Cassini2
Documentation technique

Frédéric Levavasseur

Stéphane Trainel

October 18, 2007

L'AIFE est le service à compétence nationale Systèmes d'information budgétaire, financière et comptable de l'Etat institué par le décret n°2005-122 du 11 février 2005.

Contents

1 Introduction

Le document présente les éléments de conception de l'outil Cassini2, application de cartographie du système d'information financière de l'Etat.

2 Exigences

2.1 Permettre la saisie des données

L'outil doit permettre la saisie des informations sur les applications du système d'information financière collectées lors des campagnes successives AMM-AMF.

2.2 Développer un outil simple d'utilisation

L'outil de cartographie doit rester simple d'utilisation et modulaire. Sa livraison doit pouvoir se faire par étape.

- Aucune formation particulière à la manipulation de l'outil n'est prévue.
- L'outil doit permettre une saisie simplifiée des informations.
- L'outil doit restituer rapidement l'information synthétisée.

2.3 Suivre l'évolution des applications

L'outil doit permettre de suivre les évolutions des applications des ministères dans le temps.

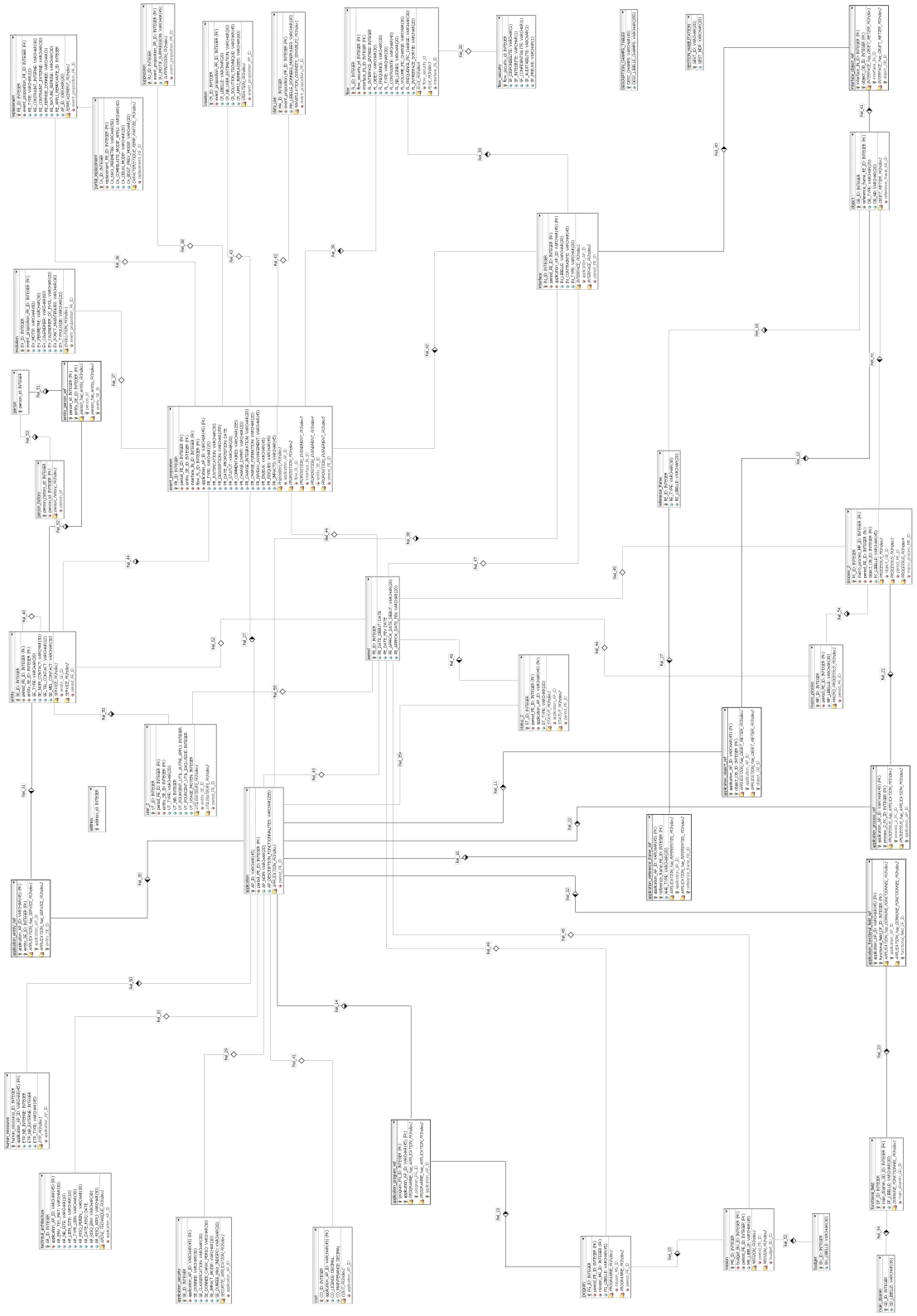
3 Sécurité

L'accès à l'outil est géré grâce à des habilitations qui reposent sur l'authentification faite par Windows. L'utilisation de ces identifiants Windows permettra d'alléger la gestion des utilisateurs.

Les habilitations respectent trois règles.

- Par défaut, un accès de type *lecture* est ouvert à tous les identifiants. Ces identifiants ne sont donc pas référencées dans les habilitations.
- Un accès de type *écriture* est ouvert, sur demande, à des identifiants concernés par les mises à jour.
- Un accès de type *admin* est ouvert à deux ou trois identifiants pour permettre la gestion des accès de type *écriture*.

4 Modèle de données



5 Architecture

5.1 Description des services d'hébergement

Pour l'exploitation de Cassini, plusieurs services d'hébergement sont à mettre en place : un service de production géré par la Bureautique, un service de pré-production et de développement pour le développement et les tests. Chacun de ces services est doté d'un hébergement PHP et d'un hébergement PostgreSQL. Des services de surveillance et de sauvegarde seront précisés ultérieurement.

5.1.1 Le service de développement

Le service de développement est basé sur l'atelier de développement proposé sur admisource (`www.admisource.gouv.fr`). Il est mis à disposition des développeurs pour leur permettre de développer l'application Cassini dans un environnement proche à celui de la production. Cet atelier propose de nombreux outils de développement ; notamment

