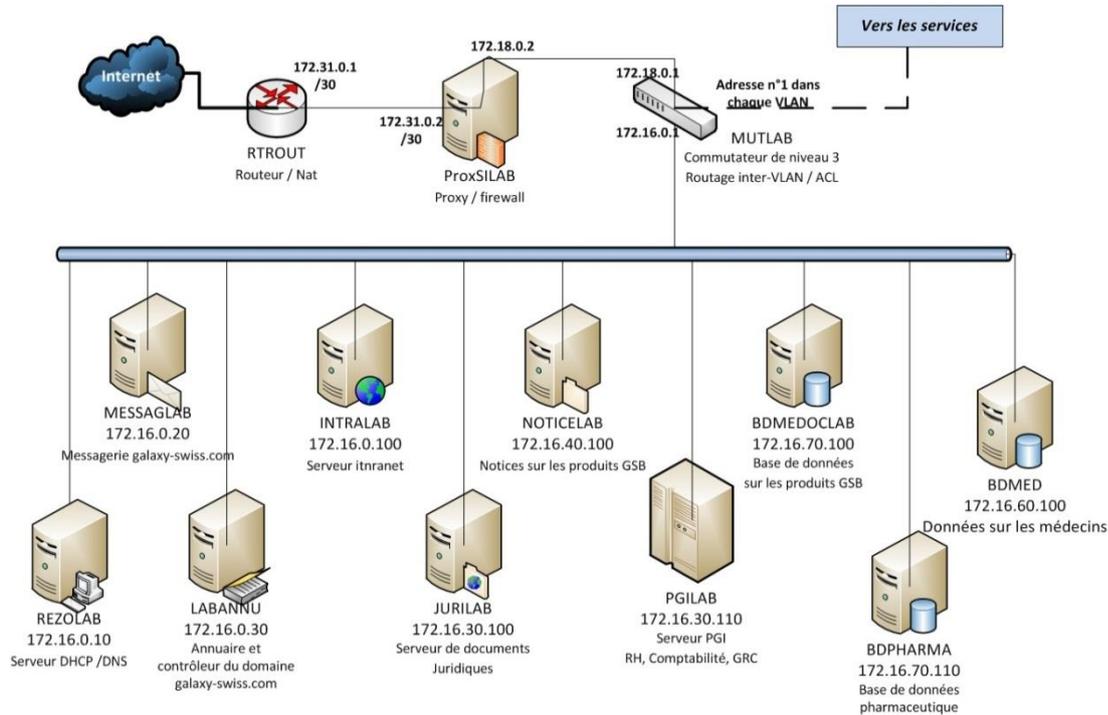
N° VLAN	Service(s)	Adressage IP
10	Réseau & Système	192.168.10.0/24
20	Direction / DSI	192.168.20.0/24
30	RH / Compta / Juridique / Secrétariat Administratif	192.168.30.0/24
40	Communication / Rédaction	192.168.40.0/24
50	Développement	192.168.50.0/24
60	Commercial	192.168.60.0/24
70	Labo-Recherche	192.168.70.0/24
100	Accueil	192.168.100.0/24
150	Visiteurs	192.168.150.0/24
200	Démonstration	192.168.200.0/24
300	Serveurs	172.16.0.0/17
400	Sortie	172.18.0.0/30

Les règles actuelles concernant les vlans sont les suivantes :

- chaque vlan (sauf le vlan visiteur) peut uniquement accéder (quel que soit le protocole) aux vlans "Serveurs" et "Sortie";
- le vlan "Visiteurs" peut uniquement interroger les serveurs DNS et DHCP et sortir sur internet ;

4. Salle serveur et connexion internet

L'organisation des serveurs est la suivante. Il n'est pas précisé si les serveurs sont virtualisés ou non. Seuls les serveurs principaux sont présentés, les redondances n'apparaissent pas.



Les bases de données des serveurs BDMED et BDPHARMA sont achetées périodiquement auprès d'organismes extérieurs et tenues à jour par les employés entre deux achats.

Le commutateur MUTLAB assure un fonctionnement de niveau 3. À ce titre, il réalise un routage inter-vlan en limitant les communications grâce à des listes de contrôles d'accès (ACL).

Le serveur de messagerie et l'intranet sont limités à un usage interne au site parisien. Des services externalisés (relai de messagerie auprès de l'opérateur et recopie d'une partie du serveur intranet sur le serveur Web hébergé chez un prestataire) permettent aux visiteurs médicaux d'utiliser la messagerie de l'entreprise et d'avoir accès aux principales informations de l'intranet (Comité d'entreprise, circulaires importantes, stratégie de l'entreprise, comptes rendus de CA, etc.).

La messagerie publique @swiss-galaxy.com est hébergée aux États-Unis.

5. Domaine d'étude

L'entreprise souhaite porter une attention nouvelle à sa force commerciale dans un double objectif : obtenir une vision plus régulière et efficace de l'activité menée sur le terrain auprès des praticiens, mais aussi redonner confiance aux équipes malmenées par les fusions récentes.

Les visiteurs

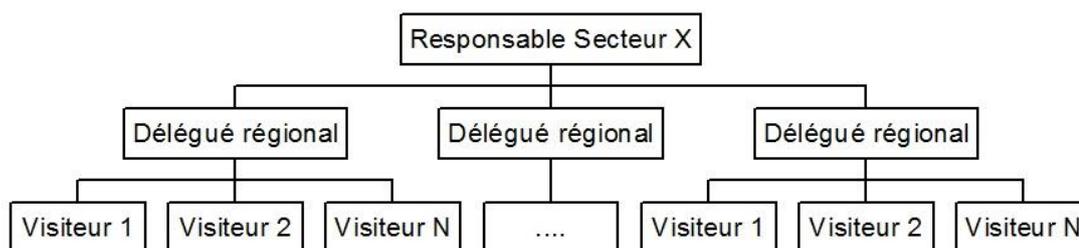
La force commerciale d'un laboratoire pharmaceutique est assurée par un travail de conseil et d'information auprès des prescripteurs. Les visiteurs médicaux (ou délégués) démarchent les médecins, pharmaciens, infirmières et autres métiers de santé susceptibles de prescrire aux patients les produits du laboratoire.

L'objectif d'une visite est d'actualiser et rafraîchir la connaissance des professionnels de santé sur les produits de l'entreprise. Les visiteurs ne font pas de vente, mais leurs interventions ont un impact certain sur la prescription de la pharmacopée du laboratoire.

Pour donner une organisation commune aux délégués médicaux, l'entreprise a adopté l'organisation de la flotte de visiteurs existant chez Galaxy, selon un système hiérarchique par région et, à un niveau supérieur, par secteur géographique (Sud, Nord, Paris-Centre, Antilles-Guyane, etc).

Il n'y a pas eu d'harmonisation de la relation entre les personnels de terrain (Visiteurs et Délégués régionaux) et les responsables de secteur. Les habitudes en cours avant la fusion ont été adaptées sans que soient données des directives au niveau local.

Hiérarchie par Secteur



On souhaite améliorer le contact entre ces acteurs mobiles autonomes et les différents services du siège parisien de l'entité Europe. Il s'agit d'uniformiser la gestion du suivi des visites.

Les visiteurs et les autres services

Les déplacements et actions de terrain menées par les visiteurs engendrent des frais qui doivent être pris en charge par la comptabilité. On cherche à agir au plus juste de manière à limiter les excès sans pour autant diminuer les frais de représentation qui font partie de l'image de marque d'un laboratoire.

Chez Galaxy, le principe d'engagement des frais est celui de la carte bancaire au nom de l'entreprise. Chez Swiss-Bourdin, une gestion forfaitaire des principaux frais permet de limiter les justificatifs. Pour tout le reste, le remboursement est fait après retour des pièces justificatives.

Une gestion unique de ces frais et remboursement pour l'ensemble de la flotte visite est souhaitée.

Les visiteurs récupèrent une information directe sur le terrain. Ceci concerne aussi bien le niveau de la confiance qu'inspire le laboratoire que la lisibilité des notices d'utilisation des médicaments ou encore les éventuels problèmes rencontrés lors de leur utilisation, etc.

Ces informations ne sont actuellement pas systématiquement remontées au siège, ou elles le sont dans des délais jugés trop longs. Le service rédaction qui produit les notices souhaite avoir des remontées plus régulières et directes. Ceci permettra également au service labo-recherche d'engager des évaluations complémentaires.

Le turn-over des visiteurs est de plus en plus important. Pour un délégué régional et plus encore un responsable de secteur, le suivi des équipes devient une véritable activité : obtenir les coordonnées auprès des services RH lors de l'arrivée d'un nouveau personnel, réaliser un suivi personnalisé et former les recrues, etc.

Un accès plus direct aux données de personnel est nécessaire.

Responsabilités

Les **équipes du service développement** auront notamment à produire puis à fournir les éléments applicatifs permettant :

- l'enregistrement d'informations en provenance des visiteurs
- la gestion des frais de déplacement

Les **équipes du service Réseau et système** fourniront les équipements et configuration réseau, ainsi que les ressources serveur nécessaires à héberger les applications mises à disposition de la flotte visite.

Partie 2

Présentation de l'application

Cette partie présente de façon détaillée l'application existante sur laquelle vous allez travailler. Vous devez étudier avec beaucoup d'attention chaque aspect de cette présentation. Vous aurez certainement à vous y référer régulièrement lors de la réalisation des différentes missions. Dans le cadre d'un stage, c'est comme si on vous avait confié un dossier d'existant.

► Pré-requis

Avoir lu et assimilé le contexte de l'entreprise, présenté dans la partie 1.

► Travail attendu en fin de partie

Avoir compris chaque aspect présenté et savoir précisément les objectifs et les fonctionnalités actuelles de l'application.

► Contenu

1.	Cahier des charges.....	16
2.	Description du domaine de gestion.....	17
3.	Spécifications fonctionnelles de l'application de gestion des frais.....	19
4.	Enregistrement des données.....	25

Le travail à réaliser porte sur le développement d'une application de gestion des frais de déplacement, de restauration et d'hébergement générés par l'activité de visite médicale.

L'application va permettre d'établir une gestion plus précise et uniforme entre les entités du laboratoire. Elle devra permettre aux visiteurs d'inscrire leurs dépenses, de visualiser la prise en charge des remboursements (enregistré, validé, remboursé).

1. Cahier des charges

Définition du besoin

Définition de l'objet

Le suivi des frais est actuellement géré de plusieurs façons selon le laboratoire d'origine des visiteurs. On souhaite uniformiser cette gestion

L'application doit permettre d'enregistrer tout frais engagé, aussi bien pour l'activité directe (déplacement, restauration et hébergement) que pour les activités annexes (événementiel, conférences, autres), et de présenter un suivi daté des opérations menées par le service comptable (réception des pièces, validation de la demande de remboursement, mise en paiement, remboursement effectué).

Forme de l'objet

L'application Web destinée aux visiteurs, délégués et responsables de secteur sera en ligne, accessible depuis un ordinateur.

La partie utilisée par les services comptables sera aussi sous forme d'une interface Web.

Le module accessible à la force de visite sera intégré à l'application de gestion des comptes-rendus de visite, mais sous forme d'une interface spécifique (elle ne doit pas être fusionnée à la saisie des CR, elle sera sur un onglet ou une page spécifique).

Accessibilité/Sécurité

L'environnement doit être accessible aux seuls acteurs de l'entreprise.

Une authentification préalable sera nécessaire pour l'accès au contenu.

Tous les échanges produits doivent être cryptés par le serveur Web.

Contraintes

Architecture

L'application respectera l'architecture des scripts fournis concernant la gestion de l'enregistrement des frais engagés par les visiteurs.

Ergonomie

Les pages fournies ont été définies suite à une consultation. Elles constituent une référence ergonomique. Des améliorations ou variations peuvent être proposées.

Codage

Le document 2 "normes de développement" (consultable dans la séquence 4) présente des règles de bonnes pratiques de développement utilisées par le service informatique de GSB pour encadrer le développement d'applications en PHP et en faciliter la maintenance ; l'application fournie (GSB-Appli) s'efforce de les mettre en œuvre.

Les éléments à fournir devront respecter le nommage des fichiers, variables et paramètres, ainsi que les codes couleur et la disposition des éléments déjà fournis.

Environnement

Le langage de script côté serveur doit être le même que celui utilisé dans les pages fournies. L'utilisation de bibliothèques, API ou frameworks est à l'appréciation du prestataire.

Modules

L'application présente deux modules :

- enregistrement et suivi par les visiteurs (code fourni)
- enregistrement des opérations par les comptables

Documentation

La documentation devra présenter l'arborescence des pages pour chaque module, le descriptif des éléments, classes et bibliothèques utilisées, la liste des frameworks ou bibliothèques externes utilisés.

Responsabilités

Le commanditaire fournira à la demande toute information sur le contexte nécessaire à la production de l'application.

Le commanditaire fournira une documentation et des sources exploitables pour la phase de test : base de données exemple, modélisation,...

Le prestataire est à l'initiative de toute proposition technique complémentaire.

Le prestataire fournira un système opérationnel, une documentation technique permettant un transfert de compétence et un mode opératoire propre à chaque module.

2. Description du domaine de gestion

La gestion des frais de déplacement

Grand poste de dépense, la gestion des frais de déplacement des visiteurs demande un suivi très précis. L'enveloppe annuelle pour ce seul poste s'élève à près de 25 millions d'euros. Il n'est donc pas question de le laisser s'envoler, tout en ne limitant pas les visiteurs à des hôtels de second ordre ou des repas chiches (il en va aussi de l'image de marque du laboratoire et de la motivation des équipes).

Les prix d'hébergement ou de nourriture étant variés d'un lieu à l'autre, d'une région à l'autre, il a été procédé à une évaluation statistique permettant de dégager un montant forfaitaire dans la fourchette haute des dépenses pour chaque type de frais standard : repas midi, relais étape (nuit plus repas), nuitée (hôtel seul), kilométrage (remboursement des frais kilométriques, chaque visiteur dispose d'un badge pour le télépéage pour éviter le remboursement de ces petits montants).

Le remboursement de l'ensemble des frais engagés par les visiteurs s'organise mensuellement et donne lieu à une fiche de frais identifiée par le numéro du visiteur et le mois de l'année.

Organisation des remboursements

La gestion est la suivante (voir fiche de remboursement fournie) :

- à chaque dépense type (hôtel, repas,...) correspond un **montant forfaitaire** appliqué (on parle de frais "forfaitisés"). Le justificatif n'est pas demandé (les rapports de visite serviront de preuve) mais doivent être conservés pendant trois années par les visiteurs. Des contrôles réguliers sont faits par les délégués régionaux qui peuvent donner lieu à des demandes de remboursement de trop-perçu par le visiteur.

- Pour toute dépense en dehors du forfait (repas en présence d'un spécialiste lors d'une animation, achat de fournitures, réservation de salle pour une conférence, etc), le visiteur enregistrera la date, le montant et le libellé de la dépense. Il doit fournir au service comptable une facture acquittée. Le système à produire doit lui indiquer le nombre de justificatifs pris en compte dans le remboursement.

Processus à informatiser

Actuellement, au plus tard le 20 de chaque mois, le service comptable adresse aux visiteurs la fiche de demande de remboursement pour le mois en cours (voir document joint). L'application devra permettre de produire automatiquement l'équivalent de ces fiches de manière à les mettre à disposition des visiteurs pour la saisie en ligne.

Saisie

Après authentification grâce aux identifiants à leur disposition, les visiteurs saisissent les quantités de frais forfaitisés et les frais hors forfait engagés pour le mois écoulé.

Ils ont accès en modification à la fiche tout au long du mois et peuvent y ajouter de nouvelles données ou supprimer des éléments saisis.

Les frais saisis peuvent remonter jusqu'à un an en arrière (au mois d'août 2017, on peut saisir des frais engagés de septembre 2016 à août 2017).

Clôture

La fiche est clôturée au dernier jour du mois. Cette clôture sera réalisée par l'application selon l'une des modalités suivantes :

- À la première saisie pour le mois N par le visiteur, sa fiche du mois précédent est clôturée si elle ne l'est pas
- Au début de la campagne de validation des fiches par le service comptable, un script est lancé qui clôture toutes les fiches non clôturées du mois qui va être traité.

