PPE 3-4 – 2SLAM
2016-2017

1) Organisation du PPE

a) Présentation générale

La durée hebdomadaire du PPE (Projet Personnel Encadré) est de quatre heures.
Vous disposez approximativement de 20 semaines pour réaliser vos projets.

Cet enseignement doit vous permettre de mettre en œuvre les notions vues en cours et également de faire des recherches pour en découvrir de nouvelles.

Pour cela, il vous est proposé un contexte professionnel dans lequel vous serez acteur pour réaliser différentes situations professionnelles. On vous demandera donc en équipe (3 membres par groupe) d'analyser un cahier des charges d'un nouveau service en tenant compte des exigences de qualité. Vous aurez donc à élaborer un dossier de choix de solutions techniques, de rédiger les spécifications techniques de la solution retenue, de réaliser cette solution et de définir les tests et les niveaux d'habilitation associés au service.

Parallèlement, ce module invite également à contribuer au processus « P5 - Gestion du patrimoine informatique » à travers la mise en place d'un outil de gestion des configurations, l'évaluation de l'investissement nécessaire à la mise en place du service, la mise en place et l'exploitation d'un dispositif de veille technologique, ainsi que l'étude d'une technologie, d'un outil ou d'une méthode afin de proposer une solution actualisée.

b) Lien avec l'examen

Epreuve E5 - Production et fourniture de services informatiques (coeff 5)
Tout d'abord, le sujet proposé est en lien direct avec le référentiel du BTS. Travailler en PPE vous permet donc de conforter les notions vues en cours et donc de préparer au mieux l'étude de cas.

Epreuve E6 – Parcours de professionnalisation (coeff 3)
Par ailleurs, le module PPE vous permet également d'alimenter votre portefeuille de compétences. Vous devez donc identifier les compétences mises en œuvre et analyser votre pratique afin de pouvoir correctement illustrer votre pratique. Attention, vous devez effectuer régulièrement cette démarche car vous serez beaucoup plus efficace que si vous l'effectuez de manière irrégulière et très ponctuelle.

Epreuve E4 – Conception et maintenance de solutions informatiques (coeff 4)
Cette épreuve repose directement sur les travaux effectués pendant le module PPE. En effet, cette épreuve consiste à modifier une situation professionnelle réalisée durant le module PPE. Il convient donc que les spéciations techniques soient à jour et que les solutions techniques soient opérationnelles.

c) Travail en groupe

Chaque groupe est formé de trois ou quatre membres dont un chef de projet. La formation des groupes est irréversible.
Vous devez impérativement tenir un planning de projet en utilisant un logiciel. Vous devrez donc établir un planning prévisionnel et faire le point sur votre avancement régulièrement.
Vous devez donc mettre en œuvre les compétences suivantes :

C1.4.1.1 Établir son planning personnel en fonction des exigences et du déroulement du projet
C1.4.1.2 Rendre compte de son activité
C1.4.2.1 Suivre l'exécution du projet
C1.4.2.2 Analyser les écarts entre temps prévu et temps consommé
C1.4.2.3 Contribuer à l'évaluation du projet
C1.4.3.1 Recenser les ressources humaines, matérielles, logicielles et budgétaires nécessaires à l'exécution du projet et de ses tâches personnelles
C1.4.3.2 Adapter son planning personnel en fonction des ressources disponibles

Par ailleurs, pour chaque technologie utilisée, chaque membre du groupe doit réaliser une tâche. En effet, l'épreuve E4 peut consister à effectuer des modifications sur des travaux effectués par un autre membre du groupe. Les technologies utilisées doivent donc être maîtrisées par l'ensemble du groupe.

d) Veille technologique et étude d'une technologie, d'un composant d'un outil ou d'une méthode

Le cadre technologique du PPE n'étant pas imposé vous allez devoir effectuer des recherches afin de prendre les bonnes décisions. Par ailleurs, les autres membres du groupe devront pouvoir bénéficier de vos recherches. Il conviendra donc de les formaliser et de les diffuser.

