Université de Caen Basse-Normandie

Rapport de stage

1 avril 2015 - 30 septembre 2015

# Open orchestra

*Auteurs :*
Amaury Lavieille

*Enseignants :*
François Rioult

9 août 2015

université de Caen
Basse-Normandie

# Remerciements

# Table des matières

# Introduction

Dans le cadre de ma deuxième année du master DNR2I (Master document numérique en réseau, ingénierie de l'Internet), j'ai effectué un stage chez Business & Decision au sein de l'agence Interakting à Caen.

Pour une durée de six mois, du 1er Avril au 30 Septembre 2015, j'ai été intégré au sein de l'équipe de développement du projet Open Orchestra.

Dans un premier temps, je vais vous présenter succinctement l'entreprise. Ensuite, je vous décrirais le projet Open Orchestra, son origine, ces particularités, etc. Enfin, je vous parlerais de l'organisation et des méthodes mises en place pour la réalisation du projet.

# Première partie

# L'entreprise

# Chapitre 1

# Présentation de l'entreprise

## 1.1 Business & Decision

Business & Decision est un groupe international spécialisé dans trois grand domaines, la Business Intelligence [1] (BI), la gestion de la relation client et enfin le e-business.

Le groupe a était créé en 1992 par Patrick Bensabat, aujourd'hui il est présent dans 15 pays et emplois plus de 2500 personnes.

Événement majeurs et évolution du groupe depuis 1992 :

**1992** : Création de Business & Decision par Patrick Bensabat autour de projets de Business Intelligence

**1993-1996** : Mise en place des premiers Datawarehouses [2] et applications de prévisions financières

**1999-2000** : Création de la division CRM (Gestion de la relation client)

**2002-2003** : Acquisition en Grande-Bretagne et au Benelux. Création d'agence régionales en France

**2004** : Acquisition en Suisse et aux Pays-bas et implantation en Tunisie

**2005** : Implantation aux Etats-Unis et acquisition de Metaphora [3]

**2007-2008** : Création de la marque Interkating

**2011** : Déploiement d'offres Cloud et Mobilité. Inauguration du Datacenter éco-responsable d'Eolas

**2013** : Lancement de Datalyse, programme de recherche en Big Data. Implantation au Pérou

**2014** : Création de Herewecan, agence de communication digitale, Lancement des pôles d'expertise Big Data, Transformation et Hub Mobile )

1. La Business Intelligence ou l'informatique décisionnelle représente les différentes solutions informatiques qui permettent l'exploitation des données du entreprise dans le but de faciliter la prise de décision

2. Les Datawarehouses ou entrepôt de données désigne une base de données utilisée pour stocker et structurer des données de production d'une entreprise afin de fournir des informations stratégiques pour la prise de décision

3. Métaphora est une société de service spécialisée dans la conduite du changement. Elle intervient dans toutes les étapes d'un projet pour faciliter l'appropriation du futur système d'informations par les utilisateurs finaux.

## 1.2 Interakting

Interakting est l'agence web du groupe B&D composé de 340 employés répartis principalement entre Caen et Paris, les équipes de l'agence interviennent sur des projets web de grande envergure (Canal +, Bnp Paribas, Inra, Moët Hennessy, PSA Peugeot Citroën, ...).

Les différents projets de l'agence sont développés avec différent outils : eZ publish (système de gestion de contenu créé par l'entreprise eZ Systems As), PHP Factory (plateforme réalisé par conjointement par Interakting et Zend Technologies) et de plus en plus avec Symfony2 (framework php écrit par SensionLabs) avec notamment Open Orchestra.

# Deuxième partie

# Open Orchestra

# Chapitre 2

# Le projet

## 2.1 Présentation

Open Orchestra est un CMS (Content Management System) open source en développement depuis un peu plus d'un an. Il est actuellement en version bêta, la sortie de la première version est prévus pour le mois de septembre.

Il est basé sur le framework PHP MVC Symfony 2 et MongoDB. Open Orchestra offre bien-sûr les fonctionnalités attendues par tous CMS : gestion de contenu, médiathèque, contribution de page, gestion de version, utilisateurs, worklow, rôles, multi-site, multi- langue, multi-device , etc.

La grande particularité d'Open Orchestra est son faible couplage au niveau du code, mais aussi au niveau des solutions techniques, c'est-à-dire que l'on peut facilement remplacer ou étendre un de ses composants. Je reviendrais plus en détail sur ce point dans une section ultérieure.