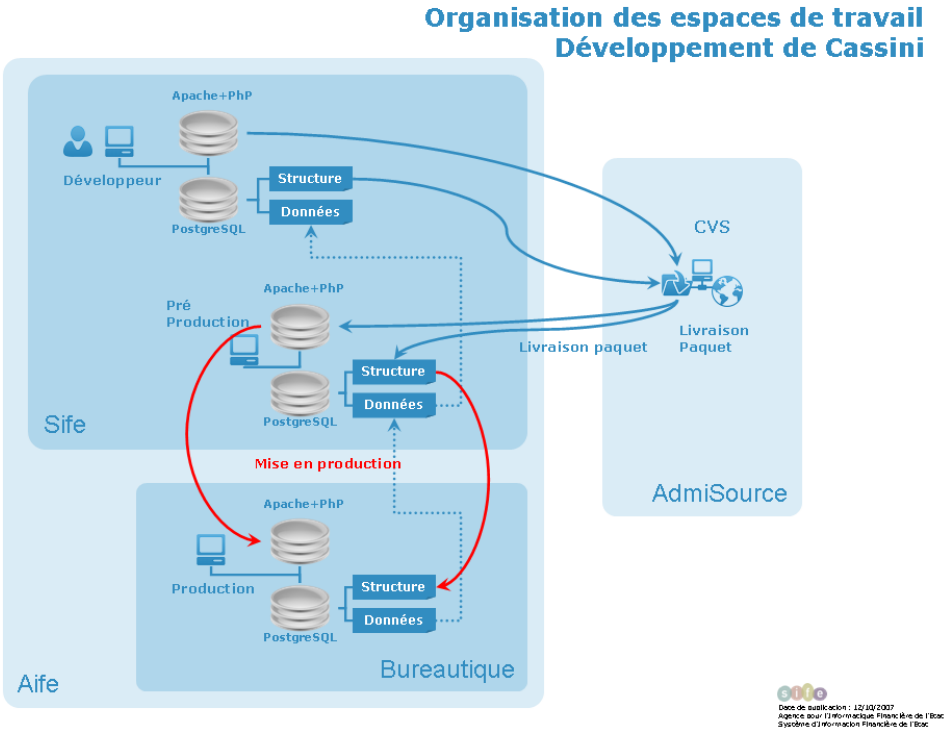
5.1.2 Le service de pré-production

Le service de pré-production est mis à disposition pour permettre aux développeurs de tester l'application dans un environnement identique à celui de la production. La plateforme de pré-production est exploitée par le Sife. L'étape de pré-production accélère et sécurise le processus de passage en production.

La plateforme de pré-production met à disposition un espace de stockage de fichiers et une base de données PostgreSQL avec les services d'administration correspondants. La base de données et les fichiers de l'espace de pré-production sont alimentés directement par les développeurs.

5.1.3 Le service de production

Les développeurs sont privés de tout accès au service de production. Celui-ci est alimenté uniquement par le service de production. De même que pour la pré-production, la plateforme de production met à disposition un espace de stockage de fichiers et une base de données PostgreSQL avec les services d'administration correspondants. Contrairement à l'espace de pré-production, l'espace de production ne peut être administré que par le service production.



5.2 Utilisation d'AdmiSource

AdmiSource (`http://admisource.gouv.fr/`) est la plate-forme collaborative proposée à l'ensemble des administrations françaises pour leurs développements de logiciels libres. Le projet Cassini s'inscrit dans cette démarche.

Les outils mis en oeuvre pour le développement de Cassini sont :

- Le suivi des bugs et des améliorations
- La gestion des documents : documentation technique et utilisateur
- La gestion du versioning avec CSV
- La gestion des livrables
- L'information par la liste de diffusion

5.3 Outils de développement

5.3.1 Atelier de développement Insee

Un atelier de développement est disponible sur AdmiSource : `http://admisource.gouv.fr/projects/atelier-dev/`. Il met à disposition une collection complète d'outils, notamment MySQL, Apache, Python, PhP, DBDesigner, etc.

5.3.2 Propel

`http://propel.phpdb.org/trac/`

5.3.3 Smarty

`http://smarty.php.net/`

5.4 Les surveillances

La surveillance de la plateforme de production consiste à vérifier les points suivants :

1. présence de la machine de production sur le réseau
2. espace disque utilisé inférieur à 90% de l'espace total
3. réponse du serveur HTTP
4. réponse du serveur PostgreSQL de production

La procédure de prise en compte et de gestion des incidents ainsi que l'information des différents acteurs s'inscrit dans la démarche bureautique.

5.5 Les sauvegardes

Deux types de sauvegardes sont mises en oeuvre :

- Sauvegarde du système : sauvegarde hebdomadaire de l'ensemble des données et des fichiers du système d'exploitation et du socle. Les sauvegardes sont conservées un mois.
- Sauvegarde des applications : sauvegarde quotidienne de l'ensemble des données et des fichiers des applications en production et en pré-production. Les sauvegardes sont conservées un mois.

Les restaurations seront effectuées sur la demande du responsable de l'application auprès de la bureautique.

6 Normes de développement

Ces normes sont inspirées des normes en vigueur à l'Insee pour le développement d'applications PHP.

Afin d'assurer la qualité et le bon fonctionnement des applications PHP à mettre en place sur la plateforme de production, des règles et des recommandations de développement sont nécessaires. Les normes de développement concernent le code et la structure des programmes eux-mêmes plutôt que leur empaquetage.