Chaque membre doit donc mettre en œuvre les compétences suivantes :

Veille technologique

C5.2.2.1 Définir une stratégie de recherche d'informations
C5.2.2.2 Tenir à jour une liste de sources d'information
C5.2.2.3 Évaluer la qualité d'une source d'information en fonction d'un besoin
C5.2.2.4 Synthétiser et diffuser les résultats d'une veille

Etude d'une technologie, d'un composant d'un outil ou d'une méthode

C5.2.4.1 Se documenter à propos d'une technologie, d'un composant, d'un outil ou d'une méthode
C5.2.4.2 Identifier le potentiel et les limites d'une technologie, d'un composant, d'un outil ou d'une méthode par rapport à un service à produire

e) Environnement de travail collaboratif

Afin de travailler en équipe, vous devrez utiliser un outil collaboratif pour votre code: SVN, GIT… et un pour vos documents : Google Drive, Office 365…

2) Présentation du contexte

VDEV est une ESN (Entreprise de services du Numérique). Depuis sa création, développe des applicatifs pour divers clients. Elle a par exemple développé des applications de gestion de terrasses pour différentes villes ou alors fait évoluer le système d'information d'une compagnie maritime.

Les applicatifs développés sont également de natures diverses. Cela peut être une application Web permettant de réserver des terrasses par des établissements d'une ville. Cette application permet également la consultation des tarifs pratiqués. Cela peut être également une application conçue à l'aide d'un langage objet permet de positionner des terrasses sur la carte d'une ville donnée, en fonction d'un certain nombre de critères. Enfin, VDEV a fait évoluer ses technologies pour tenir compte des terminaux mobiles.

Pour diversifier ses activités, VDEV, décide de s'intéresser désormais au monde rural. Développer des applications pour le monde agricole peut être générateur de profit selon les dirigeants de VDEV. Après une longue campagne de communication et marketing, un premier contrat est en passe d'être finalisé avec l'entreprise AGRUR.

AGRUR est une coopérative agricole spécialisée dans la collecte, la transformation et le conditionnement de la noix qualifiée "noix de Grenoble". Les membres de la coopérative sont des producteurs situés dans la vallée de l'Isère (affluent du Rhône).

AGRUR est engagée dans une démarche qualité et souhaite améliorer la traçabilité de ses produits. Elle envisage donc de réorganiser la partie de son système d'information relative à la gestion des approvisionnements et des ventes.

Le système d'information d'AGRUR se divise en quatre domaines de gestion.

1) <u>Les producteurs</u>

Les membres de la coopérative sont des producteurs de noix, appelés nuciculteurs. Chaque année, la coopérative accepte, dans une certaine proportion, la production de producteurs non adhérents.

Les producteurs qui sont engagés dans la lutte pour le respect de l'environnement obtiennent des certifications garantissant leur engagement dans ce domaine (par exemple les labels Agriculture Biologique). Pour les producteurs adhérents, on conserve une trace des différentes certifications qu'ils possèdent (certification et date d'obtention).

La coopérative souhaite conserver dans son système d'information certaines caractéristiques des producteurs avec lesquels elle travaille : nom et adresse de la société, nom et prénom du responsable de production. Pour les producteurs adhérents, il faut également conserver la date de leur adhésion.

2) <u>Les vergers</u>

Un producteur peut exploiter plusieurs vergers. Parmi les caractéristiques d'un verger, on trouve la variété de la noix, la superficie plantée et le nombre d'arbres à l'hectare.

Certains vergers permettent au producteur d'obtenir les labels AOC (appellation d'origine contrôlée), avec l'appui du Comité Interprofessionnel de la Noix de Grenoble.

Pour avoir droit à l'appellation d'origine contrôlée (AOC), les productions de noix doivent remplir deux conditions :

- provenir de vergers situés dans les communes référencées par la réglementation. Ces communes sont toutes situées le long de la vallée de l'Isère,
- correspondre aux variétés de noix Franquette, Mayette ou Parisienne.

3) La production

Chaque année la coopérative réceptionne les récoltes de ses producteurs. Cette activité de l'entreprise, comme toutes les autres, doit être inscrite dans une démarche qualité assurant la traçabilité des produits finis.

Afin d'assurer cette traçabilité, la coopérative identifie chaque livraison de noix réalisée par un producteur. Une livraison est caractérisée par le verger dont proviennent les noix, sa date, le type de produit (entière fraîche ou entière sèche) et la quantité livrée.