## 2.2 Architecture

Avec Open Orchestra le « Back-office » et le « Front office » peuvent être dissociés dans deux applications Symfony différente.

Ce découpage a deux nombreux avantages, tous d'abord un seul « Back-office » peux gérer plusieurs sites (« Front-office »), de plus cela permet de mettre « Back-office » et le « Front-office » sur des serveurs différents, l'unique condition est qu'ils doivent tous utiliser la même base de données.

Pour finir, Open Orchestra utilise une API RESTFull [1] pour accéder, gérer ces différentes entités (pages, utilisateurs, contenus, etc).

Le schéma 2.1 présente l'architecture d'Open Orchestra avec un « Back-office » qui gère plusieurs « Fronts ».

1. REST (Representational State Transfer) est un style d'architecture qui définie différentes réglés (ressource identifié unitairement, utilisation des verbes HTTP POST, DELETE, PUT, GET) pour accéder et manipuler des ressources

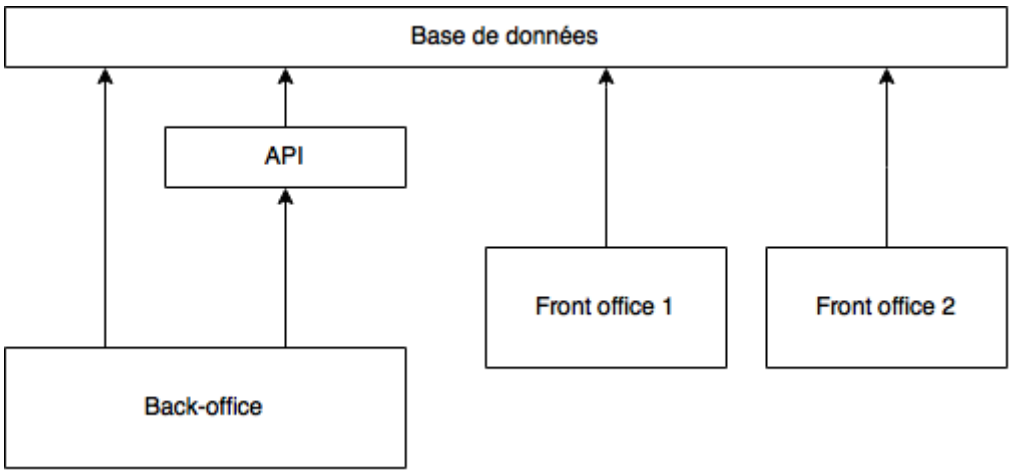


FIGURE 2.1 – Exemple d'architecture d'utilisation d'Open Orchestra

# Chapitre 3

# Fonctionnalités

Dans ce chapitre je vais vous présenter les différentes fonctionnalités d'Open Orchestra. Bien sûr, je ne vais pas tous les détailler, mais uniquement les points clés qui permettent une bonne compréhension du projet.

## 3.1 Blocs

Dans Open Orchestra tous les éléments visibles en front sont représentés par des blocs. Un bloc est simplement une entité avec des attributs qui varient selon le type de bloc. Chaque type de bloc sont indépendant, c'est-à-dire que eux seul connaissent leurs attributs, leurs façon de s'afficher en front ou en back-office ou encore la façon dont ils peuvent être contribué en back-office.

Pour permettre cette indépendance, Open Orchestra utilise le design pattern stratégie. Le pattern stratégie permet de rendre une famille d'algorithmes interchangeables et ainsi la possibilité d'exécuter un traitement spécifique selon le contexte.

Comme le présente le diagramme de classe 3.1 chaque bloc possède une classe qui indique comment il doit s'afficher pour cela la classe a deux méthodes, la première qui indique le bloc qu'elle supporte (`support`) et la méthode `show` qui fourni le rendu, d'un autre coté il y a une classe que appelé `manager` qui est la seule a connaitre toutes les stratégies des différents blocs.

