**S39-S43 Phase de réalisation du projet – Période 0**

# Le sujet lol(ce qui sera fourni)

L’existant du projet comprend :

* Les fichiers de données ;
* Le chahier des spécifications fonctionnelles détaillées ;
* Le schéma relationnel en étoile Oracle;
* Le dictionnaire des données ;
* Le script SQL de création des tables optimisées pour le décisionnel pour le SGBD Oracle. Des tables techniques proposent la gestion des utilisateurs et leur profil.
* Les fichiers déjà créés PHP/javascript de l’application web architecturée *Model View Controler* qui :
  + Pour la partie administration :
    - Pour l’implantation de la base de données, appelle le script SQL via PHP.
    - Pour la gestion de la base décisionnelle appellent dans des exemples commentés les jobs Talend à déployer donc en tant que web services Apache Axis;
    - Pour la gestion des utilisateurs, modifications directes des tables techniques de la base de données
    - proposent des options (ou alors options présentes dans le fichier de configuration central de l’application) pour la partie navigation rapports dynamiques DeskI/WebI, rapports statiques PDF originaires de DeskI/WebI
    - proposent des options (ou alors options présentes dans le fichier de configuration central de l’application) pour le mode d’appels web  :
      * formulaires HTML5 avec méthode HTTP GET/POST au choix ;
      * liens HTML avec méthode HTTP GET ;
      * iframe HTML avec méthode HTTP GET ;
      * PHP SOAP pour invocation SOAP des web services avec méthode HTTP GET/POST au choix ;
      * PHP CURL pour invocation REST des web services avec méthode HTTP GET/POST au choix ou un dérivé comme goutte;
      * AJAX XMLHttpRequest SOAP pour invocation SOAP des web services avec méthode HTTP GET/POST au choix ;
      * AJAX XMLHttpRequest pour invocation REST des web services avec méthode HTTP GET/POST au choix ;
  + Pour la partie navigation, pour les 3 profils différents  (DM, DR, DC) :
    - appellent  des rapports dynamiques SAS déployés sur le serveur SAS BI ;
    - appellent  des rapports dynamiques SAS via un serveur SAS/IntrNet .

Plus exactement, le projet [Darties-20151](http://foad.univ-ubs.fr/mod/url/view.php?id=89688) Oracle 12c, portail Apache/PHP 5.3.3 hébergé sur le serveur chelidae connecté via PDO/OCI Oracle pour générer en PHP les filtres basés sur les données relationnelles avec du code javascript pour la gestion des dépendances entre filtres. Le site PHP est basé sur une architecture MVC utilisant un modèle [PARIS](http://paris.readthedocs.org/), l'active record au dessus de l'ORM [IDIORM](https://idiorm.readthedocs.org) étendu pour supporter Oracle via PDO et le moteur de templates [TWIG](http://twig.sensiolabs.org/). L'extension cURL de PHP invoque via le web scrapper [goutte](http://goutte.readthedocs.org) les processus stockés SAS via la SAS Stored Processes Web Application et qui modifie les tableaux produits par SAS à la volée. Les Jobs Talend pour alimenter la base Oracle sont exportés en tant que web services dans une application war dérivée du moteur SOAP Apache Axis1 déployée sur un serveur d'application web tomcat hébergé sur le serveur chelidae. Les web services Talend sont invoqués via un formulaire html et ouvre une fenêtre pour afficher le résultat. La carte des régions commerciales est colorée par la proc GMAP de SAS à partir des contours des départements fournis par l'application gratuite IGN GEOFLA Départements simplifiés via les fonctions topologiques introduites dans la version 2.0 de la surcouche PostGIS du serveur PostgreSQL. La carte a été ensuite exportée en shapefile en vue de son importation sous SAS. La carte est rendue via la proc GMAP de SAS. Le placement des magasins se base sur le Répertoire Géographique des Communes édition 2015 fournis par l'IGN, aussi un fichier gratuit. Les tables avec attributs géographiques departements et rgc\_2015 partagent sous PostGIS/PostgreSQL le même système de référence. Les logos des enseignes sont placés sur la carte via un dataset annotate.

Concernant la qualité du code PHP, outre les approches SonarQube/Jenkins, l’auteur de l’application web signale le [PHP-FIG](http://www.php-fig.org/) (PHP Framework Interoperability Groupement) qui édite les PSR (PHP Standard Recommendation).

PSR, pour PHP Standard Recommendation, est un ensemble de normes pour PHP qui permet de faciliter l’interopérabilité des composants entre eux.

Elles sont éditées par le php-fig : PHP Framework Interoperability Groupement. C’est un groupement de personnes qui travaillent autour de ces recommandations. Tout le monde peut faire partie de ce groupement. Mais seuls certains membres avec le droit de vote peuvent voter sur les nouvelles recommandations. Les membres votants sont généralement des représentants de projets tels que Doctrine, Composer ou Symfony.