Après réception, les noix sont triées en fonction de leur taille. La coopérative utilise pour cela une calibreuse destinée à répartir la livraison en différents lots de production. Un lot de production est donc constitué de l'ensemble des noix d'une livraison conformes à un calibre particulier.

4) Les campagnes de ventes

Les noix sont vendues à des clients aussi appelés distributeurs qui sont pour l'essentiel des grossistes, des entreprises de la grande distribution ou des professionnels des métiers de la bouche (pâtissiers, restaurateurs).

Une commande est caractérisée par un numéro de commande, une date d'envoi de la commande et un client. Chaque commande porte sur un seul lot de production déterminé en fonction de la demande du client (variété de noix, type de produit et calibre), ce qui permet d'assurer la traçabilité des produits vendus.

Les informations concernant le client (nom, adresse, nom du responsable des achats) doivent être enregistrées.

Pour chaque commande, il faut également mémoriser les types de conditionnement et les quantités commandés ainsi que la date de conditionnement. Les différents types de conditionnement proposés par AGRUR sont : le sachet (de 250 g, 500 g et 1 kg), le filet (de 1 kg, 5 kg, 10 kg et 25 kg) et le carton de 10 kg.

Une commande peut par exemple comporter 800 sachets de 1 kg et 300 filets de 10 kg pour un poids total de 3 800 kg.

3) Travail à faire

Vous êtes développeur chez VDEV, à ce titre vous devez apporter des réponses concrètes aux souhaits d'AGRUR. Avant de répondre aux besoins, vous devez fournir un diagramme de Gantt prévisionnel avec répartitions des tâches. Le découpage des tâches devra être suffisamment fin pour qu'une que l'on puisse suivre précisément le déroulement du projet. Par ailleurs, chaque membre du projet doit être développé dans chaque technologie utilisée. Enfin, le chef de projet mettra à jour un diagramme de Gantt effectif afin de suivre l'avancement du projet.

Préciser ce qu'il faut faire pour aider AGRUR dans la refonte de son système d'information et tenir compte en autres des impératifs de traçabilité et de qualité. Un développeur de VDEV a commencé le travail sur ce besoin et a laissé quelques bribes d'informations que vous trouverez en annexes.

Contraintes : Vous présenterez un dossier d'analyse qui comportera les documents suivants :

- *Dictionnaire de données (ensemble des données non calculées manipulées)*
- *MCD (avec contraintes s'il y a lieu)*
- *Modèle relationnel*
- *Solutions logicielles retenues avec justifications*
- *Implémentation de la solution retenue*
- *Un script de création de la base de données avec un jeu d'essai conséquent*

Par ailleurs, vous choisirez un SGBD ayant des fonctionnalités avancées (contraintes de domaine, implémentation de PL/SQL…).
Une gestion très stricte au niveau de la couche donnée vous est demandée.

Situation professionnelle 1	**Mise en place d'une solution web**

Consulter: annexe 1

Bien qu'il ne soit pas explicitement décrit pour VDEV, AGRUR souhaite à terme permettre à des producteurs de fournir ou modifier directement sur le SI d'AGRUR les informations les concernant. Cela entraîne des problématiques de sécurité et nécessite une authentification.

AGRUR souhaite également suivre administrativement ces informations et distribuer des autorisations d'accès. Pour rendre le système plus attractif, AGRUR, souhaite un suivi plus fin des ventes par une édition de bons de commandes (indépendamment du conditionnement).

De plus, Les responsables d'AGRUR souhaitent que vous puissiez réduire au maximum les erreurs de saisies. Un contrôle de saisie très strict doit être effectué.
Par ailleurs, ces mêmes responsables vous demandent d'améliorer l'ergonomie et l'attractivité de votre site web. Vous devez faciliter les saisies avec par exemple des champs auto complétés. Enfin, vous devez utiliser des contrôles graphiques modernes comme les DatePicker.

Fort de leur expérience dans le développement, les responsables de VDEV vous suggèrent d'utiliser les technologies suivantes : Javascript, Ajax, JQuery, Node.js…

Enfin, AGRUR souhaite aussi tirer profit des logiciels libres et demande à VDEV de lui fournir des solutions dans ce sens.