Campagne de validation

Entre le 10 et le 20 du mois suivant la saisie par les visiteurs, le service comptabilité opère une validation des fiches.

Les comptables contrôlent que les frais forfaitisés sont conformes : nombre de jours enregistrés ne dépassant pas le nombre de jours effectivement travaillés (congru), distance kilométrique cohérente, éventuellement consultation des fiches de comptes-rendus pour s'assurer des déplacements effectifs. En cas d'incohérence ou d'erreur constatée, un contact est pris par téléphone avec le visiteur pour régler le litige. Les valeurs sont corrigées en conséquence sans que ne soit conservée trace de la modification.

Pour les frais hors forfait, le service comptable s'appuie sur les factures acquittées adressées par les visiteurs au plus tard le 10 du mois suivant la saisie.

Les agents valident ou non (frais non justifié ou non professionnel par exemple) les éléments de la demande. Un frais non validé est supprimé. Le visiteur doit être tenu informé de cette suppression par les comptables. On n'enregistrera pas la raison du refus mais les documents annotés sont conservés par le service comptable.

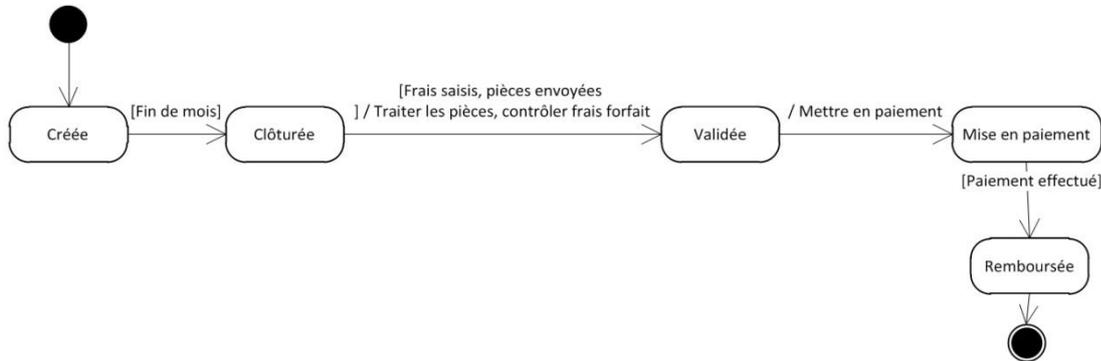
Les éléments reçus après le 10 seront reportés sur le mois ultérieur et seront basculés automatiquement sur la fiche du mois suivant leur saisie (éventuellement créée par l'application si elle ne l'est pas encore) par les comptables.

Après la clôture, les visiteurs peuvent consulter l'évolution de la fiche mais ne peuvent plus la modifier.

Les agents comptables reportent sur chaque facture reçue le numéro de matricule du visiteur, la date (année/mois) de prise en charge et les classent par ordre chronologique dans une pochette nominative pour chaque visiteur.

La mise en paiement est faite au 20 du mois suivant la saisie par les visiteurs.

L'état de la fiche de frais fera l'objet d'un suivi précis qui sera affiché lors de la consultation, selon le cycle suivant :



Les visiteurs doivent pouvoir consulter sur l'année écoulée, pour chaque mois, le montant du remboursement effectué par le laboratoire et le nombre de prestations pris en compte.

3. Spécifications fonctionnelles de l'application de gestion des frais

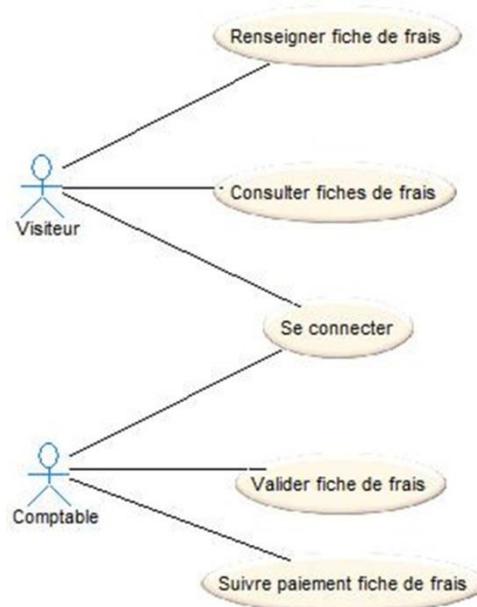
Ce document concerne les spécifications fonctionnelles de l'application web "Gestion des frais".

Cette application web est destinée aux visiteurs médicaux et personnels du service comptable, les premiers pour renseigner et consulter leurs états de frais, les seconds pour réaliser le suivi des états de frais des visiteurs médicaux jusqu'à leur règlement.

Cas d'utilisation

Les besoins sont exprimés ici à l'aide des cas d'utilisation : le diagramme des cas d'utilisation pour la vue synthétique de "qui fait quoi", puis une fiche par cas d'utilisation pour décrire les échanges entre le système et l'utilisateur.

Diagramme des cas d'utilisation



Fiches descriptives des cas d'utilisation

PROJET : Application web de gestion des frais	Description cas d'utilisation
Nom cas d'utilisation : Se connecter	
Acteur déclencheur : Visiteur médical ou Comptable	
Pré conditions : Néant	
Post conditions : L'utilisateur est reconnu visiteur médical ou comptable	
Scénario nominal : <ul style="list-style-type: none">• 1- Le système affiche un formulaire de connexion• 2- L'utilisateur saisit son login et son mot de passe et valide• 3- Le système contrôle les informations de connexion, informe que le profil Visiteur ou Comptable est activé, et maintient affichée l'identité du visiteur médical / comptable connecté.	
Exceptions : <ul style="list-style-type: none">• 3-a : le nom et/ou le mot de passe n'est pas valide<ul style="list-style-type: none">3-a.1 Le système en informe l'utilisateur ; retour à l'étape 1• 4- L'utilisateur demande à se déconnecter• 5- Le système déconnecte l'utilisateur	
Contraintes :	
Questions ouvertes :	

PROJET : Application web de gestion des frais	Description cas d'utilisation
---	--------------------------------------

Nom cas d'utilisation : Renseigner fiche de frais
Acteur déclencheur : Visiteur médical
Pré conditions : Visiteur médical authentifié
Post conditions : néant
<p>Scénario nominal :</p> <ol style="list-style-type: none"> 1. L'utilisateur demande à saisir un ou plusieurs frais pour le mois courant. 2. Le système retourne les frais actuellement saisis - éléments forfaitisés et hors forfait - pour le mois courant. 3. L'utilisateur modifie une ou des valeurs des frais au forfait et demande la validation. 4. Le système enregistre cette ou ces modifications et retourne ces valeurs à jour. 5. L'utilisateur ajoute un nouveau frais hors forfait en renseignant les différents champs – date d'engagement, libellé, montant - et valide. 6. Le système enregistre la ligne de frais hors forfait.
<p>Exceptions :</p> <ul style="list-style-type: none"> • 2.a- C'est la première saisie pour le mois courant. Si ce n'est pas encore fait, le système clôt la fiche du mois précédent et crée une nouvelle fiche de frais avec des valeurs initialisées à 0. Retour à 3. • 4.a. Une valeur modifiée n'est pas numérique : le système indique "Valeur numérique attendue". Retour à 3. • 6.a Un des champs n'est pas renseigné : le système indique : "Le champ date (ou libellé ou montant) doit être renseigné". • 6.b La date d'engagement des frais hors forfait est invalide : le système indique "La date d'engagement doit être valide". Retour à 5. • 6.c La date d'engagement des frais hors forfait date de plus d'un an. Le système indique "La date d'engagement doit se situer dans l'année écoulée". Retour à 5. • 7. L'utilisateur sélectionne un frais hors forfait pour suppression. • 8. Le système enregistre cette suppression après une demande de confirmation.
Contraintes :

PROJET : Application web de gestion des frais	Description cas d'utilisation
--	--------------------------------------

Nom cas d'utilisation : Consulter mes fiches de frais
Acteur déclencheur : Visiteur médical
Pré conditions : Visiteur médical authentifié
Post conditions : néant
Scénario nominal : <ul style="list-style-type: none"> 1. L'utilisateur demande à consulter ses frais. 2. Le système invite à sélectionner un mois donné. 3. L'utilisateur sélectionne un mois donné, puis valide. 4. Le système affiche l'état de la fiche de frais avec la date associée, les éléments forfaitisés – quantité pour chaque type de frais forfaitisé - et non forfaitisés – montant, libellé et date d'engagement - existant pour la fiche de frais du mois demandé.
Exceptions :
Contraintes :
Questions ouvertes : La sélection d'un mois sera facilitée par l'IHM. Il est possible de proposer les mois pour lesquels le visiteur médical connecté dispose d'une fiche de frais. On pourra se restreindre à remonter jusqu'au début de l'année civile précédente.

PROJET : Application web de gestion des frais	Description cas d'utilisation
--	--------------------------------------

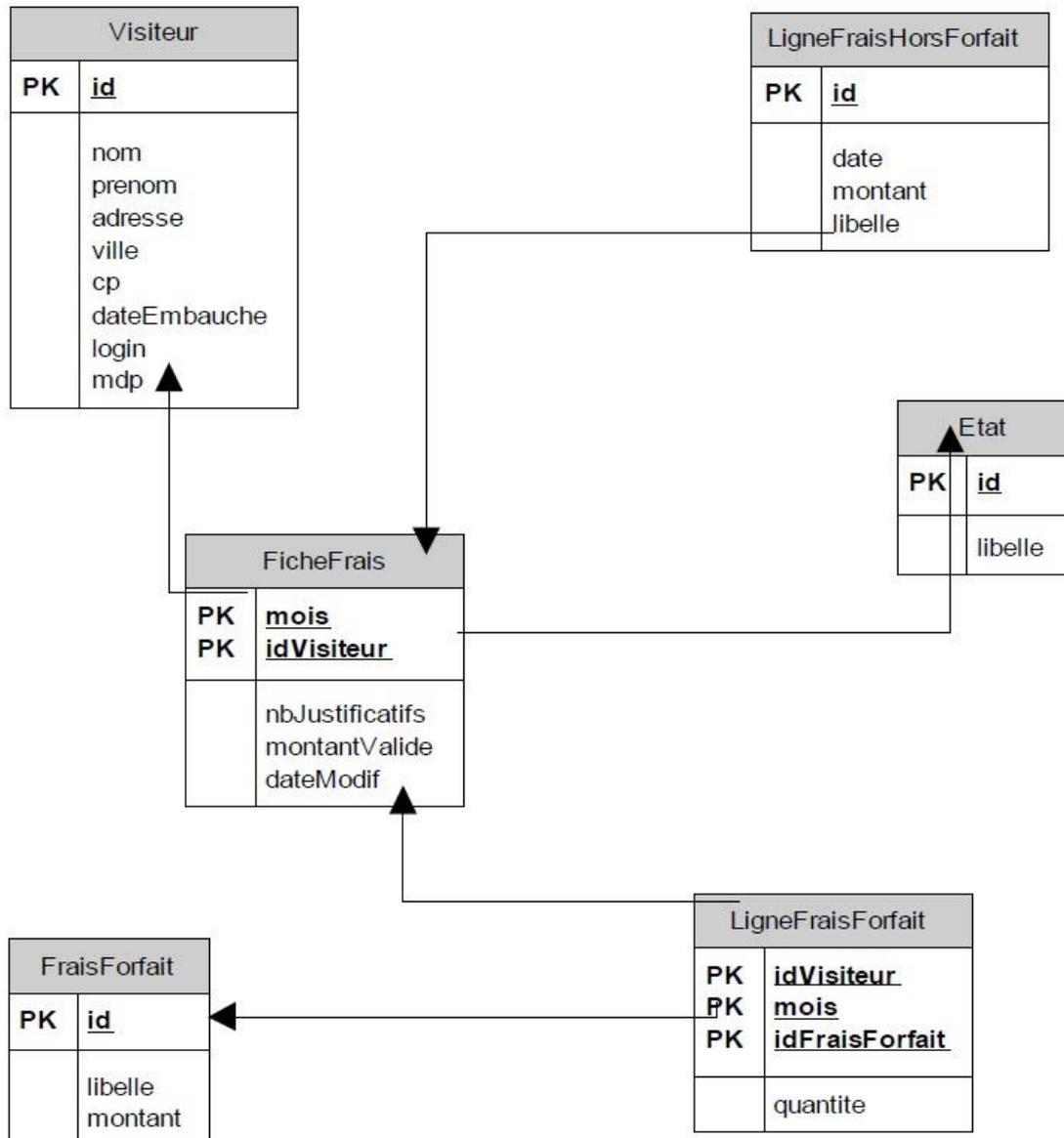
Nom cas d'utilisation : Valider fiche de frais
Acteur déclencheur : Comptable
Pré conditions : Utilisateur Comptable authentifié Toutes les fiches du mois qui vient de s'achever sont clôturées par un script Le comptable dispose de tous les éléments pour évaluer la conformité des frais forfaitisés
Post conditions : Néant
Scénario nominal : <ol style="list-style-type: none"> 1. L'utilisateur demande à valider les fiches de frais 2. Le système propose de choisir le visiteur et le mois concernés 3. L'utilisateur sélectionne les informations et valide 4. Le système affiche le détail de la fiche de frais –frais forfaitisés et hors forfait 5. L'utilisateur actualise les informations des frais forfaitisés. 6. Le système indique que la modification a été prise en compte et affiche les informations actualisées. 7. L'utilisateur demande la suppression des lignes de frais hors forfait non valides 8. Le système modifie le libellé en ajoutant en début le texte « REFUSE : ». 9. L'utilisateur valide la fiche. 10. Le système passe la fiche à l'état « Validée » et met à jour la date de modification de la fiche.
Exceptions : <ul style="list-style-type: none"> • 4-a : Aucune fiche de frais n'existe, le système affiche 'Pas de fiche de frais pour ce visiteur ce mois'. Retour au 2 • 7.a : L'utilisateur demande le report des frais hors forfait pour lesquels une facture acquittée n'a pas été reçue dans les temps. • 8.a : Le système ajoute la ligne hors forfait dans la fiche du mois suivant et la supprime de la fiche courante. Si la fiche du mois suivant n'existe pas, le système génère une nouvelle fiche pour le visiteur en cours de traitement et pour le mois suivant. Cette nouvelle fiche a des valeurs à 0 pour les frais forfaitisés et est dans l'état « Saisie en cours ». Retour au 9 • 8.b : le texte ainsi complété dépasse la taille maximale du champ « libelle » : le texte est tronqué par la fin au nombre de caractères du champ « libelle »
Contraintes :
Questions ouvertes :

PROJET : Application web de gestion des frais	Description cas d'utilisation
--	--------------------------------------

Nom cas d'utilisation : Suivre le paiement fiche de frais
Acteur déclencheur : Comptable
Pré conditions : Utilisateur Comptable authentifié
Post conditions : Néant
Scénario nominal : <ol style="list-style-type: none"> 1. L'utilisateur demande à suivre le paiement les fiches de frais. 2. Le système propose de choisir une fiche de frais parmi celles à valider et mises en paiement 3. L'utilisateur sélectionne les informations et valide 4. Le système affiche le détail de la fiche de frais –frais forfaitisés et hors forfait 5. L'utilisateur « Met en paiement » la fiche de frais 6. Le système modifie l'état de la fiche à « Mise en paiement » et met à jour la date de modification.
Exceptions : <ul style="list-style-type: none"> • 4-a : Aucune fiche de frais n'existe, le système affiche 'Pas de fiche de frais pour ce visiteur ce mois'. Retour au 2 • 5.a : L'utilisateur indique que la fiche a été effectivement payée • 6.a : Le système modifie l'état de la fiche à « Remboursée » et enregistre la date de modification
Contraintes :
Questions ouvertes :

4. Enregistrement des données

La base de données est modélisée ainsi :



Partie 3

Missions à réaliser

Cette partie contient les différentes missions qui vous sont proposées et qui portent sur l'application présentée dans la partie 2, basée sur le contexte présenté dans la partie 1. La configuration nécessaire pour réaliser les missions est expliquée dans un premier temps. Chaque mission est présentée avec son niveau de difficulté. Au final, un bilan présente le tableau de correspondance entre les points attendus dans le cahier des charges officiel publié par le Ministère pour l'épreuve pratique E5, et les points abordés par les différentes missions.