Il y a actuellement 7 normes validées qui sont prises en compte par les développeurs PHP :

* PSR-4 est la norme pour l’auto chargement des classes. PSR-1 étant déprécié.
* PSR-1 et PSR-2 vont se concentrer sur la forme du code
* PSR-3 est pour la mise en forme de log.
* PSR-6 pour la mise en cache d’objets (Redis, memcached)
* PSR-7 pour les requêtes/réponses HTTP

Toutes ces normes vont permettre de bien structurer le code, d’avoir les mêmes interfaces, et de permettre aux autres développeurs de contribuer plus facilement.

Contrainte globale sur tout le projet : Respect du Top 10 [OWASP](https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project).

# Travail à réaliser par la phase 0 :

1. Porter le script d’implantation des tables et d’alimentation originelle pour MySQL/InnoDB Exécuter ce script d’implantation des tables via PHPMyAdmin.
2. Pour les vues contenues dans le script Oracle, proposez leur équivalent pour MySQL et lors ce que ce n’est pas possible (exemple Common Table Expression Récursive) proposez une procédure stockée (qui peut renvoyer une table contrairement à une fonction) qui donne le même résultat.
3. Adapter le code IDIORM pour faire appel à ces procédures stockées MySQL.

Au pire :

$test = ORM::get\_db('connection\_name');

// stored procedure - 1 IN PARAM, 4 OUT PARAM

$statement = $test->prepare('CALL sp\_test(?, @param1, @param2, @param3, @param4)');

$value = 1;

$statement->bindParam(1, $value);

$statement->execute();

$outputArray = $test->query("SELECT @param1, @param2, @param3, @param4")->fetch(PDO::FETCH\_ASSOC);

1. Les rapports sous forme de tableaux pour la navigation proviennent de l’invocation via une URL par goutte de programmes du l’outil décisionnel SAS qui utilisent des procédures SQL voire une procédure Print pour l’affichage. Ces programmes SAS peuvent être remplacés par un enrichissement de votre modèle IDIORM et par son appel dans des pages PHP/HTML/TWIG.
2. Les cartes proviennent de l’invocation via une URL par goutte d’un programme du l’outil décisionnel SAS qui utilisent des étapes data, des procédures SQL, des procédures GMap et Mapimport. Ce programme, dans une version modifiée a permis de pré-générer des cartes sous forme de fichier PNG. A vous d’implanter l’appel au bon fichier PNG selon le contexte.
3. Le portail d’administration est à faire en IDIORM/TWIG. Il consiste en l’administration des changements de mots de passe (centralisée par l’administrateur) et décentralisé (chaque utilisateur peut changer son mot de passe dans la rubrique « Mon compte » et le changement de profil uniquement centralisé. La création et la suppression d’utilisateurs sont aussi à faire. L’attribution d’un profil temporaire peut être envisagée s’il y a un jeu de chaises musicales. La création d’un nouveau profil concernant uniquement les magasins nouvellement créés (car non déjà attribué à un profil) est aussi à faire. Eventuellement le menu d’administration de l’entrepôt de données peut être prévu. Il comprend l’exécution de scripts SQL et de script PL/SQL pour créer la structure de l’entrepôt vide. Il comprend aussi l’appel de jobs talend déployés en tant que web services. Lors de l’attribution des profils, les tables relatives à la sécurité qui font le lien entre utilisateur et entrepôt de données doivent être mise à jour.
4. L’alimentation via Talend implique le dépôt d’une archive contenant des fichiers CSV dans un répertoire accessible aux jobs Talend déployés en tant que web services. Ce dépôt peut être fait à la main ou par un File Upload en PHP ou en JSP. De plus, l’alimentation suppose la consultation d’un journal de mise à jour qui a été créé par le job.
5. Vous devez implanter des tests au niveau de l’ORM via [PHPUnit](https://github.com/sebastianbergmann/phpunit/) et procéder au suivi de la qualité via un serveur [SonarQube](http://fr.wikipedia.org/wiki/SonarQube) muni des modules pour les langages [Web](http://docs.sonarqube.org/display/PLUG/Web+Plugin)/[JavaScript](http://docs.sonarqube.org/display/PLUG/JavaScript+Plugin)/[PHP](http://docs.sonarqube.org/display/PLUG/PHP+Plugin) et lié à chaine d'intégration continue [Jenkins](https://fr.wikipedia.org/wiki/Jenkins_%28informatique%29) éventuellement doté de ses plugins relatifs à la qualimétrie PHP ([Phing](http://www.phing.info/trac/), [PHP Depend](http://pdepend.org/), [PHP Mess Detector](http://phpmd.org/), [phpcpd](https://github.com/sebastianbergmann/phpcpd), [phploc](https://github.com/sebastianbergmann/phploc), [PHP\_CodeSniffer](http://pear.php.net/package/PHP_CodeSniffer/redirected), [phpDocumentor](http://www.phpdoc.org/), [PHP\_CodeBrowser](https://github.com/Mayflower/PHP_CodeBrowser) / [Ant](http://ant.apache.org/), [Template for Jenkins Jobs for PHP Projects](http://jenkins-php.org/)) et réactive aux changements de sources gérés par [SVN](https://fr.wikipedia.org/wiki/Apache_Subversion) et/ou [Git](https://fr.wikipedia.org/wiki/Git). L’alternative est de prendre en compte les PSR (PHP Standard Recommendation) produites par le [PHP-FIG](http://www.php-fig.org/) (PHP Framework Interoperability Groupement).
6. Vous devez implanter une façade XML/json au niveau de l’ORM qui, de manière sécurisée, fournira les données aux applications mobiles.