Contraintes : vous proposerez une solution gérant les quatre domaines décrits précédemment et qui comportera les éléments suivants

- *Un diagramme de cas d'utilisation*
- *Une description textuelle des cas d'utilisation*
- *Un maquettage des IHM (les composants doivent être nommés)*
- *Le choix des Architecture logicielles retenues*
- *Une Charte graphique*
- *La compatibilité avec différents navigateurs.*
- *La visualisation correcte des informations sur terminaux mobiles*
- *Rapport de tests*
- *Une convention de nommage pour les fichiers et les variables*

Enfin, VDEV a recueilli des bonnes pratiques dans la gestion de projet web. Ils vous demandent de vous inspirer de l'exemplaire papier qu'ils vous fournissent. Vous devez mettre en œuvre un maximum de compétences. (voir le tableau de la page suivante).

1) Qu'est-ce qu'un projet web ?	Les contraintes d'un projet web	
	Le cycle de vie d'un projet web	
2) La conduite de projet web	La planification	**C1.4.1.1** Établir son planning personnel en fonction des exigences et du déroulement du projet **C1.4.1.2** Rendre compte de son activité **C1.4.3.1** Recenser les ressources humaines, matérielles, logicielles et budgétaires nécessaires à l'exécution du projet et de ses tâches personnelles
	Le suivi	**C1.4.2.1** Suivre l'exécution du projet **C1.4.2.2** Analyser les écarts entre temps prévu et temps consommé **C1.4.2.3** Contribuer à l'évaluation du projet
	Le pilotage et le management	**C1.4.3.1** Recenser les ressources humaines, matérielles, logicielles et budgétaires nécessaires à l'exécution du projet et de ses tâches personnelles **C1.4.3.2** Adapter son planning personnel en fonction des ressources disponibles
3) La phase de lancement	Le cahier des charges	**C1.1.1.1** Recenser et caractériser les contextes d'utilisation, les processus et les acteurs sur lesquels le service à produire aura un impact
4)		
5) La phase de conception	La conception fonctionnelle	**C1.2.4.1** Recenser les tests d'acceptation nécessaires à la validation du service et les résultats attendus **C1.2.5.1** Recenser les utilisateurs du service, leurs rôles et leur niveau de responsabilité **C1.2.5.2** Recenser les ressources liées à l'utilisation du service **C1.2.5.3** Proposer les niveaux d'habilitation associés au service **C4.1.2.1** Définir les spécifications de l'interface utilisateur de la solution applicative **C4.1.2.2** Maquetter un élément de la solution applicative
	La conception graphique	**C4.1.2.1** Définir les spécifications de l'interface utilisateur de la solution applicative **C4.1.2.2** Maquetter un élément de la solution applicative
	La conception technique	**C1.1.2.2** Recenser les composants de l'architecture technique sur lesquels le service à produire aura un impact
6) La phase de réalisation	La réalisation	**C4.1.6.2** Mettre en place et exploiter un environnement de test
	Définition des environnements	**C1.3.1.1** Mettre en place l'environnement de test du Service **C4.1.6.2** Mettre en place et exploiter un environnement de test
	Les tests	**C1.3.1.3** Rédiger le rapport de test **C4.2.3.1** Élaborer et réaliser des tests d'intégration et de non régression de la solution mise à jour
	Les tests unitaires	**C4.2.2.3** Élaborer et réaliser les tests unitaires des composants mis à jour
7) La phase d'exploitation	La documentation d'un projet	**C4.1.9.1** Produire ou mettre à jour la documentation technique d'une solution applicative et de ses composants logiciels **C4.1.10.1** Rédiger la documentation d'utilisation, une aide en ligne, une FAQ **C4.1.10.2** Adapter la documentation d'utilisation à chaque contexte d'utilisation
8)		
9) Les bonnes pratiques	Le référencement	**C4.1.2.3** Concevoir et valider la maquette en collaboration avec des utilisateurs **C5.2.1.2** Identifier et partager les bonnes pratiques à intégrer
	L'ergonomie	
	Le développement des pages web	
	Le graphisme	
	Les formulaires	
	Les pages d'accueil	
	La réglementation	

Situation professionnelle 2	Création d'un outil de gestion de suivi de commande

Consulter: annexes 2, 3, 4.

