|  |
| --- |
| **Estimation pour une mise en « conformité » selon recommandations audit CNAM** |

**Rappel du contexte :**

Espace Employeur n’a été maintenu qu’en attente du palier 3 du développement de IXI.  
Le code date des années 2000, plusieurs développeurs se sont succédé sur son développement et sa maintenace sans refonte aux standards actuels. Certaines recommandations sont forcément liées entre elles. (Ex : migration de base MySQL vers PostgreSQL et ORM).

* **NB : Le fonctionnel et l’ergonomie restent identiques.**
* **Pas de gestion d’authentification via AccessMaster pour les agents**
* **Pas de gestion d’authentification via Net.entreprise pour les employeurs**
* **Nous rappelons que la réécriture aux standards MD23 est la solution la plus appropriée surtout si le service espace employeur doit durer dans le temps.**

## Mise à jour ou remplacement des bibliothèques utilisées

* **FPDF** : Mise à jour de la librairie
* **Php\_mailer** : La version la plus récente est utilisée
* **PHPExcel** : Complétement dépréciée obligation de passer à PHPSpreadsheet
* **pibarcode** : Remplacement par une nouvelle bibliothèque plus récente ou suppression si non utilisée dans les modules « détachements à l’étranger » et « Signalements d’entrée dans un régime spécial »
* **ccBrowserInfo** : Suppression , utilisée que dans un fichier
* **ccSiteStyle** : Suppression n’est pas utilisée

**Temps : 15 jours (Tests manuels de non régression inclus)**

## Remaniement de l’organisation du code type MVC

* Revue de code pour éviter les redondances de définition de fonction ex :**DateA2F()** dans plus d’une quarantaine de fichiers, d’autres fonctions sont concernés.
* L’encodage des fichiers n’est pas le même pour tous les fichiers.
* Créer et déplacer le code (séparation du JavaScript , du css) puis dans des controllers, modèles, vues avec ajout de routage, implique une refonte **complète** de l’architecture du code (Une réécriture est plus à propos)

**Temps de réorganisation du code « proches » des normes MVC + tests manuels de non régression: 60 jours**

## Migration de la base de données MySQL vers PostgreSQL :

Cette recommandation implique des modifications conséquentes de code.

* Réécriture des requêtes (Sans ORM la syntaxe des fonctions changent *ex : mysql\_connect => pg\_connect*)
* Gestion des transactions sql (code à modifier)
* Des requêtes préparées pour éviter les injections SQL.
* Refonte de la base de données, uniformisation des champs qui n’ont pas le même datatype entre tables.
* Les index redondants ou absents.

**Temps de migration (ré-injection des données inclus) et modification du code en conséquence : 60 jours (L’ajout d’un ORM serait conseillé)**

## Utilisation d’un ORM  :

La mise en place d’un ORM type **Propel** ou **Doctrine** implique un remaniement de tout le code ayant accès à la base de données pour s’affranchir des requêtes SQL. Cela implique la migration de la base MySQL vers PostgreSQL (le temps de migration est à cumuler)

**Temps de mise en place + tests manuels de non régression: 30 jours + 60 jours (migration en Postgresql)**

## Migration de php 5 à php 7 :

* Les magic quotes (3 sortes de magic\_quotes) ont été supprimées dans php5.4, il faut donc modifier le code en conséquence pour éviter les attaques types SQL injection. (Un ORM est la solution idéale)
* Les fonctionnalités dépréciées dans php7 devront être supprimées et remplacées
* Les fonctions modifiées dans php7 implique des modifications de code
* L’incohérence de l’encodage, revue de tout le code et harmonisation, l’utilisation des balises < ?php en place de < ?.
* Fichiers de configuration (passage au format YAML) dans des répertoires non accessibles du navigateur, et non exécutables.
* La gestion des exceptions

Un passage de php 5.3 à php 7 implique une montée de version par palier pour éviter de trop grandes difficultés de migration. Même si des outils tel que **Php 7 Mar ou Php7aid** existent et génèrent des rapports aidant à la migration.

**Temps de mise en place + tests manuels de non régression: 30 jours**

## Gestion des erreurs accès base de données

Cette recommandation de l’audit implique la mise en place d’un ORM qui répondra à toutes les remarques concernant les transactions et les accès concurrentiels (Migration vers PostgreSQL) et la gestion des erreurs.

**Cf temps inclus : Mise en place d’un ORM.**

## Ecriture de tests unitaires et fonctionnels (pas de test d’intégrations)

Pour cette recommandation si redéveloppement avec le Framework Symfony (SK PHP CNAM) PhpUnit sera utilisé. Faire des tests unitaires ou fonctionnels sans le Framework et sans réécriture est une solution longue et difficile à mettre en place sur un code morcelé ne respectant pas les standards MVC.

**Temps d’écriture des tests pour couvrir tout le code : 60 jours**

## Framework de logs

Framework de logs : (traçabilité) Mise en place d’une librairie de gestion des logs type « monolog », pour cette recommandation, l’implémenter sans Framework n’est pas une solution propre.

**Cf temps inclus : Remaniement pour arborescence type MVC**

## Sécurité / Authentification

* Sans Framework MVC: inclusion automatique de fichiers de sécurité  
  Politique de mot de passe (renouvellement, nombre d’essais, minimum de caractères)
* Passage en HTTPS
* Traçabilité applicative
* Fichier de configuration au format yaml et non accessibles par le navigateur

**Temps : 20 jours**

## Documentation (installation, utilisateur)

**Temps : 10 jours**

## Recette et Assistance à MOA

**Temps : 30 jours**