► Pré-requis

Avoir lu et assimilé le contexte de l'entreprise, présenté dans la partie 1. Avoir lu et assimilé la présentation de l'application existante, présentée dans la partie 2. Avoir étudié les différents cours informatiques proposés dans la formation (au début de chaque mission, le niveau de difficulté sera précisé, ainsi que les prérequis : vous pourrez donc progressivement aborder les missions au fur et à mesure de l'étude des cours).

► Travail attendu en fin de partie

Avoir réalisé les missions demandées. Avoir sélectionné les missions retenues pour être présentées à l'épreuve E5 (en contrôlant que les missions retenues couvrent bien les points abordés dans le cahier des charges officiel de l'épreuve). Avoir alimenté le portefeuille de compétences en fonction des compétences acquises dans les missions. Avoir complété le tableau de synthèse associé résumant le contenu du portefeuille de compétences.

► Contenu

1.	Configuration	29
2.	Missions	29
3.	Bilan	35

1. Configuration

Pour réaliser les différentes missions, vous allez devoir utiliser un serveur web et développer sous PHP. Vous pouvez travailler en local pour vos tests, comme vous avez appris à le faire dans le cours "Développement d'applications" étudié en première année. Une autre solution consiste à travailler avec un serveur distant. Quelque soit la solution adoptée il est très conseillé au final d'installer l'application sur un serveur en ligne afin que vous puissiez montrer au jury que vous savez mettre en ligne un site, modifier les pages et le tester.

Afin de respecter le cahier des charges national : Configurez vos IDE (NetBeans, Visual Studio et Android Studio) pour qu'ils intègrent le débogage, la gestion de version, la documentation automatique, les tests unitaires et le contrôle de qualité et de respect des standards.

2. Missions

Voici les différentes missions qui vous sont confiées. Chaque mission est présentée avec son niveau de difficulté, le temps de réalisation estimé, ses prérequis (les cours à étudier avant d'aborder la mission) et les documents à consulter : vous trouverez ces documents dans la partie 4 de ce fascicule. Attention, vous travaillez dans les conditions d'un stage, donc toutes les informations ne sont pas forcément données. Dès le départ, vous pouvez être confronté à des problèmes : ceci est volontaire et doit vous pousser à réfléchir sur les choix à faire.

Mission 1 : Développement de la partie comptable

Difficulté	Moyenne
Temps estimé	70h
Prérequis	Disposer d'un serveur web avec MariaDB et PHP ainsi que de l'IDE NetBeans.
Technologies	NetBeans : PHP, MariaDB, documentation technique

Partie 3

Missions à réaliser

Page 29

Coder la partie comptable en respectant le cas d'utilisation correspondant.

Attention, vous devez respecter les règles présentées dans le document "Normes de développement". Des ébauches de formulaires ont été réalisées et sont disponibles dans : Ressources\GSB-EbaucheFormulaires.

Les tâches 1 et 2 sont obligatoires.

Tâche 1 : Validation d'une fiche de frais

Coder la page de validation d'une fiche de frais en respectant le cas d'utilisation "Valider fiche de frais".

Tâche 2 : Suivi du paiement des fiches de frais

Coder la page de suivi de paiement en respectant le cas d'utilisation "Suivre le paiement fiche de frais".

Tâche 3 : Production de la documentation

Générer la documentation (dans le document "Normes de développement" il est indiqué que l'on utilise normalement phpDocumentor, mais exceptionnellement vous pouvez choisir d'autres outils. L'important étant bien évidemment de générer automatiquement de la documentation technique).

Tâche 4 : Gestion du refus de certains frais hors forfait

Prendre en compte le fait qu'une ligne de frais hors forfait "refusée" ne doit pas être supprimée mais ne doit pas non plus être prise en compte (seul le libellé change avec l'ajout du texte "REFUSE" en début de libellé).

Tâche 5 : Sécurisation des mots de passe stockés

Hasher le mot de passe dans la base de données (SHA-224, SHA-256, SHA-384, SHA-512... au choix). À cette étape, il est important de faire des recherches sur les algorithmes de hashages existants et d'être capable de donner par exemple la raison de la non-présence de MD5 ou SHA-1 dans la liste proposée ci-dessus...

Tâche 6 : Gestion plus fine de l'indemnisation kilométrique

Distinguer l'indemnité kilométrique en fonction de la puissance du véhicule.

Vous disposez du document "Ressources\ETAT-FRAIS.docx" qui vous fournit le barème à appliquer en fonction du type de véhicule.

Tâche 7 : Génération d'un état de frais au format PDF

Au niveau de l'UC "Consulter fiche frais", rendre la fiche de frais facilement imprimable en générant un PDF (voir par exemple la classe libre FPDF sur fpdf.org). Un exemple de fiche est disponible dans : Ressources\REMBOURSEMENT_FRAIS_201707-LEPLATAUFRAY.docx.

Ajouter un lien "Télécharger PDF" dans la page de consultation des fiches de frais.

Tâche 8 : Davantage d'écologie dans l'application

Veiller à ce que le PDF ne soit généré qu'une seule et unique fois afin de ne pas effectuer de traitements inutiles (orientation "Green-IT").

Mission 2 : Gestion de la clôture

Difficulté	Facile (sauf dernière tâche : moyenne)
Temps estimé	20h
Prérequis	Disposer de Visual Studio
Technologies	Visual Studio : C#, création d'un service Windows, tests unitaires, documentation technique

Attention, dans cette mission, toutes les tâches sont obligatoires, en particulier la tâche 4. Cette mission est trop légère pour se permettre d'être tronquée.

Le cahier des charges de l'application Frais GSB stipule que la fiche d'un visiteur est clôturée au dernier jour du mois. Cette clôture sera réalisée par l'application selon l'une des modalités suivantes :

À la première saisie pour le mois N par le visiteur, sa fiche du mois précédent est clôturée si elle ne l'est pas.

Au début de la campagne de validation des fiches par le service comptable, un script est lancé qui clôture toutes les fiches non clôturées du mois qui va être traité.

Nous nous intéresserons ici à la deuxième éventualité.

D'autre part, il est dit que la mise en paiement est faite au 20 du mois suivant la saisie par les visiteurs.

Nous voudrions répondre à ces deux objectifs en développant une application C# avec VS.Net.

Cette application va devoir permettre, au début de la campagne de validation, c'est-à-dire à partir du 1er jour du mois N, la clôture de toutes les fiches créées le mois N-1.

Elle permettra, d'autre part, à partir du 20è jour du mois N la mise en remboursement des fiches créées le mois N-1.

Une nouvelle application C# doit être créée.

Tâche 1 : Création de la classe d'accès aux données

Dans la nouvelle application, une classe d'accès aux données doit être créée, permettant les fonctionnalités classiques d'accès aux données : connexion à la base (ici ce sera la base MySQL), exécution d'une requête d'administration (insert, update, delete...), gestion d'un curseur (exécution d'une requête type select et gestion du résultat avec passage à la ligne suivante, récupération d'un champ, gestion de la fin du curseur...).

Le but est de créer une classe réutilisable.

Tâche 2 : Création d'une classe de gestion de dates

Cette classe doit être abstraite et ne contenir que des méthodes statiques.

Elle doit contenir au moins les méthodes suivantes :

- `getMoisPrecedent` : ne reçoit aucun paramètre et permet de retourner sous forme d'une chaîne de 2 chiffres le numéro du mois précédent par rapport à la date d'aujourd'hui (attention, il faut forcément 2 chiffres : "01" pour janvier, "10" pour octobre...);
- `getMoisPrecedent` : surcharge de la méthode précédente. Cette fois elle reçoit un objet de type `DateTime` en paramètre et retourne le mois précédent de cette date ;
- `getMoisSuivant` : suivant la même logique que les 2 méthodes précédentes, écrire les 2 méthodes pour le mois suivant ;
- `entre` : reçoit en paramètre deux numéros de jours dans le mois, et retourne vrai si la date actuelle se situe entre ces deux jours ;
- `entre` : surcharge de la méthode précédente. Cette fois un troisième paramètre est reçu de type `DateTime` et c'est le jour de cette date qui est testée.

Faites en sorte que le code de la classe soit correctement optimisé (pas de redondance de code).

Réalisez les tests unitaires pour contrôler chaque méthode (cette classe s'y prête de façon idéale).

Tâche 3 : Création de l'application

L'application n'affiche rien. Elle doit juste réaliser le travail demandé sur la base de données, en exploitant les deux classes précédentes, le tout à intervalle régulier (donc en utilisant un timer). Pour les tests, vous utiliserez un intervalle assez court et vous contrôlerez les modifications dans la base de données (pour cela, il faut que vous ayez des informations pertinentes dans la base de données, et que vous trifouilliez les dates pour tester le début du mois, le milieu et la fin.

Rappel des modifications à apporter :

- récupération des fiches créées du mois N-1 et leur mise à jour, en les mettant à l'état 'CL' ; en supposant que la campagne de validation va se passer entre le 1er et le 10 du mois courant, on va, en comparant les dates, s'assurer que l'on se trouve bien dans cet intervalle-là ;
- de la même manière, à partir du 20è jour du mois, on va mettre à jour les fiches validées du mois précédent en les passant à l'état 'RB'.

Une fois l'application créée (et testée), essayez de voir comment générer la documentation technique sous Visual Studio.

Tâche 4 : Création d'un service Windows

Puisque cette application n'affiche rien et qu'elle doit s'exécuter à intervalle régulier, ce serait une bonne idée qu'elle s'exécute en tâche de fond, sans avoir besoin de la lancer, comme un service Windows. Visual Studio est capable de créer un service Windows (toujours en C#). Cherchez le moyen de le faire et créez ce service.

Mission 3 : Application mobile

Difficulté	Difficile (progressif)
Temps estimé	30h
Prérequis	Disposer d'Android Studio
Technologies	Android Studio : Java, MariaDB

Attention, dans cette mission toutes les tâches sont obligatoires, en particulier la tâche 5 permettant de manipuler la base de données. Sans cette tâche, le travail n'est pas assez complexe pour être présenté à l'examen.

Suite aux demandes des visiteurs, une application Android est en cours de développement. Elle doit permettre aux visiteurs de saisir en direct leurs frais (forfaitisés ou hors forfaits). L'application est une sorte de mémo qui permet d'enregistrer l'information à tout moment. Les visiteurs peuvent ensuite consulter leur mobile pour voir ce qu'ils ont enregistré et ainsi remplir le formulaire sur le site officiel. Il est prévu que l'application permette l'envoi direct des informations saisies sur le serveur web qui mettra alors à jour directement la base de données, sans que le visiteur ait à ressaisir les données dans l'application web.

Actuellement, l'application comporte un menu principal permettant d'accéder à la saisie des différentes catégories de frais ainsi que l'activité portant sur la saisie des kilomètres et des frais hors forfait. Une sérialisation permet de mémoriser les informations.

Voici les interfaces correspondantes.

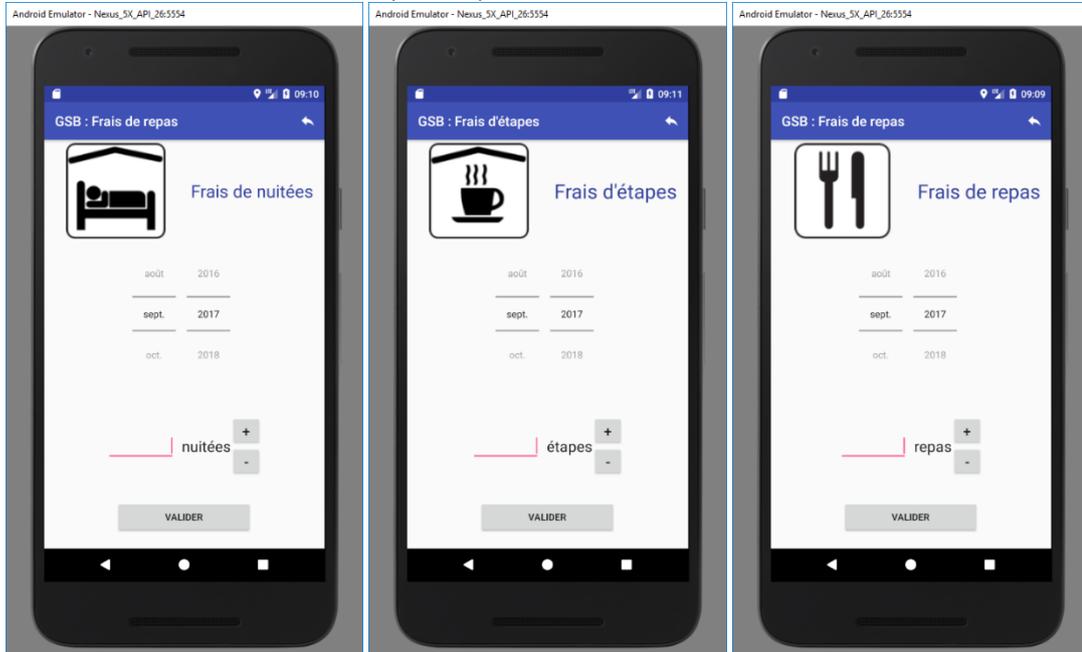
Le menu principal (déjà codé) :



La saisie des Km (déjà codée) :



La saisie des autres frais forfaitisés (à coder) :



Partie 3

Missions à réaliser

La saisie des frais hors forfait, au jour le jour (déjà codée) :



La consultation des frais hors forfait du mois (déjà codée) avec la possibilité de supprimer des lignes (à coder).



Tâche 1 : Configuration

L'application existante a été faite avec l'IDE Android Studio et le SDK de l'API 26 (avec un AVD type Nexus 5, 4,95", 1080x1920 xxhdpi).

La première étape va consister à récupérer le projet existant. Pour cela, voici un petit mode opératoire :

- lancer Android Studio
- choisir « Start a new Android Studio project »
- dans « Application name » saisir « Suivi de vos frais » et dans « Company domain » saisir « bonaparte.fr » puis cliquer ensuite sur « Next »
- dans l'écran suivant, « Phone and Tablet » est normalement déjà coché, choisir comme « Minimum SDK » l'API « 26 : Android 8.0 (Oreo) »
- dans l'écran suivant choisir « Add No Activity » et cliquer sur « Finish »
- une fois l'initialisation du projet terminé (attendre quelques minutes si nécessaire), copier/coller les fichiers fournis dans le répertoire créé par Android Studio (normalement situé

par défaut dans le répertoire C:\Users**<votreCompte>**\AndroidStudioProjects\Suividevosfrais\app\src)

- cliquer sur le menu « File » puis sur « Sync Projects with Gradle Files ».
- cliquer sur le menu « Run » et sur « Run 'app' »
- cliquer sur « Create New Virtual Device »
- choisir la ligne « Phone », « Nexus 5 4,95" 1080x1920 xxhdpi », puis cliquer sur « Next »
- choisir la ligne avec les données suivantes et cliquer « Next » puis sur « Finish » :
 - Release Name : Oreo
 - API Level : 26
 - ABI : x86
 - Target : Android 8.0 (Google Play)
- de retour dans la fenêtre « Select Deployment Target », choisir dans « Available Virtual Devices » l'AVD "Nexus 5 API 26" et cliquer sur « OK ».

Après chargement du système Android, l'application existante se lance.

Tâche 2 : Interdiction de saisie directe des quantités

Modifier le code existant pour interdire la saisie directe des km dans l'activité correspondante : la quantité doit être obtenue uniquement en utilisant les touches + et -.

Tâche 3 : Enregistrement des autres catégories de frais forfaitisés

Créer les autres activités pour la saisie des frais forfaitisés, sur le modèle de la saisie des frais km, en respectant la présentation des interfaces données ci-dessus. Il faudra aussi faire en sorte que les informations soient enregistrées (comme elles le sont déjà pour les km).