Contexte

Les distributeurs de noix sont des partenaires privilégiés de NOIXCOOP. Ils représentent en effet 80% des débouchés de la production de NOIXCOOP.

NOIXCOOP réalise 10% de son chiffre d'affaires avec l'un de ces distributeurs (SUPERMARKET). Ce distributeur vient d'actualiser son système d'information et impose à ses fournisseurs une interconnexion des systèmes d'information au moyen d'une architecture orientée "*Web Service*" (architecture WS).

Chez NOIXCOOP les commandes sont gérées par téléphone tant pour la prise de commande que pour le suivi. Afin d'améliorer le suivi des commandes et de répondre à la demande de SUPERMARKET, NOIXCOOP décide donc de mettre à la disposition de ses clients un service *web* qui leur permettra d'obtenir un document XML présentant la liste de leurs commandes en cours.

Votre rôle consiste à étudier la rentabilité de ce service *web*, puis à le développer.

Principe de fonctionnement du service web

- Le distributeur devra avoir obtenu auprès de NOIXCOOP un identifiant de connexion au service (exemple carr15432 pour le distributeur *carreclerc*).
- À l'aide de cet identifiant de connexion, une application cliente pourra interroger le service *web* à l'aide d'une requête HTTP GET :

http://ws.noixcoop.fr/services/etatCommandes.php?distributeur=carr15432

- Il obtiendra en retour la liste de ses commandes en souffrance (non expédiées) au format XML (Cf *annexe*).

Un ensemble de classes présentées en *annexes* est en cours de développement.

- La classe *PersistanceSQL* permet d'accéder à la base de données. Elle permet de charger et/ou d'enregistrer les données dans la base.
- La classe *GestionCommandes* permet d'orchestrer les traitements liés au service Web.
- La dépendance *Utilise* représente simplement le fait que la classe *GestionCommandes* doit avoir connaissance de la classe *Distributeur* puisqu'elle possède une méthode retournant un objet de la classe *Distributeur*.

Chargement des données depuis la base

Le fonctionnement de la méthode *getDistributeur*() de la classe *GestionCommandes* est le suivant :

- Elle utilise la classe *PersistanceSQL* pour charger les données concernant un distributeur depuis la base *gestcommande*(l'identifiant de ce distributeur étant passé en paramètre).
- Ce chargement provoque de manière automatique le chargement des données concernant l'ensemble des commandes de ce distributeur, ainsi que les produits associés.
- Elle retourne l'objet *Distributeur* ainsi créé, cet objet possédant la collection des commandes du distributeur.

On vous demande donc de produire une application de type « client lourd ».
Cette application permettra de générer le fichier XML nécessaire au Web Service. Le fichier doit contenir les commandes en souffrance d'un distributeur sélectionné au préalable.

Votre applicatif permettra au gestionnaire de gérer l'envoi des commandes. (les données devront être mises à jour). Par ailleurs, il aura la possibilité d'éditer le détail d'une ou de commande(s) d'un distributeur donné.

Les documents d'analyse sont donnés à titre indicatif et peuvent évoluer.

Contraintes : vous proposerez une solution permettant l'édition de brochure au format pdf et portera les éléments suivants
- *Un diagramme de cas d'utilisation*
- *Une description textuelle des cas d'utilisation*
- *Un diagramme de classe UML*
- *Un maquettage des IHM (les composants doivent être nommés)*
- *Le choix des Architecture logicielles retenues*
- *Rapport de tests*
- *Tests unitaires*
- *Une documentation technique au format html*
- *Vous utiliserez les conventions de nommage appliquées dans le langage java*

Situation professionnelle 3	Création d'une application mobile

Les fortes contraintes concurrentielles obligent AGRUR à changer l'ensemble des vergers Elle souhaite travailler avec des propriétaires de vergers qui respectent les contraintes environnementales.

Des agents sont chargés de collecter les informations sur ces nouveaux vergers. AGRUR souhaite que ces informations soient saisies dans une application native. Les smartphones n'ayant pas été fournis à ses agents vous avez le choix pour la technologie utilisée.