6.1 Pages HTML valides

Les pages HTML générées par le PHP doivent respecter la forme d'un des standards HTML courants. Plus précisément, cela signifie :

- la présence d'un DOCTYPE en tête de document
- l'annonce de l'encodage du document dans une balise meta de la section head
- le document doit être conforme au DOCTYPE annoncé (balises correctement imbriquées notamment)

L'intérêt est d'une part d'assurer le bon rendu des pages par un navigateur standard (IE 6.0, Firefox ou autrre), d'autre part de permettre le parcours automatique des pages générées par des outils tiers (moteur d'indexation par exemple). Une page ne vérifiant pas ces points ne s'affichera probablement plus correctement dans les navigateurs à venir. Il est fortement recommandé de faire cet effort jusqu'au bout au moins sur la page de garde de l'application, ne serait-ce que pour se familiariser avec les techniques en jeu. À titre de référence, une traduction de la recommandation HTML 4.01 est disponible à l'URL suivante: http://www.la-grange.net/w3c/html4.01/cover.html

6.2 Requêtes LDAP

6.3 Requêtes SQL et accès à la base de données

6.4 Mécanisme d'identification de l'utilisateur

6.5 Présence des commentaires dans le code

Pour faciliter la lecture et la maintenance du code de l'application, il est préférable que les sources soient abondamment commentés. Evidemment, toute application non triviale doit aussi fournir une documentation destinée au développeur final. L'outil Doxygen (`http://www.stack.nl/~dimitri/doxygen/`), présent dans l'atelier de développement, est particulièrement recommandé pour la rédaction de la documentation technique.

6.6 Application relogeable

Lors d'évolutions ultérieures du moteur d'exécution PHP, il est possible que l'organisation du système de fichiers de la plateforme change. Pour qu'une application puisse fonctionner correctement indépendamment des spécifications techniques de la plateforme sous-jacente, il faut que l'application soit *relogeable*, c'est-à-dire qu'elle puisse être déployée sur une autre plateforme sans modification du code. Cela implique notamment l'absence de chemins absolus. Dans ce but, les chemins absolus des plateformes de préproduction et production ne sont pas mis à disposition des développeurs.

6.7 Inclusion de fichiers

Pour éviter une inclusion cyclique de fichiers, on utilisera l'instruction *require_once* ou *include_once* plutôt que leurs équivalents moins sûrs *require* et *include*.

6.8 Bon fonctionnement de l'application

L'application doit réaliser correctement la tâche pour laquelle elle a été conçue, sans générer de *notice*, *warnings* ou *errors*. Elle doit être suffisamment robuste pour fonctionner en permanence dans l'environnement de production de la plateforme.

Remarque: un mécanisme d'authentification défectueux peut mettre en danger le fonctionnement d'une application, même en l'absence de volonté de nuisance des utilisateurs.

6.9 Normes de codage

6.9.1 Du bon sens

- Utiliser les balises longues <?php (et non les balises courtes <?, qui peuvent engendrer des conflits avec d'autres langages, comme XML)
- Toujours utiliser les accolades pour délimiter les boucles et instructions conditionnelles (ne pas utiliser de syntaxe raccourcie, généralement nuisible à la bonne lisibilité du code source).
- Nommer les objets (variables, fonctions, classes) de façon cohérente au sein de l'application (desNomsEnChameau ou des_noms_a_la_unix par exemple). Pour une cohérence avec Java, les nomsEnChameau sont préférés. Les noms des fonctions commencent par une minuscule ; les classes par une majuscule.
- Écrire le code HTML en minuscules (balises, attributs...)
- Proscrire les accents ou les caractères spéciaux dans les noms du langage.
- Donner, si possible, des noms signifiants en anglais aux objets, pour être cohérents avec les instructions du langage. Cela signifie aussi que les variables du type *$a*, *$tmp* et autres sont à bannir (sauf éventuellement pour les compteurs de boucle).
- Obliger les fonctions de test à retourner *TRUE* ou *FALSE*. Ne pas se baser sur les mécanismes de conversion de types implicite.
- Obliger les fonctions qui retournent un objet à retourner null quand aucun objet n'est trouvé.
- Dans les classes, simuler les membres privés par des noms commençant par _, par exemple *$_tab* pour un tableau privé.
- Faire réfléter aux noms de fichiers leur contenu. Une classe doit être isolée dans un fichier donné. Par exemple, le code de la classe *Requete* se trouvera dans le fichier *requete.class.php*.