Tâche 4 : Suppression de frais hors forfait

Dans l'activité qui affiche le récapitulatif mensuel des frais hors forfait, rajouter un bouton pour la suppression d'une ligne (comme cela est montré dans la capture d'écran ci-dessus). Le bouton devra être ajouté dans le layout_list qui formate l'affichage d'une ligne du listview utilisé pour le récapitulatif. Coder le bouton pour que la ligne soit supprimée et que le frais correspondant soit supprimé dans l'enregistrement. Le codage va se faire dans l'"adapter" de la liste : FraisHfAdapter. Il faut donc dans un premier temps bien s'approprier le code existant pour comprendre le fonctionnement d'un listView.

Pour tester si l'enregistrement se fait correctement, il faut relancer l'émulateur.

Tâche 5 : Synchronisation avec la base de données distante

Écrire le code derrière le bouton de synchronisation du menu principal, qui va permettre d'insérer dans la base distante tout ce qui a été enregistré en local.

Cette tâche est nettement la plus complexe et la plus importante. Il faut penser à la reconnaissance de la personne. Cela suppose que vous devez prévoir une authentification. Vous êtes libre de la méthode à utiliser, le but final étant que la base distante doit être capable de savoir à qui appartiennent les frais reçus pour les enregistrer au bon endroit.

Si vous choisissez de présenter la mission Android, cette tâche est incontournable.

3. Bilan

Ce bilan permet de vous rappeler le cahier des charges et de contrôler les points couverts, dans le cahier des charges officiel de l'épreuve E5, par les différentes missions réalisées.

Cahier des charges épreuve E5	Missions
<p>1.1 Un contexte est composé d'une organisation cliente et d'un prestataire informatique interne ou externe à l'organisation cliente. Ces organisations sont réelles ou directement inspirées du réel. L'organisation cliente et le prestataire informatique sont décrits à travers leurs principaux processus métier et support, leur système d'information et l'ensemble de leurs relations formalisées (contrats ou catalogue de services, politique de sécurité, charte, etc.).</p>	Contexte
<p>1.2 Les besoins de l'organisation cliente en matière de création ou d'amélioration de services informatiques sont clairement identifiés dans un ou plusieurs cahiers des charges qui définissent les contraintes techniques, financières et temporelles à respecter.</p>	Contexte
<p>1.3 L'environnement technologique d'apprentissage supportant le système d'information de l'organisation cliente comporte au moins :</p> <ul style="list-style-type: none"> - un service d'authentification pour les utilisateurs internes et externes à l'organisation ; - un SGBD ; - un accès sécurisé à internet ; - un environnement de travail collaboratif ; - un logiciel de gestion d'incidents ; - un logiciel de gestion des configurations ; - deux serveurs, éventuellement virtualisés, basés sur des systèmes d'exploitation différents, dont l'un est un logiciel open source ; - une solution de sauvegarde ; - des ressources dont l'accès est sécurisé et soumis à habilitation ; - deux types de solution technique d'accès dont une mobile (type smartphone, tablette, ou encore assistant personnel). 	Contexte
<p>1.4 Les logiciels de simulation ou d'émulation sont utilisés en réponse à des besoins de l'organisation. Ils ne peuvent se substituer à des équipements réels dans l'environnement technologique d'apprentissage. Une solution d'infrastructure réduite à une simulation par un logiciel ne peut être acceptée</p>	
<p>1.5 Tous les documents et ressources qui décrivent un contexte doivent être accessibles en ligne aux commissions de correction à partir d'une date fixée par les autorités académiques :</p> <ul style="list-style-type: none"> - documents de présentation des organisations (organisation cliente et prestataire informatique) ; - description de l'environnement technologique d'apprentissage ; - tout ou partie des documents de référence utilisés par l'organisation cliente et par le prestataire informatique qui sont utiles pour définir le contexte (référentiels de bonnes pratiques, normes ou standards, processus, données métiers, etc.) et nécessaires pour le déroulement de l'épreuve ; - les schémas d'infrastructure réseau ; - la documentation technique des services disponibles ; - les fichiers de configuration, la documentation technique des équipements matériels et des logiciels disponibles ; 	

- les éléments financiers et juridiques liés aux services et aux équipements disponibles.	
1.6 Lorsque les deux situations professionnelles présentées par un candidat s'appuient sur deux contextes différents, chaque contexte et son environnement technologique d'apprentissage doivent respecter les règles communes aux deux parcours. Le respect des règles relatives au parcours du candidat (SISR ou SLAM) est mesuré à partir du cumul des caractéristiques des deux environnements technologiques d'apprentissage.	Contexte
3.1 L'environnement technologique supportant le système d'information de l'organisation cliente comporte au moins : - un ou deux environnements de développement disposant d'outils de gestion de tests et supportant un framework et au moins deux langages ; - une bibliothèque de composants logiciels ; - un SGBD avec langage de programmation associé ; - un logiciel de gestion de versions.	Missions 1, 2 ou 3
3.2 Les activités de l'organisation cliente s'appuient sur aux moins deux solutions applicatives opérationnelles permettant d'offrir un accès sécurisé à des données hébergées sur un site distant. Au sein des architectures de ces solutions applicatives, doivent figurer l'exploitation de mécanismes d'appel à des services applicatifs distants et au moins trois des situations ci-dessous : 3.2.1 du code exécuté sur le système d'exploitation d'une solution technique d'accès fixe (type client lourd) ; 3.2.2 du code exécuté dans un navigateur web (type client léger ou riche, applet, etc.) ; 3.2.3 du code exécuté sur le système d'exploitation d'une solution technique d'accès mobile ; 3.2.4 du code exécuté sur le système d'exploitation d'un serveur (servlet, procédure cataloguée, etc.).	Missions 1, 2 ou 3
3.3 Une solution applicative peut être issue d'un développement spécifique ou de la modification du code d'un logiciel (open source par exemple).	Contexte (Développement spécifique)
3.4 Les solutions applicatives présentes dans le contexte sont opérationnelles et leur code source est accessible dans un environnement de développement opérationnel au moment de l'épreuve.	Mise en ligne + Copie locale (à avoir pour l'épreuve)

Partie 4

Documents

Cette partie présente les documents complémentaires nécessaires pour réaliser certaines missions. Chaque mission fait référence aux documents dont elle a besoin.

► Contenu

1.	Architecture applicative de l'application Web.....	38
2.	Normes de développement.....	45
3.	Ébauches des formulaires.....	70

1. Architecture applicative de l'application Web

Principes d'organisation de l'application PHP Gsb-AppliFrais

Nous ne présentons pas ici les avantages de la structuration du code relevant de l'architecture Modèle-Vue-Contrôleur ; de nombreux documents se penchent sur la question.

Des frameworks (Zend, Symfony, PhpCake) fournissent les classes mettant en œuvre cette technologie. Nous avons fait le choix ici de faire le travail « à la main »

Le langage HTML respectera la norme HTML5. Les pages seront validées à l'aide du validateur du W3C : <http://validator.w3.org/check>.

Les règles de style respecteront la norme CSS3. Elles seront validées auprès du validateur <http://jigsaw.w3.org/css-validator/>.

Un développement guidé par les cas d'utilisation

C'est le propre de l'architecture MVC ; le système (l'application) doit répondre aux sollicitations de l'utilisateur. Les cas d'utilisation sont les moyens textuels de décrire ces sollicitations et les réponses.

Prenons l'exemple du cas d'utilisation suivant :

PROJET : Application web de gestion des frais	Description cas d'utilisation
Nom cas d'utilisation : Renseigner fiche de frais	
Acteur déclencheur : Visiteur médical	
Pré conditions : Visiteur médical authentifié	
Post conditions : néant	
Scénario nominal :	
<ul style="list-style-type: none">○ L'utilisateur demande à saisir un ou plusieurs frais pour le mois courant.○ Le système retourne les frais actuellement saisis – éléments forfaitisés et hors forfait - pour le mois courant.○ L'utilisateur modifie une ou des valeurs des frais au forfait et demande la validation.○ Le système enregistre cette ou ces modifications et retourne ces valeurs à jour.○ L'utilisateur ajoute un nouveau frais hors forfait en renseignant les différents champs – date d'engagement, libellé, montant - et valide.○ Le système enregistre la ligne de frais hors forfait.	
Exceptions :	
2.a- C'est la première saisie pour le mois courant. Si ce n'est pas encore fait, le système clôt la fiche du mois précédent et crée une nouvelle fiche de frais avec des valeurs initialisées à 0. Retour à 3. 4.a. Une valeur modifiée n'est pas numérique : le système indique 'Valeur numérique attendue '. Retour à 3. 6.a Un des champs n'est pas renseigné : le système indique : 'Le champ date (ou libellé ou montant) doit être renseigné'. 6.b La date d'engagement des frais hors forfait est invalide : le système indique 'La date d'engagement doit être valide'. Retour à 5. 6.c La date d'engagement des frais hors forfait date de plus d'un an. Le système indique 'La date d'engagement doit se situer dans l'année écoulée'. Retour à 5. 7. L'utilisateur sélectionne un frais hors forfait pour suppression. 8. Le système enregistre cette suppression après une demande de confirmation.	
Contraintes :	

L'utilisateur sollicite à 4 reprises le système (points 1, 3, 5 et 7 en italique gras). Le contrôleur (fichier spécifique) doit donc répondre à ces 4 sollicitations :

```
21 $action = filter_input(INPUT_GET, 'action', FILTER_SANITIZE_STRING);
22 switch ($action) {
23 + case 'saisirFrais':{...5 lines }
28 + case 'validerMajFraisForfait':{...9 lines }
37 + case 'validerCreationFrais':{...17 lines }
54 + case 'supprimerFrais':{...4 lines }
58 }
59 $lesFraisHorsForfait = $pdo->getLesFraisHorsForfait($idVisiteur, $mois);
60 $lesFraisForfait = $pdo->getLesFraisForfait($idVisiteur, $mois);
61 require 'vues/v_listeFraisForfait.php';
62 require 'vues/v_listeFraisHorsForfait.php';
```

Remarque : le code des cases a été plié ici pour se concentrer sur l'essentiel.

Pour chacune des sollicitations, le système réagit et agit en conséquence, par exemple pour la demande de saisie des frais :

```
21 $action = filter_input(INPUT_GET, 'action', FILTER_SANITIZE_STRING);
22 switch ($action) {
23 - case 'saisirFrais':
24 -     if ($pdo->estPremierFraisMois($idVisiteur, $mois)) {
25 -         $pdo->creerNouvellesLignesFrais($idVisiteur, $mois);
26 -     }
27 -     break;
28 + case 'validerMajFraisForfait':{...9 lines }
37 + case 'validerCreationFrais':{...17 lines }
54 + case 'supprimerFrais':{...4 lines }
58 }
59 $lesFraisHorsForfait = $pdo->getLesFraisHorsForfait($idVisiteur, $mois);
60 $lesFraisForfait = $pdo->getLesFraisForfait($idVisiteur, $mois);
61 require 'vues/v_listeFraisForfait.php';
62 require 'vues/v_listeFraisHorsForfait.php';
```

Le système teste (ligne 10) si c'est la première fois que l'utilisateur accède à cette demande de saisie de frais –cf extension 2.a- et va chercher en base (lignes 45-46) les données concernant les frais forfaitisés et non forfaitisés afin d'afficher les deux vues demandées (lignes 47-48). Ici, ces affichages sont communs aux autres cases.

Les deux vues affichées sont ici :

Renseigner ma fiche de frais du mois 08-2017

Éléments forfaitisés

Forfait Etape

12

Frais Kilométrique

562

Nuitée Hôtel

6

Repas Restaurant

25

Ajouter

Effacer

Vue « Liste des frais au forfait »

Descriptif des éléments hors forfait			
Date	Libellé	Montant	
12/08/2017	Achat de fleurs	29.90	Supprimer ce frais
14/08/2017	Taxi	32.50	Supprimer ce frais

Nouvel élément hors forfait

Date (jj/mm/aaaa):

Libellé

Montant :

Vue « Liste des frais hors forfait »

Nous avons fait le choix de présenter deux vues distinctes – nous aurions pu bien sûr mettre ce code dans un seul fichier – pour éventuellement réutiliser une de ces vue dans un autre cas d’utilisation. Dans cette architecture, l’affichage des vues est provoqué par un ordre include (ou require) nomVue.

Pour respecter l’indépendance des couches (vue, modèle), le modèle (fichier php) retourne des tableaux :

```

169 public function getLesFraisForfait($idVisiteur, $mois)
170 {
171     $requetePrepare = PdoGSB::$monPdo->prepare(
172         'SELECT fraisforfait.id as idfrais, '
173         . 'fraisforfait.libelle as libelle, '
174         . 'lignefraisforfait.quantite as quantite '
175         . 'FROM lignefraisforfait '
176         . 'INNER JOIN fraisforfait '
177         . 'ON fraisforfait.id = lignefraisforfait.idfraisforfait '
178         . 'WHERE lignefraisforfait.idvisiteur = :unIdVisiteur '
179         . 'AND lignefraisforfait.mois = :unMois '
180         . 'ORDER BY lignefraisforfait.idfraisforfait'
181     );
182     $requetePrepare->bindParam(':unIdVisiteur', $idVisiteur, PDO::PARAM_STR);
183     $requetePrepare->bindParam(':unMois', $mois, PDO::PARAM_STR);
184     $requetePrepare->execute();
185     return $requetePrepare->fetchAll();
186 }

```

La vue construit le code HTML à partir du tableau retourné :

```

17 <div class="row">
18 <h2>Renseigner ma fiche de frais du mois
19 <?php echo $numMois . "-" . $numAnnee ?>
20 </h2>
21 <h3>Eléments forfaitisés</h3>
22 <div class="col-md-4">
23 <form method="post"
24     action="index.php?uc=gererFrais&action=validerMajFraisForfait"
25     role="form">
26 <fieldset>
27 <?php
28     foreach ($lesFraisForfait as $unFrais) {
29         $idFrais = $unFrais['idfrais'];
30         $libelle = htmlspecialchars($unFrais['libelle']);

```

```

31     $quantite = $unFrais['quantite']; ?>
32     <div class="form-group">
33         <label for="idFrais"><?php echo $libelle ?></label>
34         <input type="text" id="idFrais"
35             name="lesFrais[<?php echo $idFrais ?>]"
36             size="10" maxlength="5"
37             value="<?php echo $quantite ?>"
38             class="form-control">
39     </div>
40     <?php
41     }
42     ?>
43     <button class="btn btn-success" type="submit">Ajouter</button>
44     <button class="btn btn-danger" type="reset">Effacer</button>
45 </fieldset>
46 </form>
47 </div>
48 </div>