NB : juste après le lancement du projet, un TP (MySQL InnoDB avec déclencheurs, procédures stockées et fonctions) avec libre accès au cours portera sur les transactions InnoDB .

**S45-S51 Phase de réalisation du projet – Période 1**

Les 5 utilisateurs ayant le profil directeur régional et les 48 utilisateurs ayant le profil directeur magasin doivent pouvoir accès à leur tableau de bord sur Google Android. Il faut certainement un vrai certificat sur le serveur de production. Les tests seront faits pour Android. Spring-Android est un client REST pour Android tandis que ksoap2 API for SOAP est un client SOAP pour android.

# Travail à réaliser par la phase 1 :

1. A remplir par M. Le Lain.
2. Faire les tests sur les classes du client REST/SOAP avec JUnit et procéder au suivi de la qualité via un serveur [SonarQube](http://fr.wikipedia.org/wiki/SonarQube) et lié à chaine d'intégration continue [Jenkins](https://fr.wikipedia.org/wiki/Jenkins_%28informatique%29) et réactive aux changements de sources gérés par [SVN](https://fr.wikipedia.org/wiki/Apache_Subversion) et/ou [Git](https://fr.wikipedia.org/wiki/Git).

**S01-S03 Phase de réalisation du projet – Période 2**

Développement Java de l’application Web (servlet) / MVC avec [Maven](https://fr.wikipedia.org/wiki/Apache_Maven), [Hibernate](http://hibernate.org/) / JPA 2 pour l’ORM, le moteur de *templates* [Apache Tiles 3](http://tiles.apache.org/) éventuellement intégré à [Spring MVC 4](http://docs.spring.io/spring/docs/4.0.x/spring-framework-reference/html/mvc.html) ([Spring Data JPA](http://projects.spring.io/spring-data-jpa/) pour l’ORM Hibernate dans ce cas là) ou une portlet [Liferay](https://www.liferay.com/) couplée à Spring avec utilisation du service Builder Liferay (outil de génération de sources ). L’interface [Vaadin](https://vaadin.com/swing) peut être intégrée à [Spring UI](https://spring.io/guides/gs/crud-with-vaadin/). Le périmètre peut être réduit au directeur commercial et à l’administrateur. Pour l’alimentation, les fichiers CSV sont donnés.

# Travail à réaliser par la phase 2 :

1. Porter le code PHP/Paris-Idiorm/TWIG/MySQL de la phase 0 sans la façade vers Java/Hibernate/Tiles/MySQL.
2. Faire les tests sur les classes JPA avec JUnit et procéder au suivi de la qualité via un serveur [SonarQube](http://fr.wikipedia.org/wiki/SonarQube) et lié à chaine d'intégration continue [Jenkins](https://fr.wikipedia.org/wiki/Jenkins_%28informatique%29) et réactive aux changements de sources gérés par [SVN](https://fr.wikipedia.org/wiki/Apache_Subversion) et/ou [Git](https://fr.wikipedia.org/wiki/Git).
3. L’alimentation sera prise en charge par des jobs Talend déployés en tant que web service.

**S05-S07 Phase de réalisation du projet – Période 3 (temps plein seulement)**

Développement de l’application java Swing ? et ORM vers [MongoDB](http://www.mongodb.com/) avec [Hibernate OGM](http://hibernate.org/ogm/) ou [Spring Data pour MongoDB](http://projects.spring.io/spring-data-mongodb). Des anciens projets LP CSD ont livré deux applications Java pour le profil chef de Produit et Directeur Magasin. Les spécifications sont disponibles. L’OGM étend JPA au NoSQL.

# Travail à réaliser par la phase 3 :

1. Finir l’application de la phase précédente.
2. Définir l’OGM vers MongoDB.
3. Tester les classes de l’OGM avec JUnit et procéder au suivi de la qualité via un serveur [SonarQube](http://fr.wikipedia.org/wiki/SonarQube) et lié à chaine d'intégration continue [Jenkins](https://fr.wikipedia.org/wiki/Jenkins_%28informatique%29) et réactive aux changements de sources gérés par [SVN](https://fr.wikipedia.org/wiki/Apache_Subversion) et/ou [Git](https://fr.wikipedia.org/wiki/Git).