Les informations doivent au minimum être stockées sur le smartphone mais il est bien évident qu'une solution mettant à jour la base de données serait à privilégier.

Contraintes : vous proposerez une solution mobile et apporterez les éléments suivants

- *Un diagramme de cas d'utilisation*
- *Une description textuelle des cas d'utilisation*
- *Un maquettage des IHM (les composants doivent être nommés)*
- *Le choix des Architecture logicielles retenues*
- *Rapport de tests*

Mission transversale	Gestion des incidents

Afin de

- faciliter les échanges entre AGRUR et VDEV pour ce qui concerne des anomalies ou des incidents de fonctionnement relatifs aux applications que VDEV va réaliser pour AGRUR,
- garantir la traçabilité de chaque demande (nature de la demande, nom du technicien VDEV qui a pris en charge cette demande, statut et informations de suivi de la demande, ….),

VDEV a retenu la solution du logiciel GLPI pour permettre la déclaration des incidents et leur suivi.

La direction de VDEV vous demande :

- de mettre en place cette solution
- de la paramétrer en fonction des objectifs ci-dessous.
- d'effectuer l'ensemble des tests qui permettent de valider votre paramétrage par rapport aux objectifs fixés.

Le paramétrage à effectuer doit répondre aux objectifs suivants :

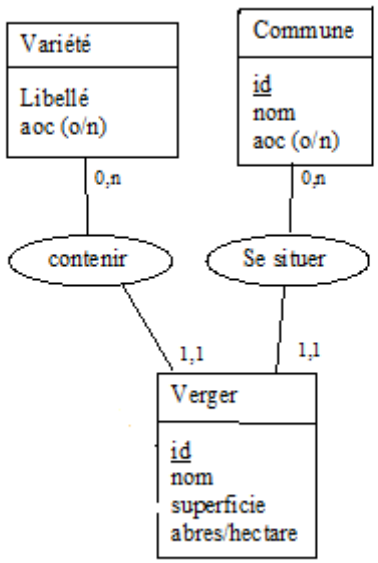
- Tous les clients de VDEV doivent être en mesure d'utiliser cet outil. AGRUR est l'un d'entre eux, mais votre paramétrage doit permettre que d'autres clients y accèdent.
- Tous les salariés de AGRUR (ou d'une entreprise Y pour laquelle VDEV a développé une application) qui utilisent vos applications doivent pouvoir saisir directement leurs anomalies ou incidents dans l'outil.

- Par contre la Direction de VDEV ne veut absolument pas qu'un client puisse avoir accès aux problèmes qui se sont posés chez un autre client !
- Au sein des équipes de développement de VDEV n'importe quel développeur peut être amené à prendre en charge un incident qui s'est produit chez n'importe quel client.
- Chez un client donné :
    - Un utilisateur ne peut accéder qu'aux tickets qu'il a lui-même saisi.
    - Mais, pour faciliter le suivi du projet et du contrat avec VDEV, un responsable chez le client est autorisé à avoir une vision de l'ensemble des problèmes qui ont été remontés au sein de son entreprise et uniquement au sein de celle-ci. Il ne peut donc pas accéder aux problèmes des autres clients conformément à la volonté de la Direction.
- Par ailleurs, au sein de VDEV, on doit pouvoir faire le point sur tous les problèmes signalés par n'importe quel client sur une application donnée.

Par ailleurs, vous réfléchirez à une stratégie de sauvegarde et de restauration de données et vous mettrez en place les techniques en conséquence.

Annexe 1 - Extrait d'un MCD et d'un modèle relationnel

Les bribes d'informations laissées par le développeur de VDEV :