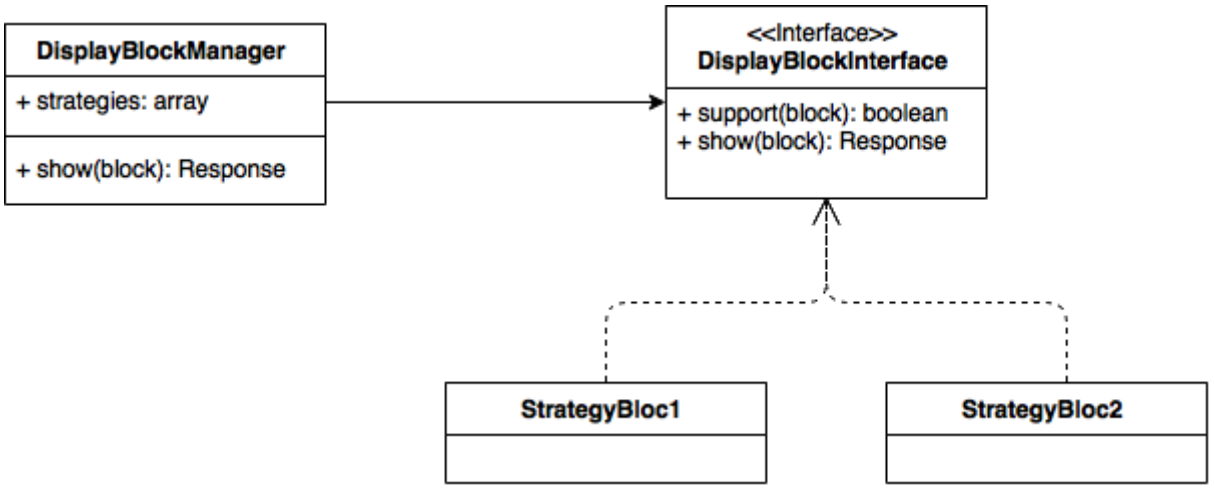


FIGURE 3.1 – Utilisation du pattern stratégie pour l'affichage des bloc en front

Ainsi, lorsque que l'on désire afficher un bloc, il suffit de demander au manager d'afficher ce bloc, ce dernier va chercher parmi les stratégies qu'il connait laquelle supporte (méthode `support`) le bloc et s'il en trouve une alors il exécute la méthode (`show`) de la stratégie.

Par défaut, le CMS propose de nombreux types blocs comme par exemple un bloc pour lister un type de contenu, afficher un texte formaté, afficher un carrousel ou un média, un menu, une carte, un bloc de contact, etc.

Bien-sûr, il possible pour un intégrateur [1], de créer son propre type de bloc, il lui suffit de développer les différentes stratégies (affichage en front et back-office, formulaire en back-office, etc) nécessaire à un bloc.

## 3.2 Nodes

Un des points central d'un CMS est les pages. Sur Open Orchestra les pages sont identifiées comme des noeuds (« nodes »). Les nodes sont simplement des conteneurs qui contiennent des zones. Les zones permettent d'organiser la page, elles contiennent des sous-zones ou des blocs ce qui permet un découpage fin de la page pour simplifier sont organisation.

Il existe trois types de nodes :

— Les nodes qui représentent les pages visibles en « front ».

— Les « node tranverse » qui contiennent les blocs transverses, c'est-à-dire les blocs qui sont communs à plusieurs pages comme le bloc « menu » ou encore le bloc « footer ».

— Le dernier type de nodes sont ceux qui permettent de contribuer les pages d'erreurs (404, 503) d'un site.

Le schéma 3.2 montre un exemple typique de page avec trois zones, le header qui contient un bloc menu, le footer un bloc footer et une zone principale découpé elle-même en deux zones avec un bloc de contact pour la zone de droite et un bloc de texte à gauche.

1. Dans le cadre de ce rapport un intégrateur indique une personne ou une équipe qui utilise Open Orchestra afin de l'étendre pour des besoins particuliers