```

Fonctionnement de l'application

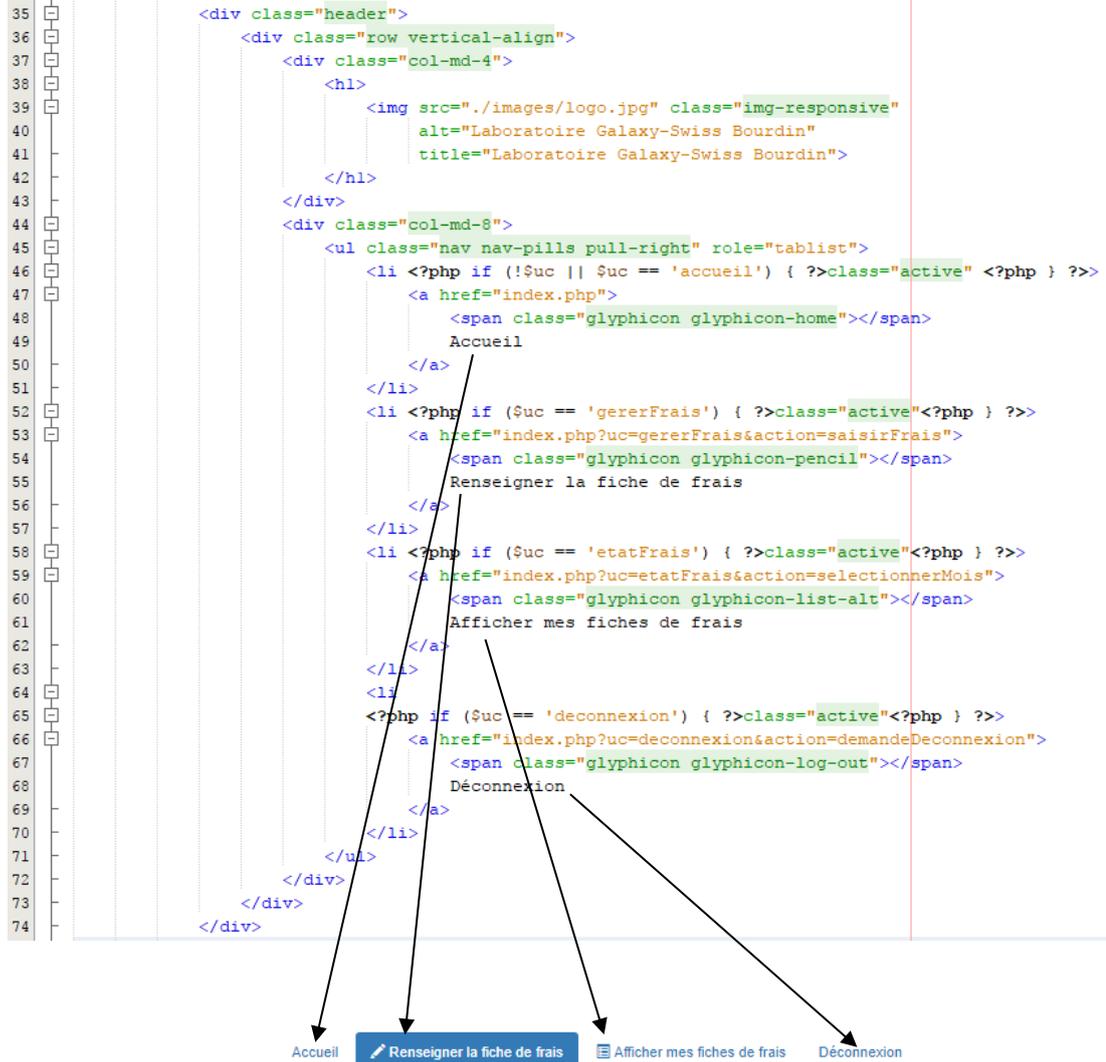
C'est la page index qui sert d'aiguilleur principal et oriente vers un contrôleur de cas d'utilisation :

```

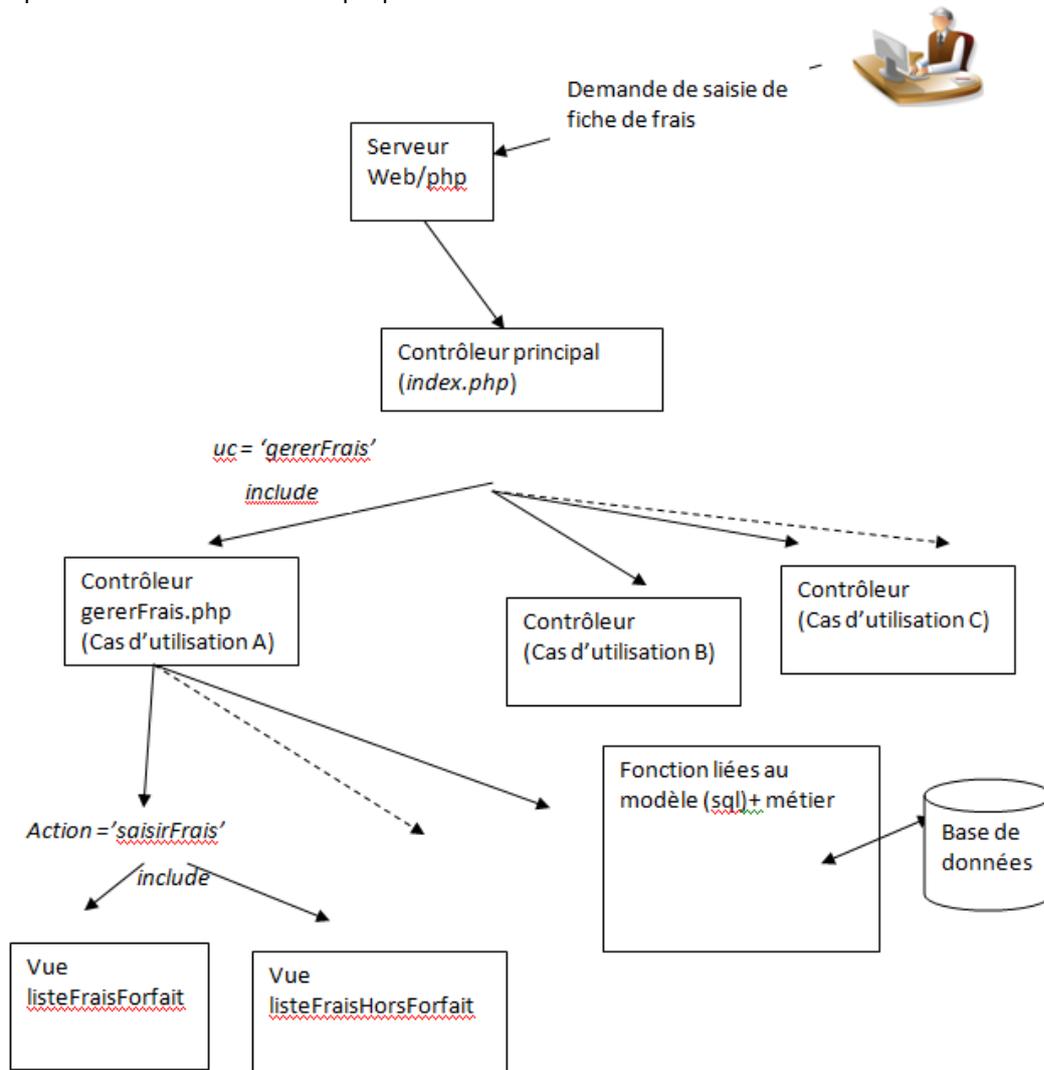
17     require_once 'includes/fct.inc.php';
18     require_once 'includes/class.pdogsb.inc.php';
19     session_start();
20     $pdo = PdoGsb::getPdoGsb();
21     $estConnecte = estConnecte();
22     require 'vues/v_entete.php';
23     $uc = filter_input(INPUT_GET, 'uc', FILTER_SANITIZE_STRING);
24     if ($uc && !$estConnecte) {
25         $uc = 'connexion';
26     } elseif (empty($uc)) {
27         $uc = 'accueil';
28     }
29     switch ($uc) {
30     case 'connexion':
31         include 'controleurs/c_connexion.php';
32         break;
33     case 'accueil':
34         include 'controleurs/c_accueil.php';
35         break;
36     case 'gererFrais':
37         include 'controleurs/c_gererFrais.php';
38         break;
39     case 'etatFrais':
40         include 'controleurs/c_etatFrais.php';
41         break;
42     case 'deconnexion':
43         include 'controleurs/c_deconnexion.php';
44         break;
45     }
46     require 'vues/v_pied.php';

```

Ceci est à associé à ce que l'utilisateur sélectionne dans le sommaire :



On peut résumer cette cinématique par un schéma :



Ainsi chaque page reçue est construite à partir de l'index comme une succession de fichiers include selon le cas d'utilisation. L'action demandée entraîne un traitement, à partir de la base de données et des règles métier (responsabilité de la couche Modèle) et expose les vues associées.

L'arborescence du site reflète cette architecture :

controleurs	23/08/2017 14:38	Dossier de fichiers
images	23/08/2017 14:38	Dossier de fichiers
includes	23/08/2017 14:38	Dossier de fichiers
styles	23/08/2017 14:38	Dossier de fichiers
tests	23/08/2017 14:38	Dossier de fichiers
vues	23/08/2017 14:38	Dossier de fichiers
index.php	23/08/2017 14:34	Fichier PHP

Le répertoire includes contient les fichiers utiles au modèle : accès à la base, fonctions métier, gestion des erreurs.

Utilisation d'une bibliothèque d'accès aux données

Php contient en standard une classe PDO, d'accès aux données ; elle a l'avantage d'être générique-indépendante du SGBD- et propose des services performants qui évitent la plupart du temps de faire soi-même les boucles de parcours des jeux d'enregistrements :

```
188 | □ | /**  
189 | |  * Retourne tous les id de la table FraisForfait  
190 | |  *  
191 | |  * @return un tableau associatif  
192 | |  */  
193 | | public function getLesIdFrais()  
194 | □ | {  
195 | |     $requetePrepare = PdoGsb::$monPdo->prepare(  
196 | |         'SELECT fraisforfait.id as idfrais '  
197 | |         . 'FROM fraisforfait ORDER BY fraisforfait.id'  
198 | |     );  
199 | |     $requetePrepare->execute();  
200 | |     return $requetePrepare->fetchAll();  
201 | | }
```

Cette classe n'est pas associée au modèle MVC, mais elle allège grandement la gestion des données.

Dans l'application GSB, la classe PDO est encapsulée dans une classe PdoGsb.

Règles de nommage côté scripts PHP

Outre les règles énoncées dans le document 2 "Normes de développement", viennent s'ajouter les règles de nommage suivantes :

Item	Règle de nommage ²
Variables	
jeu d'enregistrements	\$idJeu suivi du rôle
ligne du jeu d'enregistrements	tableau \$lg suivi du rôle
chaîne contenant une requête SQL	\$req ou \$requete
autre variable	pas de règle (le nom choisi doit toujours être porteur de sens par rapport au rôle de la variable)
Fonctions	
fonction retournant une requête SQL	obtenirReq suivi du rôle (exemple : obtenirReqEltsForfaitFicheFrais est le nom de la fonction qui retourne la requête permettant d'obtenir les données sur une fiche de frais)
fonction retournant une ligne ou une valeur	obtenir suivi du rôle (exemple : obtenirDetailFicheFrais est le nom de la fonction qui retourne la ligne correspondant à la fiche de frais demandée)
fonction d'ajout ou modification ou suppression	verbe ajouter ou modifier ou supprimer suivi du nom de la table. Si la fonction prend en charge plusieurs actions, on accole les noms des actions (exemple : ajouterFicheFrais)
fonction de vérification	son nom sera formé de estUn ou verifier ou existe suivi du rôle
Fichiers	
fichier contrôleur + affichage vue	Tout fichier contrôlant la cinématique des cas d'utilisation et la gestion de l'affichage commence par la lettre c (c comme contrôleur)

Règles de nommage côté base de données

Concernant le schéma de la base de données les règles d'écriture suivantes ont été appliquées :

- pas de blanc ni de caractère accentué dans les noms de table ou d'attribut ;
- chaque nom de table commence par une majuscule et est suivi de minuscules. S'il est composé de deux mots, ils sont collés et distingués par une majuscule ;
- chaque nom d'attribut est écrit en minuscule. S'il est composé de deux mots, ils sont collés et distingués par une majuscule. Le nom choisi pour l'attribut représente le rôle de son domaine dans la table ;
- une clef étrangère porte un nom significatif de son rôle dans la table.

² Tous les noms respectent la règle « Camel » qui est notamment utilisée pour la programmation en langage Java.

Structure de chaque page de l'application

Toutes les pages contrôleur d'un cas ou sous-cas d'utilisation sont construites selon cette structure.

```
<?
$repInclude = "./include/";
require($repInclude . "_init.inc.php");

require($repInclude . "_entete.inc.html");
require($repInclude . "_sommaire.inc.php");
?>
<!-- Division pour le contenu principal -->
<div id="contenu">
  <h2>Mes fiches de frais</h2>
  ...
</div>
<?php
require($repInclude . "_pied.inc.html");

require($repInclude . "_fin.inc.php");
?>
```

Le fichier `_init.inc.php` (voir ci-dessous) contient toutes les initialisations de variables (identifiant de connexion au serveur MySQL, tableau des erreurs).

L'en-tête (titre et barre de menus) est affiché grâce à l'exécution du code contenu dans les fichiers `_entete.inc.html` et `_sommaire.inc.php`. Dans ce fichier se trouve également la déclaration de la feuille de style `styles.css`.

Le pied de page est affiché grâce à l'exécution du code contenu dans le fichier `pied.inc.html`.

Le fichier `_fin.inc.php` libère les ressources (identifiant de connexion au serveur MySQL).

```
Fichier _init.inc.php
<?php
require("_bdGestionDonnees.lib.php");

require("_utilitairesEtGestionErreurs.lib.php");

// initialement, aucune erreur ...
$tabErreurs = array();

// établissement d'une connexion avec le serveur de données
// puis sélection de la BD qui contient les données des anciens
$idConnexion=connecterServeurBD();

if (!$idConnexion) {
    ajouterErreur($tabErreurs, "Echec de la connexion au serveur MySQL");
}

elseif (!activerBD($idConnexion)) {
    ajouterErreur($tabErreurs, "La base de données gsb_frais est inexistante ou non accessible");
}
?>
```

Fonctions pour la gestion des données.

Fonctions utilitaires et de gestion des erreurs.

Création d'un tableau vide destiné à recevoir les messages d'erreur.

Appel de la fonction de connexion au serveur MySQL.

Appel de la fonction de sélection de la base de données gsb frais.

Détail des choix pour la gestion des données et la gestion des erreurs

Gestion des données (fichier `_bdGestionDonnees.lib.php`)

Les interrogations de la base retournant une seule ligne sont entièrement prises en charge dans une fonction déportée ; cette fonction retourne alors le résultat dans un tableau (si plusieurs colonnes ont été demandées) ou dans une variable élémentaire.

Exemples :

- `obtenirDetailUtilisateur($idCnx, $unId)` : retourne un tableau contenant les données de l'utilisateur d'id `$unId`.
- `obtenirDernierMoisSaisi($idCnx, $unIdVisiteur)` : retourne une chaîne correspondant au mois (forme AAAAMM) de la dernière fiche de frais du visiteur d'id `$unIdVisiteur`.

Les interrogations de la base pouvant retourner plus d'un enregistrement sont traitées ainsi :

- constitution de la requête dans une fonction,
- exécution de la requête et traitement du jeu d'enregistrements dans le code de la page appelante.

Exemple : `obtenirReqLibellesFraisForfait`

Appel à la fonction pour constituer la requête :

```
$req=obtenirReqEltsForfaitFicheFrais();  
// obtenirReqEltsForfaitFicheFrais est la fonction qui constitue le texte  
// de la requête permettant d'obtenir la liste des éléments forfaitisés
```

Exécution de la requête :

```
$idJeuEltsFraisForfait = mysql_query($req,$idConnexion);
```

Traitement du jeu d'enregistrements :

```
$lgEltForfait = mysql_fetch_assoc($idJeuFraisForfait);  
while ( is_array($lgEltForfait) ) {  
    . . .  
    $lgEltForfait = mysql_fetch_assoc($idJeuFraisForfait);  
}  
mysql_free_result($idJeuFraisForfait);
```

Les mises à jour au sens large (modification, insertion, suppression) sont entièrement réalisées dans des fonctions.

Exemple : `ajouterLigneHorsForfait(...)`

Gestion des erreurs (fichier `_utilitairesEtGestionErreurs.lib.php`)

Principes de la bibliothèque de fonctions de gestion des erreurs

Par convention dans cette application, lorsqu'une erreur est détectée, un message d'erreur approprié est construit et fourni au système de gestion des erreurs. Ceci est simplifié par l'utilisation de la fonction `ajouterErreur`.

```
function ajouterErreur(&$tabErr, $msg) {  
    $tabErr[count($tabErr)]=$msg;  
}
```

`$tabErr` est le paramètre formel correspondant au tableau destiné à recevoir les différents messages d'erreur. Il est ici passé par référence car la fonction `ajouterErreur` doit modifier le contenu du tableau en y ajoutant un message dans le tableau.