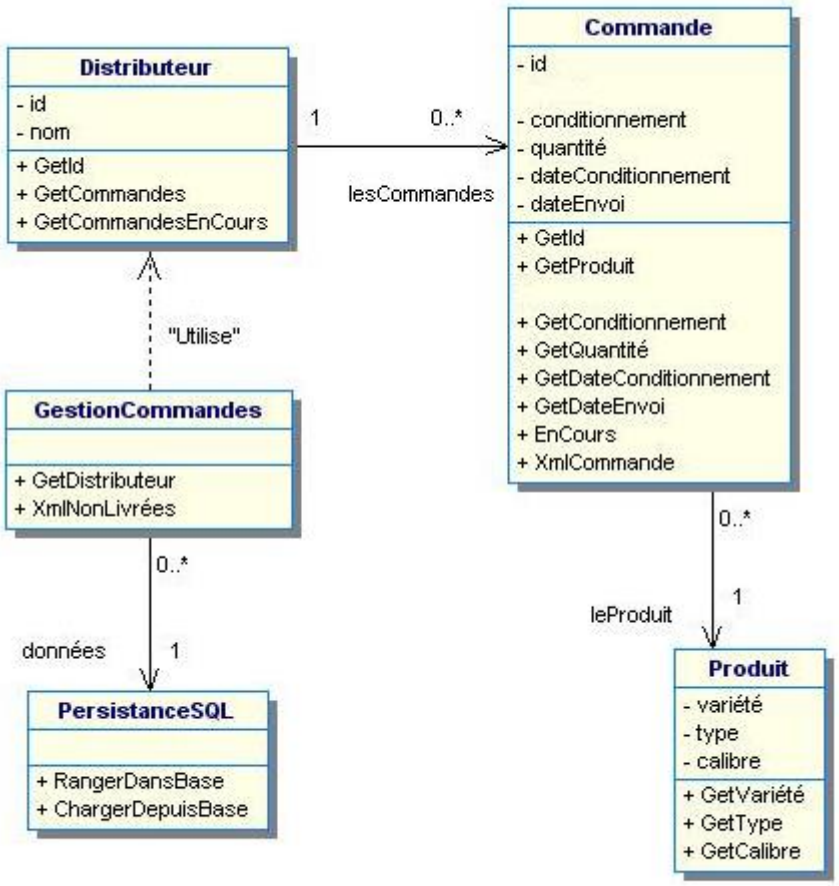
Commande(<u>numéro</u>, date, numLot#)
Conditionnement(<u>id</u>, libellé, poids)
Comporter(<u>numéro#, id#</u>, quantité)

Annexe 2 - Exemple d'utilisation du service *web*

<u>Exemple</u> : le distributeur *carreclerc* désire obtenir l'état de ses commandes restant à livrer. Une commande qui reste à livrer est une commande dont la date d'envoi n'est pas renseignée (sa valeur est NULL). Il sollicite le service *web* "etatCommandes" et obtient le fichier XML ci-dessous.

```
<?xml version="1.0" encoding="UTF-8"?>
<commandes idDistributeur="carr15432" xmlns:xlink="http://www.w3.org/1999/xlink">
        <commande id="00213">
                <produit variete="Mayette" type="Fraiche Entière" calibre="2" />
                <conditionnement type="filet 1kg" />
                <quantite>50</quantite>
                <date_conditionnement>12-05-14</date_conditionnement>
                <date_envoi>null</date_envoi>
        </commande>
        <commande id="00215">
                <produit variete="Parisienne" type="Cerneaux" calibre="1" />
                <conditionnement type="filet 5kg" />
                <quantite>100</quantite>
                <date_conditionnement>08-05-14</date_conditionnement>
                <date_envoi>null</date_envoi>
        </commande>
</commandes>
```

Annexe 3 - Diagramme de classes partiel



Par souci de simplification, on considère ici qu'il n'y a qu'un seul conditionnement possible par commande. Vous choisirez une des deux solutions.

Annexe 4– Descriptifs des classes

Classe PersistanceSQL

Public
    // Constructeur
    PersistanceSQL (ipBase : chaîne, port : entier, nomBaseDonnee : chaîne)
    // Construit un objet *PersistanceSql.* Cet objet permettra de charger les données depuis une base
    // de données ou de les sauvegarder dans la base.

    procédure RangerDansBase (unObjet : Objet)
    // Stocke les données de l'objet dans la base de données.

    fonction ChargerDepuisBase (id : chaîne, nomClasse : chaîne) : Objet de la classe *NomClasse*.
    // Retourne l'objet de la classe *NomClasse* dont l'identifiant est "id". Cet objet est chargé
    // depuis la base de données, ainsi que l'ensemble de ses objets liés (voir l'exemple d'utilisation
    // ci-dessous). Retourne NULL si aucun objet de cette classe ne possède cet identifiant.