7 Livrable

7.1 Arborescence

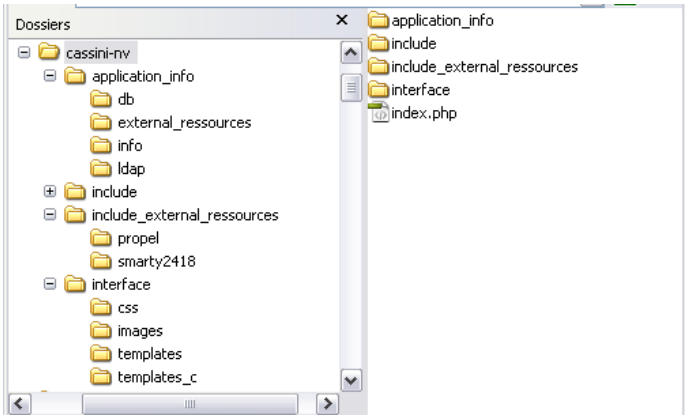
7.2 Noms des fichiers et des répertoires

Pour éviter des problèmes de compatibilité d'encodage entre les plateformes Windows et Linux, les noms de fichiers et de répertoires devront être composés exclusivement de lettres de l'alphabet latin, de tirets, de blancs soulignés, de chiffres et de points. En particulier, les lettres accentuées, espaces et caractères spéciaux sont proscrits.

Voici la liste des caractères possibles pour former le nom d'un fichier :

abcdefghijklmnopqrstuvwxyz
1234567890-_

Pour éviter des différences de comportement d'exécution entre Windows et Linux, il est obligatoire de n'utiliser que des minuscules dans les noms de fichiers et répertoires. Afin de distinguer les fichiers PHP produisant des pages HTML des fichiers ne contenant que des fonctions ou des paramètres, il est conseillé d'utiliser l'extension *.php.inc* pour les fichiers contenant des fonctions et paramètres. Cette extension ne



permet pas d'accès direct au fichier. Les fichiers **.php.inc* sont, de préférence, à placer dans un répertoire *include* à la racine du projet.

7.3 Destination du livrable

Le livrable destiné à passer en production est une archive zip contenant l'application et uniquement l'application. En particulier, la livraison de la documentation peut faire l'objet d'un livrable séparé.

Les fichiers et répertoires suivants ne doivent pas faire partie du livrable :

- Les répertoires CVS ou SVN
- Les fichiers de conflit CVS ou SVN
- Les projets eclipse ou autre
- Les fichiers de test
- Les fichiers de spécification
- Les fichiers de sauvegarde du type **.bak*, **.*~*

7.4 Contenu

Afin de pouvoir automatiser le passage en production, le fichier contenant les références d'accès à la base de données associée au projet est normalisé. Il s'agit d'un fichier nommé *db.php.inc* placé dans le répertoire : *nom_de_l_application/application_info/db*. Ce fichier doit suivre strictement le format suivant :

```
<?php
$bdd_host='nom_machine_ou_adresse_ip';
$bdd_port=3306;
$bdd_name='nom_base_de_donnee';
$bdd_user_standard='ps-nom_unix_projet';
$bdd_password_standard='xxx';
$bdd_user_admin='pr-nom_unix_projet';
$bdd_password_admin='xxx';
?>
```

7.5 Bibliothèques associées à l'application

Toutes les bibliothèques utilisées par l'application qui ne se trouvent pas dans le socle PHP doivent être livrées avec l'application dans le même livrable.

8 Liens utiles

- Application Cassini en pré-production : `http://cassini/`
- Atelier de développement : `http://admisource.gouv.fr/projects/cassini/`
- CVS
- Espace de pré-production
- Espace de production