Une fonction nbErreurs a été écrite pour retourner le nombre d'erreurs ; cela permet de tester le nombre d'erreurs avant d'appeler la fonction d'affichage des erreurs.

```
function nbErreurs($tabErr) {
    return count($tabErr);
}
```

La fonction d'affichage des erreurs parcourt le tableau des erreurs et les affiche les unes sous les autres.

```
function afficherErreurs($tabErr) {
    echo '<div class="erreur">';
    echo '<ul>';
    foreach($tabErr as $erreur) {
        echo "<li>$erreur</li>";
    }
    echo '</ul>';
    echo '</div>';
}
```

Principes d'utilisation des fonctions de gestion d'erreurs

Nous illustrons ces principes grâce aux contrôles effectués sur le formulaire de modification d'une fiche de frais.

```
// l'utilisateur valide les éléments forfaitisés
// vérification des quantités des éléments forfaitisés
$ok = verifierEntiersPositifs($tabQteEltsForfait);
if (!$ok) {
    ajouterErreur($tabErreurs, "Chaque quantité doit être renseignée
et numérique positive.");
}
else { // mise à jour des quantités des éléments forfaitisés
    modifierEltsForfait($idConnexion, $mois, obtenirIdUserCon-
necte(), $tabQteEltsForfait);
}

// si besoin, affichage des erreurs
if ( $etape == "validerSaisie" ) {
    if ( nbErreurs($tabErreurs) > 0 ) {
        echo toStringErreurs($tabErreurs);
    }
}
```

2. Normes de développement

Applications web écrites en PHP - Référence : GSB-STDWEBPHP - Version : 1.0

Introduction

Ce document s'appuie sur différentes sources de règles de codage, en particulier du projet communautaire **PEAR** - "**PHP Extension and Application Repository**"³ et du cadre **Zend Framework**⁴ qui fournissent entre autres des règles de codage pour les scripts PHP. Le document s'inspire aussi des règles de codage issues d'autres langages tels que Java.

Les règles énoncées par le présent document comportent au minimum :

- une **description** concise de la règle,

Si nécessaire :

- des **compléments** par rapport à la description,
- des **exemples** illustrant la règle, et éventuellement des **exceptions**,
- une partie **intérêts** en regard des critères qualité.

NB : La version actuelle des règles de codage ne comporte pas de règles sur les notions de POO (classe, niveau d'accès, membres d'instance ou de classe, etc.).

Fichiers

Ce paragraphe a pour but de décrire l'organisation et la présentation des fichiers mis en jeu dans un site Web dynamique écrit en PHP.

Extension des fichiers

Description :

Les fichiers PHP doivent obligatoirement se terminer par l'extension **.php** pour une question de sécurité. En procédant ainsi, il n'est pas possible de visualiser le source des fichiers PHP (qui contiennent peut-être des mots de passe), le serveur web les fait interpréter par PHP.

Les fichiers qui ne constituent pas des pages autonomes (des fichiers destinés à être inclus dans d'autres pages web) se terminent par l'extension **.inc.php**.

Les fichiers contenant uniquement des définitions de fonctions se terminent par l'extension **.lib.php**.

Un fichier contenant une classe se nommera **class.<nom de la classe>.inc.php**

Les fichiers contenant des pages statiques (sans code PHP) doivent porter l'extension **.html**.

Nom des fichiers

Description :

Seuls les caractères alphanumériques, tirets bas et tirets demi-cadratin ("-") sont autorisés. Les espaces et les caractères spéciaux sont interdits.

³ <http://pear.php.net/manual/fr/standards.php>

⁴ <http://framework.zend.com/manual/fr/coding-standard.html>

Format des fichiers

Description :

Tout fichier .php ou page .html doit :

Etre stocké comme du texte ASCII

Utiliser le jeu de caractères UTF-8

Etre formaté Dos

Compléments :

Le << formatage Dos >> signifie que les lignes doivent finir par les combinaisons de retour chariot / retour à la ligne (CRLF), contrairement au << formatage Unix >> qui attend uniquement un retour à la ligne (LF). Un retour à la ligne est représenté par l'ordinal 10, l'octal 012 et l'hexa 0A. Un retour chariot est représenté par l'ordinal 13, l'octal 015 et l'hexa 0D.

Préambule XML

Description :

Les pages Web doivent se conformer à une des normes HTML ou XHTML.

Toute page Web devra donc débiter par la directive <!DOCTYPE précisant quelle norme est suivie.

Elles seront validées à l'aide du validateur en ligne <http://validator.w3.org>

Exemple :

Pour exemple, voici l'en-tête d'un fichier XHTML 1.0 :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Inclusion de scripts

L'inclusion de scripts peut être réalisée par plusieurs instructions prédéfinies en PHP : include, require, include_once, require_once. Toutes ont pour objectif de provoquer leur propre remplacement par le fichier spécifié, un peu comme les commandes de préprocesseur C #include.

Les instructions include et require sont identiques, hormis dans leur gestion des erreurs. include produit une alerte (warning) tandis que require génère une erreur fatale . En d'autres termes, lorsque le fichier à inclure n'existe pas, le script est interrompu. include ne se comporte pas de cette façon, et le script continuera son exécution.

La différence entre require et require_once (idem entre include et include_once) est qu'avec **require_once()**, on est assuré que le code du fichier inclus ne sera ajouté qu'une seule fois, évitant de ce fait les redéfinitions de variables ou de fonctions, génératrices d'alertes.

Description : L'inclusion de scripts sera réalisée à l'aide de l'instruction require_once lorsque les fichiers inclus contiennent des bibliothèques, require sinon.

Exemple :

Script prem.inc.php

```
<?php
    fonction uneFonction () {
        echo "Fonction définie dans prem.inc.php <br />";
    }
?>
```

Script second.inc.php

```
<?php
    require_once("prem.inc.php");
    uneFonction();
    echo "Pas de problème : uneFonction n'a pas été redéfinie 2 fois.
    Merci require_once ! <br />";
?>
```

Script monscript.php

```
<?php
    require_once(prem.inc.php);
    require_once(second.inc.php);
    echo "Tout va bien ! <br />";
?>
```

Présentation du code

Tag PHP

Description :

Toujours utiliser `<?php ?>` pour délimiter du code PHP, et non la version abrégée `<? ?>`. C'est la méthode la plus portable pour inclure du code PHP sur les différents systèmes d'exploitation et les différentes configurations.

Intérêts :

Portabilité

Séparation PHP/ HTML

Description :

Les balises HTML doivent se situer au maximum dans les sections HTML et non incluses à l'intérieur du texte des messages de l'instruction d'affichage echo.

Exemples :

Ne pas écrire

```
<?php
    echo "<select id=\"lstAnnee\" name=\"lstAnnee\">";
    $anCours = date("Y");
    for ( $an = $anCours - 5 ; $an <= $anCours + 5 ; $an++ ) {
        echo "<option value=\"\" . $an . \">\" . $an . "</option>";
    }
    echo "</select>";
?>
</select>
```

Mais écrire

```
<select id="lstAnnee" name="lstAnnee">
<?php
    $anCours = date("Y");
    for ( $an = $anCours - 5 ; $an <= $anCours + 5 ; $an++ ) {
?>
        <option value="<?php echo $an ; ?>">echo $an; ?</option>
<?php
    }
?>
</select>
```

Intérêts :

Dans les sections HTML, l'éditeur de l'outil de développement applique la coloration syntaxique sur les balises et attributs HTML. Ceci accroît donc la lisibilité et la localisation des erreurs de syntaxe au niveau du langage HTML. Il est aussi plus aisé d'intervenir uniquement sur la présentation, sans effet de bord sur la partie dynamique.

Maintenabilité : lisibilité

Portabilité : indépendance

Caractères et lignes

Description :

Chaque ligne doit comporter au plus une instruction.

Les caractères accentués ne doivent pas être utilisés dans le code source, excepté dans les commentaires et les messages texte.

Un fichier source ne devrait pas dépasser plus de **500 lignes**.

Exemples :

Ne pas écrire

```
$i-- ; $j++ ;
```

Mais écrire

```
$i-- ;  
$j++ ;
```

Intérêts :

Maintenabilité : lisibilité et clarté du code.

Indentation et longueur des lignes

Description :

Le pas d'indentation doit être **fixe** et correspondre à **4 caractères**. Ce pas d'indentation doit être paramétré dans l'éditeur de l'environnement de développement. L'indentation est stockée dans le fichier sous forme de 4 caractères espace (sans tabulation réelle).

Il est recommandé que la longueur des lignes ne dépasse pas 75 à 85 caractères.

Lorsqu'une ligne d'instruction est trop longue, elle doit être coupée après une virgule ou avant un opérateur. On alignera la nouvelle ligne avec le début de l'expression de même niveau de la ligne précédente.

Exemples :

Exemple 1 : découpage d'un appel de fonction

On découpe la ligne après une virgule :

```
$maVar = fonctionA(expressionLongue1, expressionLongue2,  
                    fonctionB(expressionLongue3,  
                               expressionLongue4)) ;
```

Exemple 2 : découpage d'une expression arithmétique

La ligne est découpée avant un opérateur :

```
$maVar = expressionLongue1 * expressionLongue2  
        + expressionLongue3 - expressionLongue4
```

Exemple 3 : découpage d'une expression conditionnelle

La ligne est découpée avant un opérateur :

```
if(((condition1 && condition2) || (condition3 && condition4))  
    && !(condition5 && condition6)) {  
    doSomething();  
}
```

Intérêts :

Maintenabilité : lisibilité.

Espacement dans les instructions

Description :

1. Un mot-clé suivi d'une parenthèse ouvrante doit être séparé de celle-ci par un espace. Ce n'est pas le cas entre un identificateur de fonction et la parenthèse ouvrante.
2. Tous les opérateurs binaires, sauf l'opérateur « -> » doivent être séparés de leurs opérandes par un espace.
3. Les opérateurs unaires doivent être accolés à leur opérande.

Exemples :

1.

```
while (true) {  
    ...  
}
```
2.

```
$totalTTC = $totalHT + ($totalHT * ($tauxTVA / 100));  
$totalTTC = round($totalTTC, 2);  
$existe = $unElt->hasAttribute();
```
3.

```
$nb = 0;  
$fin = false;  
while (!$fin) {  
    ... $nb++;  
}
```
4.

```
for ($nbLignes = 1; $nbLignes < 4; $nbLignes++) {  
}
```

Intérêts :

Maintenabilité : lisibilité

Présentation des blocs logiques

Description :

1. Chaque bloc logique doit être délimité par des accolades, même s'il ne comporte qu'une seule instruction (cf. exemple 1),
2. Dans une instruction avec bloc, l'accolade ouvrante doit se trouver sur la fin de la ligne de l'instruction ; l'accolade fermante doit débiter une ligne, et se situer au même niveau d'indentation que l'instruction dont elle ferme le bloc (cf. exemple 2),
3. Les instructions contenues dans un bloc ont un niveau supérieur d'indentation.

Exemples :

Exemple 1 :

Ne pas écrire

```
if ($prime > 2000)  
    $prime = 2000;
```

Mais écrire

```
if ($prime > 2000) {  
    $prime = 2000;  
}
```

Exemple 2 : écriture des instructions avec blocs :

Structures de contrôle conditionnelles

```
if (...) {
    ...
} elseif (...) {
    ...
} else {
    ...
}
switch (...) {
    case ... :
        ...
    case ... :
        ...
    default :
        ...
}
```

Définition de fonction

```
function uneFonction() {
    ...
}
```

Structures de contrôle itératives

```
for (...; ...; ...) {
    ...
}
while (...) {
    ...
}
do {
    ...
}while (...);
```

Intérêts :

La présence d'accolades ainsi que l'indentation facilitent la localisation des débuts et fins de blocs et réduit le risque d'erreur logique lors de l'ajout de nouvelles lignes de code.

Maintenabilité : lisibilité.

Appels de fonctions / méthodes

Description :

Les fonctions doivent être appelées sans espace entre le nom de la fonction, la parenthèse ouvrante, et le premier paramètre ; avec un espace entre la virgule et chaque paramètre et aucun espace entre le dernier paramètre, la parenthèse fermante et le point virgule.

Il doit y avoir un espace de chaque côté du signe égal utilisé pour affecter la valeur de retour de la fonction à une variable. Dans le cas d'un bloc d'instructions similaires, des espaces supplémentaires peuvent être ajoutés pour améliorer la lisibilité.

Exemples :

```
<?php
$total = round($total, 2);
?>

<?php
$courte          = abs($courte);
$longueVariable = abs($longueVariable);
?>
```

Intérêts :

Maintenabilité : lisibilité

Définition de fonctions

Description :

Les fonctions définies à usage exclusif d'un script seront définies en début du script.

La déclaration des fonctions respecte l'indentation classique des accolades.

Les arguments possédant des valeurs par défaut vont à la fin de la liste des arguments.

Exemples :

```
<?php
function maFonction($arg1, $arg2 = '') {
    if (condition) {
        statement;
    }
    return $val;
}
?>
```

Documentations et commentaires

Introduction

La documentation est essentielle à la compréhension des fonctionnalités du code. Elle peut être intégrée directement au code source, tout en restant aisément extractible dans un format de sortie tel que HTML ou PDF. Cette intégration favorise la cohérence entre documentation et code source, facilite l'accès à la documentation, permet la distribution d'un code source auto-documenté. Elle rend donc plus aisée la maintenance du projet.

L'intégration de la documentation se fait à travers une extension des commentaires autorisés par le langage PHP. Nous utiliserons celle proposée par l'outil PHPDocumentor, dont les spécifications sont disponibles à l'URL <https://www.phpdoc.org/>.

Pour rappel, les commentaires autorisés par le langage PHP adoptent une syntaxe similaire aux langages C et C++ (`/* ... */` et `//`). Ils servent à décrire en langage naturel tout élément du code source.

L'extension de l'outil PHPDocumentor est la suivante `/** ... */`. Leur utilisation permettra de produire une documentation dans un ou plusieurs formats de sortie tels que HTML, XML, PDF ou CHM.

La syntaxe spécifique à l'outil PHPDocumentor sera utilisée au minimum pour les entêtes de fichiers source et les entêtes de fonctions. Des éléments sur l'installation, les spécifications et l'utilisation de l'outil PHPDocumentor sont fournis en annexe 1.

Entêtes de fichier source

Description :

Chaque fichier qui contient du code PHP doit avoir un bloc d'entête en haut du fichier qui contient au minimum les balises phpDocumentor ci-dessous.

Compléments :

Format d'entête de fichier source :

```
<?php
/**
 * Description courte des fonctionnalités du fichier

 * Description longue des fonctionnalités du fichier si nécessaire
 * @author nom de l'auteur
 * @package default
 */
```

Entêtes de fonction

Description :

Toute définition de fonction doit être précédée du bloc de documentation contenant au minimum :

- Une description de la fonction
- Tous les arguments
- Toutes les valeurs de retour possibles

Compléments :

Format d'entête de fonction :

```
/**
 * Description courte de la fonction.

 * Description longue de la fonction (si besoin)
 * @param type nomParam1 description
 * ...
 * @param type nomParamn description
 * @return type description
 */
function uneFonction($nomParam1, ...) {
```

Exemple :

```
/**
 * Fournit le compte utilisateur d'une adresse email.

 * Retourne le compte utilisateur (partie identifiant de la
 * personne) de l'adresse email $email, c'ad la partie de l'adresse
 * située avant le caractère @ rencontré dans la chaîne $email.
 * Retourne l'adresse complète si pas de @ dans $email.
 * @email string adresse email
 * @return string compte utilisateur
 */
function extraitCompteDeEmail ($email) {
    ...
}
```

Commentaires des instructions du code

Description :

Il existe deux types de commentaires :

1. les commentaires mono ligne qui inactivent tout ce qui apparaît à la suite, sur la même ligne :
//
2. les commentaires multi-lignes qui inactivent tout ce qui se trouve entre les deux délimiteurs, que ce soit sur une seule ligne ou sur plusieurs /* */

Il est important de ne réserver les commentaires multi-lignes qu'aux blocs utiles à PHPDocumentor et à l'inactivation de portions de code.