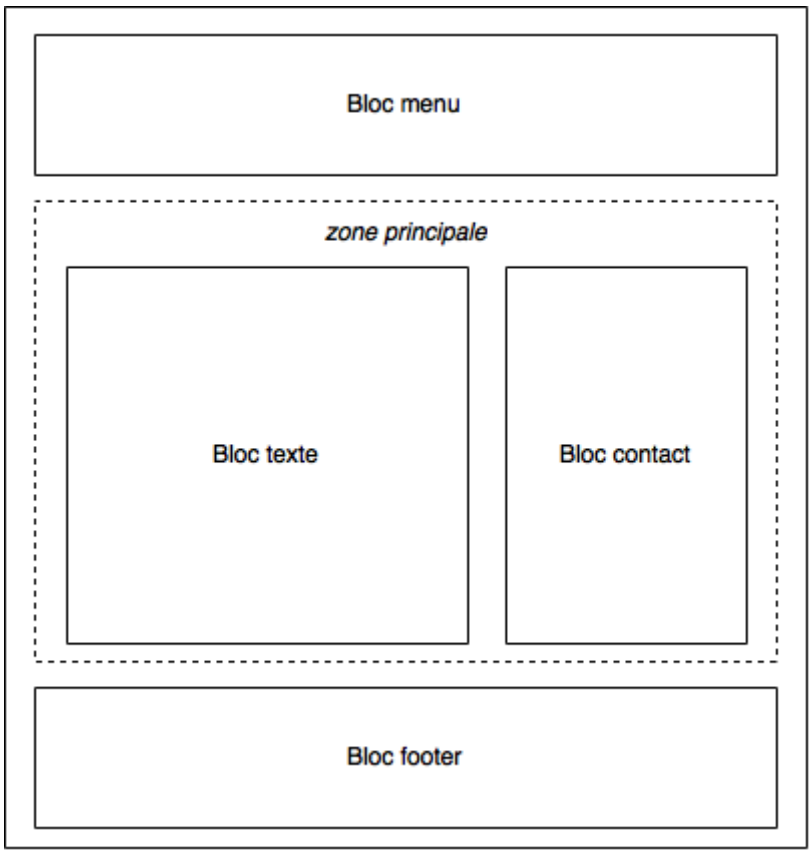


FIGURE 3.2 – Exemple de page

## 3.3 Content type

Le second point important d'un CMS est les contenus avec Open Orchestra il est facile de créer des types de contenus (actualité, client, etc, ...).

En effet, Open Orchestra propose une approche graphique grâce à un formulaire pour créer ou éditer des types de contenus comme l'illustre la figure. Un type de contenu est composé de différent champs avec différente option, par exemple pour un champ de type texte il peut y avoir une option pour limiter le nombre de caractère ou encore une valeur par défaut.

Par défaut, Open Orchestra offre une liste de types de champs (date, texte, monnaie, média, email, entier, zone de texte riche, etc) qui comme pour tous les composants d'Open Orchestra peut être étendu par d'autre type de champ personnalisé.

# Chapitre 4

# Caractéristiques

## 4.1 Performance

Open Orchestra a été développé pour supporter une charge de trafic importante. Pour cela le CMS exploite au niveau du front l'ESI (Edge Side Includes) couplé à un reverse proxy [1].

L'ESI est un langage de balisage HTML qui permet de diviser une page en différent éléments dont les rendus sont faits dans différentes requêtes par le serveur web. Ce qui permet d'avoir un cache HTTP sur les différents éléments et ainsi rafraichir seulement les éléments obsolètes de la page et non toute la page.

Comme nous l'avons vu dans la section **??** sur Open Orchestra les pages sont déjà découpées en différent blocs ainsi l'utilisation de l'ESI est adapté et permet une amélioration significative des performances.

Si nous reprenons l'exemple de notre page (schéma 3.2), le schéma 4.1 présente le processus effectué par le serveur dans le cas où les blocs `header` et `footer` sont déjà en cache lorsqu'un utilisateur demande la page.

1. Un reverse proxy est un serveur qui traite les requêtes en amont du serveur web. L'intérêt d'un reverse proxy est multiple gestion de cache, chiffrement, répartition de charge

