## 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 E4, 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 E4 (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

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 |



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.

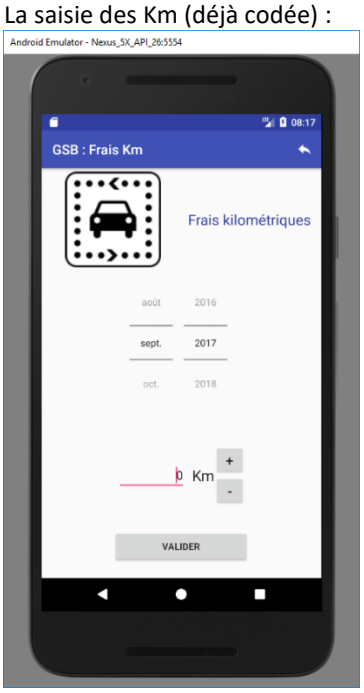
Il est prévu que l'application permette l'envoi direct des informations saisies sur le serveur web qui mettra alors à jour directement la base de données, sans que le visiteur ait à ressaisir les données dans l'application web.

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) :

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\Suivide-vosfrais\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 E4, par les différentes missions réalisées.

| Cahier des charges épreuve E4 | Missions |
|---|---|
| 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.). | Contexte |
| 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. | Contexte |
| 1.3 L'environnement technologique d'apprentissage supportant le système d'information de l'organisation cliente comporte au moins :<br>- un service d'authentification pour les utilisateurs internes et externes à l'organisation ;<br>- un SGBD ;<br>- un accès sécurisé à internet ;<br>- un environnement de travail collaboratif ;<br>- un logiciel de gestion d'incidents ;<br>- un logiciel de gestion des configurations ;<br>- deux serveurs, éventuellement virtualisés, basés sur des systèmes d'exploitation différents, dont l'un est un logiciel open source ;<br>- une solution de sauvegarde ;<br>- des ressources dont l'accès est sécurisé et soumis à habilitation ;<br>- deux types de solution technique d'accès dont une mobile (type smartphone, tablette, ou encore assistant personnel). | Contexte |
| 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 | |
| 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 :<br>- documents de présentation des organisations (organisation cliente et prestataire informatique) ;<br>- description de l'environnement technologique d'apprentissage ;<br>- 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 ;<br>- les schémas d'infrastructure réseau ;<br>- la documentation technique des services disponibles ;<br>- 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 :<br>- un ou deux environnements de développement disposant d'outils de gestion de tests et supportant un framework et au moins deux langages ;<br>- une bibliothèque de composants logiciels ;<br>- un SGBD avec langage de programmation associé ;<br>- 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 :<br>3.2.1 du code exécuté sur le système d'exploitation d'une solution technique d'accès fixe (type client lourd) ;<br>3.2.2 du code exécuté dans un navigateur web (type client léger ou riche, applet, etc.) ;<br>3.2.3 du code exécuté sur le système d'exploitation d'une solution technique d'accès mobile ;<br>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 spécififique) |
| 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) |