Les commentaires mono-ligne permettant de commenter le reste, à savoir, toute information de documentation interne relative aux lignes de code. Ceci afin d'éviter des erreurs de compilation dues aux imbrications des commentaires multi-lignes.

Exemples :

1. Insertion d'un commentaire mono-ligne pour expliquer le comportement d'un code

Exemple 1 :

```
function extraitCompteDeEmail ($email) {
    // le traitement se fait en 2 temps : recherche de la position $pos dans
    // l'adresse de l'occurrence du caractère @, puis si @ présent,
    // extraction du morceau de chaîne du 1er caractère sur $pos caractères
    $pos = strpos($email, "@");
    if ( is_integer($pos) ) {
        $res = substr($email, 0, $pos);
    }
    else {
        $res = $email;
    }
    return $res;
}
```

Exemple 2 :

```
// page inaccessible si visiteur non connecté
if (!estVisiteurConnecte()) {
    header("Location: cSeConnecter.php");
}

// acquisition des données reçues par la méthode post
$mois = lireDonneePost("1stMois", "");
$etape = lireDonneePost("etape", "");

Inactivation d'une portion de code pour débogage
/*
// page inaccessible si visiteur non connecté
if (!estVisiteurConnecte()) {
    header("Location: cSeConnecter.php");
}
*/
```

Nommage des identificateurs

Cette convention concerne les éléments suivants du langage :

- les fonctions,
- les paramètres formels de fonctions,
- les constantes,
- les variables globales à un script,
- les variables locales,
- les variables de session.

Pour l'ensemble de ces éléments, la clarté des identificateurs est conseillée. Le nom attribué aux différents éléments doit être aussi explicite que possible, c'est un gage de compréhension du code.

Nommage des fonctions

Description :

L'identificateur d'une fonction est un verbe, ou groupe verbal.

Les noms de fonctions ne peuvent contenir que des caractères alphanumériques. Les tirets bas ("_") ne sont pas permis. Les nombres sont autorisés mais déconseillés.

Les noms de fonctions doivent toujours commencer avec une lettre en minuscule. Quand un nom de fonction est composé de plus d'un seul mot, la première lettre de chaque mot doit être mise en majuscule. C'est ce que l'on appelle communément la "notationCamel".

Exemples :

```
filtrerChaineBD(), verifierInfosConnexion(), estEntier()
```

Intérêts :

Maintenabilité : lisibilité.

Nommage des constantes

Description :

Les constantes doivent être déclarées grâce à la commande **define()** en utilisant un nom réellement significatif. Les constantes peuvent contenir des caractères alphanumériques et des tirets bas. Les nombres sont autorisés.

Les constantes doivent toujours être en majuscules, les mots séparés par des '_'.

On limitera l'utilisation des constantes littérales (nombre ou chaîne de caractères) dans les traitements.

Exemples :

```
define("TYPE_USER_ADMIN", "ADM")  
// définit la constante de nom TYPE_USER_ADMIN et de valeur ADM
```

Exceptions :

Les constantes numériques -1, 0, 1 peuvent toutefois être utilisées dans le code.

Intérêts :

Maintenabilité : lisibilité.

Nommage des variables et paramètres

Description :

L'identificateur d'une variable ou paramètre indique le rôle joué dans le code ; c'est en général un nom, ou groupe nominal. Il faut éviter de faire jouer à une variable plusieurs rôles.

Les noms de variables et paramètres ne peuvent contenir que des caractères alphanumériques. Les tirets bas sont autorisés uniquement pour les membres privés d'une classe. Les nombres sont autorisés mais déconseillés.

Comme les identificateurs de fonctions, les noms de variables et paramètres adoptent la notation Camel.

Exemples :

```
$nomEtud, $login
```

Intérêts :

Maintenabilité : lisibilité.

Algorithmique

Fonctions/méthodes

Modularité

Description :

Le codage doit être réalisé en recherchant le plus possible la modularité :

- chaque fonction doit réaliser un et un seul traitement,
- chaque fonction doit être construite de manière à posséder la plus forte cohésion et la plus grande indépendance possible par rapport à son environnement.

Intérêts :

Maintenabilité et fiabilité : modularité.

Nombre de paramètres des fonctions

Description :

Les fonctions ne doivent pas comporter un trop grand nombre de paramètres. La limite de 5 à 6 paramètres est recommandée. Tout dépassement de cette limite doit être justifié.

Compléments :

Cette règle s'applique, tout spécialement, dans le cadre de la programmation par objets qui permet justement de réduire le nombre de paramètres des fonctions.

Intérêts :

Maintenabilité : lisibilité.

Instructions

Écriture des instructions d'affectation

Description :

Il faut utiliser dès que possible les formes abrégées des instructions d'affectation.

Compléments :

Les instructions d'affectation du type :

$$\tilde{A} = A \langle op \rangle \langle exp \rangle ;$$

peuvent être notées sous leur forme abrégée :

$$\tilde{A} \langle op \rangle = \langle exp \rangle ;$$

Exemples :

Écrire

```
$total *= 0.90;
```

au lieu de

```
$total = $total * 0.90;
```

Parenthésage des expressions

Description :

Il est recommandé d'utiliser les parenthèses à chaque fois qu'une expression peut prêter à confusion.

Exemples :

Il ne faut pas écrire ...

```
if ($nbLignes == 0 && $nbMots == 0)
```

...mais plutôt écrire

```
if (($nbLignes == 0) && ($nbMots == 0))
```

Intérêts :

L'ajout de parenthèses dans les expressions comportant plusieurs opérateurs permet d'éviter des confusions sur leur priorité.

Maintenabilité : lisibilité.

Interdiction des instructions imbriquées

Description :

Les instructions imbriquées doivent être évitées quand cela est possible. En particulier, les types d'instructions imbriquées suivantes sont à bannir :

- affectations dans les conditions, dans les appels de fonctions et dans les expressions ;
- affectations multiples.

Compléments :

Une expression ne doit donc contenir que :

- des variables,
- des constantes,
- des appels de fonctions dont les arguments ne sont pas eux-mêmes des éléments variables.

Exemples :

Éviter les affectations dans les conditions :

```
while ($ligne = mysql_fetch_assoc($idJeu))
  if ($nb++ != 20)
```

Éviter les affectations dans les appels de fonctions :

```
uneFonction($nb = rand(10,20), $qte);
```

Éviter les affectations dans les expressions :

```
$a = ($b = $c--) + $d;
```

Éviter les affectations multiples :

```
$a = $b = $c = $i++;
```

Intérêts :

La complexité des expressions peut donner lieu à des erreurs d'interprétation. Par exemple, l'affectation dans une condition peut être lue comme un test d'égalité.

Maintenabilité : lisibilité.

Limitation de l'utilisation des break et continue dans les itératives

Description :

Utilisation modérée

Les ruptures de séquence `break` et `continue` doivent être utilisées avec modération dans les itératives.

Compléments :

L'abus de ce type d'instructions peut rendre le code difficile à comprendre. Elles pourront toutefois être utilisées ponctuellement. Dans ce cas, un commentaire devra le signaler.

Intérêts :

Limiter les instructions `break` et `continue` améliore la structuration du code. Ces instructions (qui sont des "goto" déguisés), lorsqu'elles sont utilisées fréquemment, peuvent en effet dénoter une mauvaise analyse des conditions d'itérations dans certains cas.

Écriture des switch

Description :

1. Tout le contenu à l'intérieur de l'instruction "switch" doit être indenté avec 4 espaces. Le contenu sous chaque "case" doit être indenté avec encore 4 espaces supplémentaires.
2. Les structures `switch` doivent obligatoirement comporter une clause `default`.
3. Le niveau d'imbrication des `switch` ne doit pas dépasser 2.
4. Chaque cas ou groupe de cas doit se terminer normalement par une instruction `break`. Les cas ne se terminant par un saut **break** doivent spécifier un commentaire rappelant que l'exécution se poursuit.
5. L'instruction `break` est obligatoire à la fin du cas par défaut. Cela est redondant mais protège d'une erreur en cas de rajout d'autres cas par la suite.

Compléments :

```
switch (choix) {
    case expression1 :
        instructions
        /* pas de break */
    case expression2 :
    case expression3 :
        instructions
        break;
    default :
        instructions;
        break;
}
```

Intérêts :

Fiabilité : robustesse, clarté.

Utilisation de l'opérateur ternaire conditionnel

Description :

Il faut éviter d'utiliser l'abréviation « ? : » du « `if ... else` », sauf si les conditions suivantes sont réunies (cf. exemple) :

- la valeur de l'expression conditionnelle est effectivement utilisée (dans un retour ou un appel de fonction, une affectation, etc.),
- les 3 opérandes ne sont pas trop complexes.

Si toutefois on utilise cet opérateur, il faut mettre la condition (placée avant le « ? ») entre parenthèses.

Compléments :

Exemple

Le cas suivant...

```
if ($a > $b) {
    $maxi = $a;
} else {
    $maxi = $b;
}
```

...se prête bien à l'utilisation de l'opérateur conditionnel ternaire

```
$maxi = ($a > $b) ? $a : $b;
```

En effet, on utilise la valeur de l'expression conditionnelle et les opérandes ne sont pas trop complexes.

Intérêts : Maintenabilité : lisibilité.

Gestion des formulaires HTML

Nommage des formulaires et des champs de formulaires

Description :

Les noms des éléments HTML débuteront par un préfixe rappelant leur type.

Les préfixes retenus concernent les formulaires et les champs contenus dans les formulaires.

Type d'élément	Préfixe
Formulaire	frm
Zone de texte mono_ligne (text, password) / multi_lignes	txt
Champ caché	hd
Bouton d'option (bouton radio)	opt
Case à cocher	chk
Zone de liste	lst
Bouton de type reset	br
Bouton de type button	bt
Bouton de type submit	cmd

Méthodes de soumission des formulaires

Les méthodes de soumission d'un formulaire sont au nombre de 2 : GET et POST. La première véhicule les noms et valeurs des champs dans l'URL de la requête HTTP, la seconde dans le corps de la requête HTTP.

Description :

La méthode POST est à préférer pour des raisons de taille de données et de confidentialité. À noter que la confidentialité se résume ici à ne pas voir apparaître les noms et valeurs de champs dans la zone d'adresse du navigateur : les données sont, dans les 2 cas, transmises en clair sur le réseau dans le cas où le protocole applicatif utilisé reste HTTP.

Le choix de la méthode GET peut cependant se justifier s'il est souhaitable de pouvoir conserver les différentes soumissions d'un formulaire en favoris.

Variables superglobales \$_GET, \$_POST

Les valeurs saisies dans un formulaire sont mises à disposition des scripts PHP dans les tableaux associatifs superglobaux \$_GET et \$_POST. Le tableau \$_GET contient également les valeurs transmises via la constitution d'un lien.

Description :

Au cours de la mise au point des scripts, il est recommandé d'appeler la fonction var_dump sur ces tableaux \$_GET et \$_POST afin d'apprécier réellement les données (nom et valeur) reçues.

De plus, pour éviter de se référer dans tout le script soit au tableau \$_GET, soit au tableau \$_POST, tout script PHP affectera initialement les éléments des deux tableaux dans des variables. Ceci permettra également de migrer facilement d'une méthode de soumission POST vers GET ou vice-versa.

On pourra définir des fonctions spécialisées afin de récupérer les valeurs des éléments à partir d'un tableau ou d'un autre, en prévoyant des valeurs par défaut en cas d'inexistence d'un élément.

Exemples :

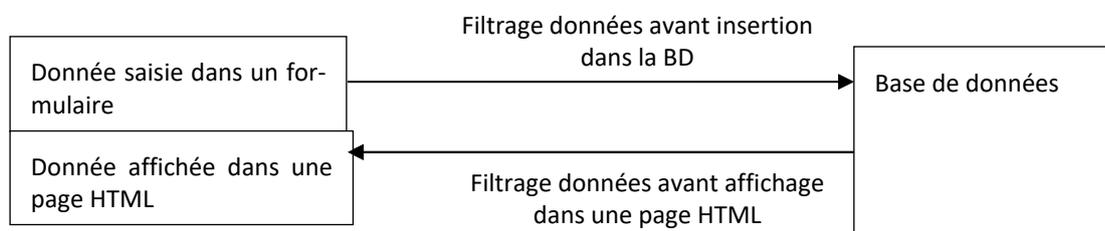
```
// acquisition des données reçues par la méthode post
$mois = $_POST["1stMois"];
$etape = $_POST["etape"];
// à commenter après débogage
var_dump($_GET, $_POST);
```

Éléments de sécurité sur la protection des données

Introduction

Toute donnée saisie à l'aide d'un formulaire peut engendrer des dysfonctionnements, que ce soit lors de l'enregistrement dans la base de données ou lors de son affichage ultérieur dans une page HTML.

Ces dysfonctionnements sont dus à l'interprétation de certains caractères dits spéciaux, soit par le SGBDR, soit par le navigateur. Il est donc nécessaire d'annuler les effets de ces caractères spéciaux en traitant les données d'une part avant de les insérer dans la base de données et d'autre part, avant de les restituer dans une page HTML.



Protection des données avant insertion dans la base de données

Comment protéger les données ?

Certains caractères spéciaux ont une signification précise pour le SGBDR. Il en est ainsi du guillemet simple ' qui a pour rôle de délimiter une valeur chaîne au sein d'une requête SQL.

Il n'est pourtant pas exclu que ce caractère se trouve dans une valeur chaîne, en particulier dans les noms de famille et surtout dans des champs commentaires. Le rôle du guillemet simple doit donc être annulé avant d'être inclus dans une requête SQL, sous peine de provoquer un refus d'exécution de la part du moteur SGBDR.

D'autre part, l'absence de traitement de ce caractère spécial laisse la porte ouverte à des attaques par injection SQL.

Toute valeur alphanumérique à enregistrer dans la base doit faire l'objet d'un traitement des caractères spéciaux. Ce traitement consiste à ajouter le caractère d'échappement \ devant chaque caractère spécial pour imposer que le caractère doit être interprété comme un caractère normal.

Certaines fonctions PHP prédéfinies tq *addslashes* ou *mysql_real_escape_string* prennent en charge ce traitement.

Il faut aussi noter que la directive PHP *magic_quotes_gpc* présente dans le fichier *php.ini* peut prendre les valeurs On ou Off : elle permet de gérer automatiquement ou non l'appel à la fonction *addslashes* sur toutes les données GET, POST et COOKIE. Il ne faut donc pas appeler la fonction *addslashes* sur des données déjà protégées avec la directive *magic_quotes_gpc* sinon les protections seront doublées.

La fonction *get_magic_quotes_gpc* est utile pour vérifier la valeur de la directive *magic_quotes_gpc*.

Afin d'être indépendant de cette directive de configuration (dont la valeur peut varier suivant les environnements), il est recommandé de définir une fonction utilitaire *filtrerChainePourBD* testant cette directive et échappant une chaîne si besoin.

Fabrication d'une requête SQL sécurisée

Description :

La fonction utilitaire `filtrerChainePourBD` doit être appelée sur toute valeur alphanumérique avant d'être insérée dans la base de données.

Exemple de fabrication d'une requête sécurisée :

```
<?php
    $query = "SELECT * FROM users";
    $query .= " WHERE user='" . filtrerChainePourBD($user) . "'";
    $query .= " AND password = " . filtrerChainePourBD($password) . "'";
?>
```

Protection des données d'une page HTML

Comment protéger les données d'une page HTML ?

Le langage HTML est fondé sur la notion de balises marquées par les caractères < et >. La valeur des attributs d'une balise est délimitée par des guillemets simples ou doubles.

C'est le rôle du navigateur d'interpréter ces balises pour générer la présentation attendue de la page. Les données elles-mêmes doivent donc être exemptes de ces caractères réservés pour éviter une interprétation erronée de la page.

Exemple 1 : La valeur du champ `txtComment` suivant :

```
<input type="text" name="txtComment" value="Bonjour "Dupont""/>
```

sera tronquée par le navigateur : la valeur initiale du champ `txtComment` sera Bonjour.

Exemple 2 : La valeur du champ `txtComment` suivant.

```
<input type="text" name="txtComment" value="<script> while (true)
alert('Erreur'); </script>"/>
```

sera interprétée comme une séquence de script et provoquera indéfiniment l'affichage d'une boîte message. Le navigateur lui-même devra être arrêté.