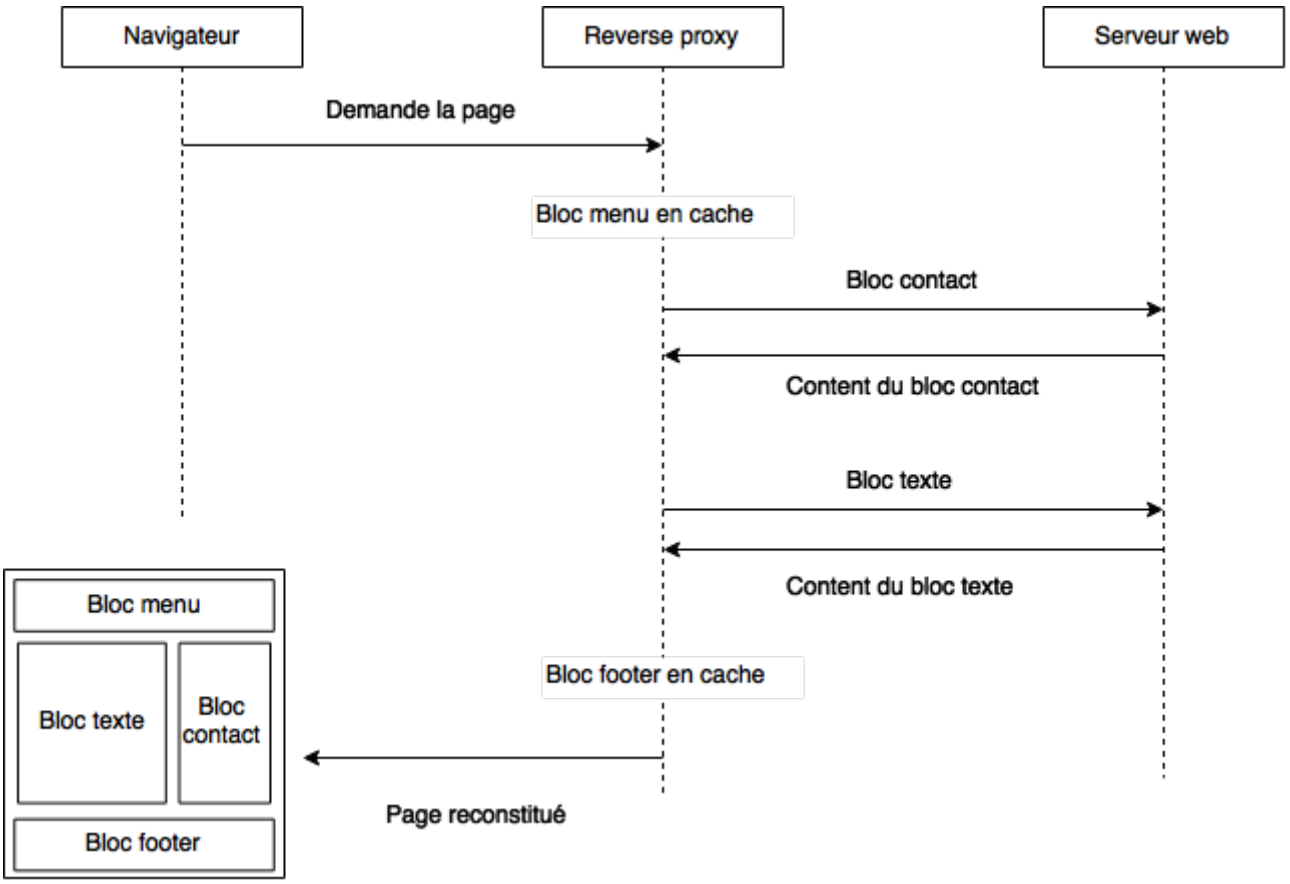


FIGURE 4.1 – Processus d'affichage d'une page qui utile l'ESI

## 4.2 Modularités

Comme je vous l'expliquais en introduction la principale particularité d'Open Orchestra est sa modularité. Pour cela, les différents composants du CMS sont répartis dans différents bundles [2]

- **open-orchestra-base-bundle** : Classes communes au back-office et front-office
- **open-orchestra-base-api-bundle** : Api d'Open Orchestra
- **open-orchestra-base-api-mongo-model-bundle** : Implémentation des models nécessaires à l'api pour `MongoDB`
- **open-orchestra-cms-bundle** : Logique du back-office
- **open-orchestra-front-bundle** : Logique du front-office
- **open-orchestra-display-bundle** : Logique d'affichage des blocs en Front-office
- **open-orchestra-media-bundle** : Médiathèque d'Open Orchestra
- **open-orchestra-media-admin-bundle** : Administration de la médiathèque d'Open Orhcestra
- **open-orchestra-model-interface** : Interfaces des models utilisés par les autres bundles
- **open-orchestra-model-bundle** : Implémentation des models pour `MongoDB`
- **open-orchestra-user-bundle** : Gestion des utilisateurs

2. Sous Symfony2 un bundle est un ensemble de fichiers structurés (contrôleur, entité commande, listener, formulaire) qui permettent d'implémenter une ou des fonctionnalités qui dans l'idéale peuvent être réutilisés dans différent projets

— **open-orchestra-workflow-function-bundle** : Système de workflow pour le back-office

Une application front-office n'a aucun intérêt à charger toute la logique du back-office ainsi grâce à ce découpage chaque application (front-office ou back-office) charge uniquement les composant qui lui est nécessaire.

De plus, cela permet de désactiver facilement une fonctionnalités. Par exemple, si un intégrateur n'a pas bessoin de gérer des médias il peut alors désactiver la médiathèque en n'utilisant pas les bundles (`open-orchestra-media-bundle` et `open-orchestra-media-admin-bundle`).