Fin classe

Exemple d'utilisation :

// *persist* est une instance de PersistanceSQL
Distributeur leDistributeur ← persist.chargerDepuisBase ("2", "Distributeur")
// *leDistributeur* est l'instance de la classe Distributeur dont l'identifiant est 2.

Toutes les commandes du distributeur sont automatiquement chargées dans la collection *leDistributeur.lesCommandes*. Chaque produit commandé est également chargé, et se trouve donc référencé par le champ *leProduit* de l'objet Commande correspondante.

Classe Distributeur

Privé
    id : chaîne
    nom : chaîne
    lesCommandes : Collection de <Commande> // Toutes les commandes du distributeur.

Public
    // Constructeur
    Distributeur (unId : chaîne , unNom : chaîne)
    // Construit un objet Distributeur. Á ce stade, il n'est pas stocké dans la base de données.

    fonction getId () : chaîne
    // Retourne l'identifiant du distributeur.

    fonction getCommandes () : Collection de <Commande>
    // Retourne l'ensemble des commandes passées par ce distributeur.

    fonction getCommandesEnCours () : Collection de <Commande>
    // Retourne une collection constituée des commandes en cours (non expédiées) du distributeur.

Fin classe

Classe Commande

Privé
- id : entier // Identifiant de la commande.
- leProduit : Produit // Le produit commandé (catégorie de noix).
- conditionnement : chaîne // Type de conditionnement.
- quantité : entier // Quantité de produits conditionnés commandée.
- dateConditionnement : Date // Date de conditionnement de la commande.
- dateEnvoi : Date // Date d'envoi de la commande.

Public

// Constructeur
Commande (…)
// Construit un objet Commande, la liste des paramètres est sans importance.
// Le champ dateEnvoi est initialisé à NULL.

fonction getId () : entier
// Retourne l'identifiant de la commande.

fonction getProduit () : Produit
// Retourne le produit commandé.

fonction getConditionnement () : chaîne
// Retourne le type de conditionnement des produits de cette commande.

fonction getQuantite () : entier
// Retourne la quantité commandée.

fonction getDateConditionnement () : Date
// Retourne la date de conditionnement de la commande.

fonction getDateEnvoi () : Date
// Retourne la date d'envoi de la commande.

fonction EnCours () : booléen
// Renvoie vrai si la commande n'est pas encore expédiée, faux sinon.
// Une commande n'est pas expédiée si sa date d'envoi contient NULL.

fonction XmlCommande () : chaîne
// Retourne la chaîne correspondant au code XML représentant la commande (voir *annexe*).
// Cette fonction est appelée par la méthode *XmlNonLivrees()* de la classe
// *GestionCommandes* décrite ci-après.

Fin classe

Classe Produit

Privé
    variete : chaîne   // Variété de noix, exemple : "Mayette".
    type : chaîne   // Type de noix, exemple : "fraîche entière".
    calibre : entier   // Calibre des noix, exemple : 2.

Public
    // Constructeur
    Produit (…)
    // Construit un objet Produit, la liste des paramètres est sans importance.

    fonction getVariete () : chaîne
    // Renvoie la variété du produit.

    fonction getType () : chaîne
    // Renvoie le type du produit.

    fonction getCalibre () : entier
    // Renvoie le calibre du produit.

Fin classe

Classe GestionCommandes

Privé
    donnees : PersistanceSQL
    // Attribut qui permet de rendre les objets métiers accessibles.

Public
    //Constructeur
    GestionCommandes (lesDonnees : PersistanceSQL)
    // Construit un objet GestionCommandes avec un modèle de persistance associé.

    fonction getDistributeur (idDistributeur : chaîne) : Distributeur
    // Retourne l'objet Distributeur qui possède l'identifiant idDistributeur passé en paramètre,
    // retourne null si aucun Distributeur ne possède cet identifiant.

    fonction XmlNonLivrees (unDistributeur : Distributeur) : chaîne
    // Retourne une chaîne de caractères qui représente le document XML de la liste des commandes
    // non livrées du distributeur passé en paramètre comme le montre l'exemple de l'*annexe*.

Fin classe