Pour éviter ces effets néfastes, ces caractères réservés doivent être traduits en symboles nommés HTML (aussi appelés entités HTML). Ainsi, le caractère < doit être transformé en `<`, > en `>`, etc.

La fonction PHP prédéfinie `htmlspecialchars` prend en charge ce traitement.

```
string htmlspecialchars ( string string , int quote_style ,
string charset )
```

Les remplacements effectués sont :

- " & " (et commercial) devient " `&` ; "
- " " " (guillemets doubles) devient " `"` ; " lorsque `ENT_NOQUOTES` n'est pas utilisé.
- " ' " (single quote) devient " `'` ; " uniquement lorsque `ENT_QUOTES` est utilisé.
- "<" (supérieur à) devient " `<` ; "
- ">" (inférieur à) devient " `>` ; "

Les guillemets simples et doubles ne sont pas systématiquement traduits : cela dépend de la valeur du paramètre optionnel `quote_style`.

`ENT_COMPAT`, la constante par défaut, va convertir les guillemets doubles, et ignorer les guillemets simples; `ENT_QUOTES` va convertir les guillemets doubles et les guillemets simples;

`ENT_NOQUOTES` va ignorer les guillemets doubles et les guillemets simples.

Exemple :

```
<?php
$new = htmlspecialchars("<a href='test'>Test</a>", ENT_QUOTES);
echo $new;
// <a href=&#039;test&#039;>Test</a>
?>
```

Fabrication d'une page HTML sécurisée

Description :

La fonction `htmlspecialchars` doit être appelée sur toute valeur en provenance de la base de données avant d'être affichée dans une page HTML. On l'appellera avec pour second argument la constante `ENT_QUOTES` pour convertir à la fois les guillemets doubles et simples.

Configuration du fichier php.ini

Description :

Afin de favoriser la détection des erreurs, ainsi que la portabilité d'une configuration PHP à une autre, il est obligatoire de réaliser le développement d'une application Web avec certaines directives de configuration PHP affectées aux valeurs recommandées suivantes.

Nom de la directive	Description	Valeur imposée
<code>short_open_tag</code>	Définit si les balises courtes d'ouverture de PHP (<code><? ?></code>) sont autorisées ou non.	Off
<code>output_buffering</code>	Définit l'activation ou non de la bufferisation de sortie. L'activation de la bufferisation peut être autorisée moyennant d'être dûment justifiée.	Off
<code>error_reporting</code>	<p>Fixe le niveau d'erreur. Ce paramètre est un entier, représentant un champ de bits. Cette directive est renseignée en utilisant les types d'erreurs définis sous forme de constantes et en utilisant les opérateurs bits à bits <code>&</code> (ET), <code> </code> (OU), <code>~</code> (SAUF), de même que l'opérateur booléen <code>!</code> (SAUF).</p> <p>Le rapport d'erreur de niveau E_NOTICE (inclus dans E_ALL) durant le développement a des avantages. En terme de débogage, les messages d'alertes signalent des bogues potentiels dans le code. Par exemple, l'utilisation de valeurs non initialisées est signalée.</p> <p>Comme E_STRICT, nouveau niveau d'erreur introduit en PHP5, n'est pas inclus sans E_ALL, il faut explicitement l'ajouter. Il permet d'être alerté de l'utilisation de fonctions non recommandées.</p>	E_ALL E_STRICT

Compléments :

Les directives de configuration PHP peuvent être appliquées à 3 niveaux :

- à l'ensemble des applications Web d'un serveur Web par le biais du fichier `php.ini`,
- à l'ensemble des scripts PHP d'un répertoire par le biais d'un fichier caché situé dans ce répertoire et analysé par le serveur Web (directive `php_flag` du fichier `.htaccess` analysé par Apache),
- à un seul script PHP par le biais de l'appel de la fonction `ini_set`.

Intérêts :

Portabilité : indépendance du logiciel par rapport à son environnement

Fiabilité : robustesse

Annexe 1 – Éléments sur l'outil PHPDocumentor⁵

PHP Documentor est un outil de génération automatique de documentation à partir des commentaires inclus dans les programmes PHP.

PHPDocumentor peut être utilisé soit via la ligne de commande, soit via une interface web.

Le téléchargement de l'archive de l'outil PHP Documentor se fait là : <https://www.phpdoc.org/>

Pour installer phpDocumentor, il faut décompresser l'archive dans un répertoire en respectant la structure interne des dossiers. L'utilisation de l'interface web implique de décompresser l'archive dans un dossier accessible par le serveur Web. Dans la suite du document, nous considérons que ce dossier se nomme *phpdoc*.

Les blocs de commentaires

Format d'un bloc de commentaires

La documentation exploitée par PHPDocumentor doit se trouver dans un bloc de commentaires respectant le format suivant :

```
/**
 * Mes explications ...
 */
```

Ce bloc de commentaires est un bloc de commentaires étendu qui commence par un "/*" et présente un "*" au début de chaque ligne. Les blocs de commentaires précèdent les éléments qu'ils documentent.

Toute ligne dans un bloc de commentaires qui ne commence pas par un « * » sera ignorée.

⁵ Référence site officiel : www.phpdoc.org

Contenu d'un bloc de commentaires

Un bloc de commentaires contient trois segments de base dans l'ordre suivant :

- une **description courte** : débute sur la première ligne, et peut se terminer par un point ou une ligne blanche.
- une **description longue** : peut occuper autant de lignes que nécessaire.
- des **marqueurs** : des mots préfixés par le caractère @. Ils informent phpDocumentor sur la façon d'afficher la documentation. Tous les marqueurs sont optionnels, mais ils doivent respecter une syntaxe spécifique pour être interprétés correctement.

Voici quelques marqueurs à utiliser :

- **@author** : nom de l'auteur
- **@param** : type, nom et description d'un paramètre
- **@return** : type et description du résultat retourné par une fonction
- **@see** : nom d'un autre élément documenté, produisant un lien vers celui-ci
- **@link** : url
- **@todo** : changements à faire dans le futur

Éléments de code à documenter

Plusieurs éléments de code peuvent être documentés : des fichiers, des fonctions, des classes, des méthodes, des propriétés, des variables globales, des constantes.

Nous présentons ci-après un exemple concernant les fonctions.

Une fonction est caractérisée par :

- son nom
- son type et sa valeur de retour (@return)
- ses paramètres (@param)
- sa description

Il est donc possible de la documenter ainsi :

```
/**
 * Echappe les caractères spéciaux d'une chaîne.
 *
 * Envoie la chaîne $str échappée, càd avec les caractères considérés
 * spéciaux par MySQL (tq la quote simple) précédés d'un \, ce qui annule
 * leur effet spécial
 *
 * @param string $str chaîne à échapper
 * @return string
 */
function filtrerChainePourBD($str) {
    if ( ! get_magic_quotes_gpc() ) {
        $str = mysql_real_escape_string($str);
    }
    return $str;
}
```

filtrerChainePourBD^[line 60] - Echappe les caractères spéciaux d'une chaîne.

```
string filtrerChainePourBD( string $str)
```

Envoie la chaîne \$str échappée, c-à-d avec les caractères considérés spéciaux par MySQL (tq la quote simple) précédés d'un \, ce qui annule leur effet spécial

Tags:

return: chaîne échappée

Parameters

string **\$str** chaîne à échapper
[\[Top \]](#)

On peut ajouter autant de lignes @param qu'il y a de paramètres.

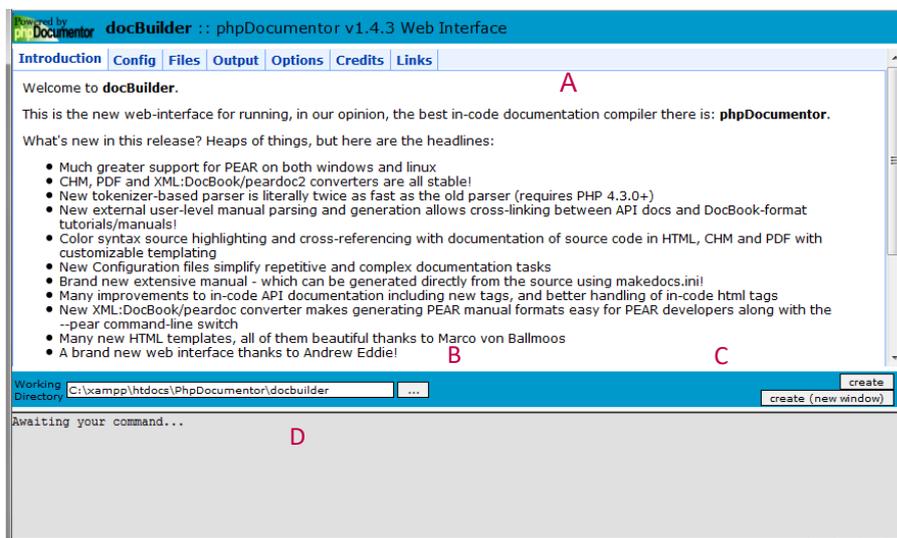
En utilisant le template HTML:Smarty:PHP, on obtient l'extrait suivant dans la page HTML produite par l'outil phpDocumentor :

Génération de la documentation

La génération portera d'une part, sur les fichiers source à interpréter et d'autre part, sur le choix des formats de sortie.

Génération par l'interface web

Via un navigateur, il faut accéder au sous-dossier /docbuilder du dossier /phpdoc accessible du serveur web sur lequel il est installé. Dans le cas où navigateur et serveur web sont sur le même poste, on saisira l'url suivante : http://localhost/phpdoc/docbuilder. La page d'accueil suivante sera alors affichée :



Zone A – Menu horizontal permettant de choisir toutes les options

Zone B – Répertoire de travail, c'est-à-dire le répertoire utilisé par PHP Documentor pour créer ses fichiers temporaires.

Zone C – Boutons permettant de lancer la création de la documentation, soit dans la même page, soit dans une nouvelle page.

Zone D – Console, espace dans lequel les informations d'exécution seront affichées dans le cas où la création sera réalisée dans la même page.

Onglet "Config"

Dans l'onglet *Config*, on peut choisir une configuration spéciale (fichier .ini) déjà préparée. Cette configuration inclut le répertoire source, le répertoire cible, les fichiers à interpréter, ceux à ne pas interpréter, etc... Il suffit donc de créer ce fichier ini, de le sélectionner et de lancer la création de la documentation. Des fichiers .ini exemple se trouvent dans le sous-dossier user du dossier d'installation de PHP Documentor.

Onglet "Files"

Cet onglet permet de définir les fichiers/dossiers à interpréter ou à ne pas interpréter. Trois zones de texte sont donc disponibles pour préciser la liste des fichiers à analyser, la liste des dossiers à analyser et la liste des fichiers à ne pas analyser.

Chaque liste attend des chemins d'accès locaux (ex: d:/xampp/htdocs/appli/fichier.php) séparés par des virgules. Les caractères spéciaux "*" et "?" sont autorisés.

Onglet "Output"

Cet onglet permet de définir dans quel dossier et sous quelle forme les documentations sont à générer. La zone de texte "Target" désigne l'emplacement de la documentation qui sera générée et la zone de texte "Output format" le(s) modèle(s) de sortie qu'elle adoptera. Un format de sortie se compose du format (HTML, XML, PDF, CHM), d'un convertisseur disponible en fonction du format, et d'un modèle proposé par le convertisseur choisi.

Quelques exemples, parmi les nombreux formats disponibles :

- **HTML:frames:default**, le modèle par défaut
- **HTML:Smarty:PHP**, le modèle proche de celui adopté par la documentation sur PHP
- **XML:DocBook/peardoc2:default**
- **PDF:default:default**
- **CHM:default:default**

NB1 : Concernant le format HTML⁶, dans le cas où les pages PHP suivent l'encodage de caractères utf-8, les templates de phpdocumentor pour les entêtes html doivent être modifiés : ce sont les fichiers header.tpl dans le sous-dossier phpDocumentor\Converters\frames ou Smarty. Il faut alors remplacer l'encodage iso-8859-1 par UTF-8 dans la balise *meta* de l'entête html.

NB2: Le menu déroulant sous le champ "Output format" permet de sélectionner un modèle et d'en avoir un aperçu via une petite icône.

Onglet "Options"

Dans cet onglet, il est possible de changer le nom de la documentation (Generated Documentation Title), les noms par défaut des packages et catégories.

Génération par l'interface ligne de commande sous Windows

En premier lieu, il faut ajouter dans la variable d'environnement système path le chemin d'accès qui contient l'interpréteur PHP.

Puis, le générateur de la documentation est appelé comme suit :

```
php repertoire_installation_phpdocumentor\phpdoc [options ligne de commande]
```

Par exemple, la ligne de commande suivante :

```
php c:\xampp\htdocs\phpdoc\phpdoc -f c:\xampp\htdocs\appli\monFichier.php -t c:\docAppli -o HTML:Smarty:PHP
```

interprétera le fichier c:\xampp\htdocs\appli\monFichier.php pour en générer la documentation sous le dossier c:\docAppli au format HTML:Smarty:PHP.

La ligne de commande `php c:\xampp\htdocs\phpdoc\phpdoc -h` fournit la liste des options proposées par la commande `phpdoc`.

⁶ Concernant les formats autres que HTML, l'encodage utf-8 des pages PHP pose problème pour les pages de documentation générées.

3. Ebauches des formulaires

Saisie des frais



Accueil

Renseigner la fiche de frais

Afficher mes fiches de frais

Déconnexion

Renseigner ma fiche de frais du mois 08-2017

Eléments forfaitisés

Forfait Etape

Frais Kilométrique

Nuitée Hôtel

Repas Restaurant

Ajouter

Effacer

Descriptif des éléments hors forfait

Date	Libellé	Montant	
12/08/2017	Achat de fleurs	29.90	Supprimer ce frais
14/08/2017	Taxi	32.50	Supprimer ce frais

Nouvel élément hors forfait

Date (jj/mm/aaaa):

Libellé

Montant :

€

Ajouter

Effacer

Consultation des frais



Mes fiches de frais

Sélectionner un mois :

Mois :

Proposer un calendrier pour la période

Fiche de frais du mois 08-2017 :

Etat : Fiche créée, saisie en cours depuis le 22/08/2017
Montant validé : 0.00

Date d'enregistrement de la dernière opération correspondant à la ligne

Eléments forfaitisés

Forfait Etape	Frais Kilométrique	Nuitée Hôtel	Repas Restaurant
12	562	6	25

Descriptif des éléments hors forfait - 0 justificatifs reçus

Date	Libellé	Montant
12/08/2017	Achat de fleurs	29.90
14/08/2017	Taxi	32.50

Lignes remplies en fonction des données de la base

Validation des frais

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



Choisir le visiteur : Villechalane Louis Mois : 08/2017

Automatiser le remplissage à partir de la base de données des visiteurs. Permettre de saisir le nom pour aller plus vite

Automatiser le remplissage en fonction de la date du jour

Valider la fiche de frais

Eléments forfaitisés

Forfait Etape

Frais Kilométrique

Nuitée Hôtel

Repas Restaurant

[Corriger](#) [Réinitialiser](#)

Rempli automatiquement d'après les saisies des visiteurs et le mois choisi. Modifiable s'il y a contestation par le service comptable (qui pourra vérifier le nombre de rapports et les lieux de déplacement)

Descriptif des éléments hors forfait

Date	Libellé	Montant	
<input type="text" value="12/08/2017"/>	<input type="text" value="Achat de fleurs"/>	<input type="text" value="29.90"/>	Corriger Réinitialiser
<input type="text" value="14/08/2017"/>	<input type="text" value="Taxi"/>	<input type="text" value="32.50"/>	Corriger Réinitialiser

Nombre de justificatifs :

[Valider](#) [Réinitialiser](#)

Rempli automatiquement d'après les saisies des visiteurs et le mois choisi. Il y aura autant de lignes que de saisies "hors forfait" par le visiteur

La validation nécessite l'enregistrement de la date de l'opération.