Un autre avantage est la facilité de remplacement d'un composants. Par défaut Open Orchestra utilise `MongoDB` comme système de gestion de base de données, mais si un intégrateur veut utiliser un autre système comme `Mysql` alors il lui suffit d'écrire son propre `open-orchestra-model-bundle` qui utilise bien-sûr les différentes interfaces de `open-orchestra-model-interface` adapté a `Mysql`.

# Troisième partie

# Organisation du développement

Introduction, pas eu de tâche particulière intégration au sein de l'équipe avec diff tâches (ajout fonctionnalités, correction bug, Documentation en aglais, refacto, optimisation, formation

# Chapitre 5

# Gestion de projet agile

## 5.1 Méthodes agiles

Le projet a été mené selon une approche agile. La gestion de projet agile repose sur un développement itératif qui consiste à découper un projet en plusieurs itérations appelé sprint. Les méthodes agiles définis plusieurs valeurs fondamentales :

- La communication au sein de l'équipe
- La collaboration avec le client, celui-ci doit être impliqué tous au long du développement.
- L'adaptation au changement, c'est-à-dire que le planification initiale doit être flexible pour permettre l'évolution de la demande.

Il existe différentes type de méthode agile (Scrum, XP, RAD...) qui possèdent leurs propres caractéristiques.

Pour le développement d'Open Orchestra, c'est la méthode Scrum qui est utilisé. Scrum définis trois rôles au sein d'une équipe :

- Le « Product Owner (PO) » qui porte la vision du produit
- Le « Scrum Master » est le responsable de la mise en œuvre de la méthode, il doit s'assurer que cette dernière est correctement appliqué
- L'équipe de développement qui est chargé de transformer les besoins exprimés par le Product Owner.

La réalisation d'un projet utilisant la méthode est rythmée par différents évènements :

Tous d'abord, le sprint qui est représente une période de courte durée, de une à quatre semaines, durant laquelle l'équipe effectue un nombre de tâches définis à l'avance.

La réunion de planification où l'équipe répond à deux questions, « Quoi ? » c'est-à-dire les tâches du backlog [1] qu'elle réalisera au prochain sprint et « Comment ? » c'est a ce moment que l'équipe estime les tâches choisis au « Quoi »

Le « daily scrum » qui est une réunion réalisé quotidiennement. Durant cette réunion chaque membre de l'équipe indique les tâches qu'il a réalisé depuis le dernier daily et celles qu'il va effectuer jusqu'au prochaine daily et pour finir les difficultés qu'il a ou pense rencontrer. Cette réunion permet à tous les membres de connaître les tâches de chacun afin de s'entraider et d'anticiper plus facilement les obstacles.

1. Le backlog est un ensemble de fonctionnalités ou de tâches nécessaire pour la réalisation satisfaisante d'un projet

Pour finir à la fin du sprint, les membres de l'équipe se réunisse pour la revue de sprint durant laquelle les différentes tâches réalisés durant le sprint sont présentés et validés. Puis pour la rétrospective qui permet de mettre en évidence les points positif et les actions a mettre en place pour améliorer le prochain sprint.

SCHEMA (schéma sprint)

## 5.2 Scrum avec Open Orchestra

MIse en place au sein de l'équipe (Trello, cérémonie, dayly) détaille des colonnes trello Céremonie toutes les semaines, Daily tous les midis (indique ce que l'on a fait et que ce compte faire permet de savoir qui fait quoi en permancence, prévenir si l'on voit un problème) Workflow d'un ticket (colonnes backlog , TODO, Doing, ToValidate => Faile, Done)

# Chapitre 6

# Intégration continue

Git (Worflow, tag, travis) Utilisation de Git , présentatation rapide de git et github Systémé de PR avec validation par l'équipe. Pour eviter au maximun les régréssion utilsiation de travis description plus tard) Plate forme d'integration (la ou valide le PO), un déploiement plusieurs fois par jour sur l'environnement prod) utililisation capistrano ( présentation de l'outil) Deploiement (capistrano)

# Chapitre 7

# Qualité du code

PSR2 (qu'est ce que c'est, présentation) code climate (outil qui vérifie la qualité du code et fournis une note) SensioInsignt (Outil de qualite d'application Symfony) Travis, (Détail a quoi ça sert, comment cela fonctionne (hook github lancement des test) Test unitraite, Test fonctionel, (Mise en place de test pour tous les codes metiers) Behat (mise en place de test behat)