 DOCUMENT DE TRAVAIL

ExTAC

PÔLE QUALITE

Plan d’intégration

26 janvier 2015

Maxime Mangel – Lionel Malloggi – Vincent Receveur – Julien Boisseau



Table des matières

Introduction 2

1.1. Document 2

1.2. Tableau des versions 2

1.3. Objet du document 3

1.4. Domaine d’application 3

1.5. Responsabilité et évolution du document 3

1.6. Organisation du document 3

1.7. Caractère particulier du document 3

Le projet Cap Co Boost 5

2.1. Le contexte 5

2.2. Objectif 5

2.3. Objectif de la solution 5

2.4. Limites du projet 5

2.5. Livrables du projet 6

2.6. Méthodologie appliquée au projet 6

Environnements 7

2.1. Limites du projet 7

# Introduction

## Document

|  |  |
| --- | --- |
| Document | 2015-01-26-SCRUM-PlanDIntegration.docx |
| Etat | En cours. |
| Auteur(s) | Maxime Mangel – Lionel Malloggi – Vincent Receveur – Julien Boisseau |
| Niveau de confidentialité | Interne |

## Tableau des versions

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Commentaire | Auteur(s) |
| 1.0 | 26.01.2015 | Création du document | Maxime Mangel |

## Objet du document

Le présent plan d’intégration est élaboré pour le projet Cap Co Boost de la société Cap Cohérence. Il décrit l’ensemble des mesures, processus, outil et environnement mis en place afin d’assurer l’intégration du projet. Il définit quels sont les délais planifiés en interne pour la livraison produit. Il présente également les différents environnements utilisés avec leur spécification technique ainsi que les méthodes appliqués par le pôle technique pour collaborer à plusieurs sur le projet.

Ce document est à utiliser tout au long du projet, à cet effet, il sera revu en fonction des nouveaux éléments ou à la demande d’un acteur du projet.

## Domaine d’application

Le plan d’intégration est à appliquer sur l’ensemble du projet concernant la partie technique au sens global :

* Management de la livraison produit
* Mise en relation des différentes parties développées

## Responsabilité et évolution du document

La création, modification, mise à jour et/ou évolution du présent document ainsi que sa mise en application relève de la responsabilité des pôles technique et transverse.

Toute demande de modification, amélioration, évolution du présent document devront être proposée à l’un des responsables de ces pôles par mail.

## Organisation du document

Le document est organisé de la manière suivante :

* le présent chapitre introduit le document ;
* le chapitre 2 présente l’aspect management du plan d’intégration
* le chapitre 2 présente l’aspect technique du plan d’intégration

## Caractère particulier du document

Ce document revêt un caractère confidentiel et contractuel.

Les destinataires de ce document s’engagent à ne communiquer aucun élément, partie ou information qu’il contient à des personnes physiques ou morales tierces sans accord préalable du Directeur de Projet ExTAC.

# Management

## 2.1 Lots de livraison

Chaque livraison de sprint regroupe une liste d'user stories ajoutant des fonctionnalités au produit que l'on nommera composant. La récupération des lots qui constituent l'ensemble des composants à intégrer sont définis par le sprint backlog géré par le product owner en accord avec les scrum master (MOE) et le client. La livraison de ces lots est défini par un planning découpé en sprint prévu initialement au début du projet par les scrums master qui garantissent des composants fonctionnels et validés.

## 2.2 Délais de livraison

La MOE (scrum master) est en charge de la gestion des délais de livraison des composants en accord avec le planning établi en début de sprint. Le planning est susceptible d'évoluer en fonction de l'avance / retard sur chaque sprint. Il peut évoluer à chaque revue de sprint en accord avec le client.

## 2.3 Planification des intégrations

La planification des intégrations des composants se base directement sur le planning des livraisons par sprint.

## 2.4 Planning initial des livraisons

## 

Planning 1: Revue de sprint S1

# Environnements

## Environnement de développement

Les environnements de développement sous présent sous Mac OSX ou Windows. Le système d’exploitation n’a pas d’importance car les technologies utilisées sont cross-plateforme.

Les technologies suivantes sont requises pour développer l’application :

* Ruby version 1.9.3-p545
* MySQL version 5.5.x
* NodeJS avec npm (gestionnaire de paquet)

Les gems suivantes sont requises pour le développement :

* turbolinks
* jbuilder version 2.0
* sdoc version 0.4.0
* digest

Les modules npm suivants sont requis :

* gulp version 3.8.10
* gulp-ruby-sass version 0.7.1
* gulp-livereload version 2.1.1
* gulp-concat version 2.4.1
* gulp-plumber version 0.6.6
* gulp-uglify version 1.0.1
* gulp-newer version 0.3.0
* gulp-cached version 1.0.1
* gulp-changed version 1.0.0

## Environnement de production

L’environnement de production requière les mêmes technologies, gems et modules npm que l’environnement de développement.

La différence est que le serveur utilise le système d’exploitation Unix : CentOS release 6.6 (Final)

# Outils

## Outils

Les outils de développement choisis sont :

* Netbeans 8.0.2 comme éditeur de code avec le plugin Rails
* Git comme gestionnaire de version
* SourceTree comme logiciel pour utiliser plus simplement Git

## Netbeans

Netbeans est un IDE (Integrated Development Environment) multi-plateforme écrit en java. Il est possible via des plugins de lui ajouter des fonctionnalités. Pour ce projet, on a simplement ajouter le plugin permettant d’utiliser rails.

## Git

Git est un outil de gestion de version. Il permet de collaborer à plusieurs sur un projet de façon indépendante et mettre en comment les développements personnels de chacun en alertant en cas de conflit.

Pour gérer les développements parallèles des développeurs, une branche sera crée par fonctionnalité de cette façon, on est capable de compartimenter les composants et garder un contrôle sur les tests à effectuer.

En cas de conflit, les développeurs sont autorisés à régler les conflits s’ils en ont les compétences et l’autorisation par leur Scrum Master. Dans tous les cas, si un problème survient lors du merge de fonctionnalités, alors ce sont les Scrum Master qui sont les responsables chargés de corriger ce problème.

Git permet de créer des Tags sur les branches utilisées. Ce système de tag permet de mettre en avant les différentes étapes d’un logiciel et sera utilisé lors de projet pour symboliser une version majeure du produit (qui correspond donc à une version livré en production ou en test au client).

# Processus d’intégration

1. Les développeurs reçoivent leurs tâches de développement par leur Scrum Master.
2. Une branche est créée par composant afin de cloisonner les développements. Il est possible de travailler à deux sur une branche, il y potentiellement des conflits à gérer.
   1. Développement des fonctionnalités
   2. Création des tests unitaires
3. Une fois le développement finis, le développeur crée un commit qui permet de sauvegarder l’état de son projet.
4. Ensuite, il push sa version sur une branche de pré-production en gérant les conflits si ils existent et que son Scrum Master lui donne son accord. Sinon, le Scrum Master résout le conflit pour lui.
5. Une fois le développement de plusieurs composants terminés, les Scrums Masters doivent fusionner ces derniers sur une branche de pré-production afin de tester les différents composants ensemble.
   1. Tests unitaires
   2. Tests d’intégration / fonctionnels
6. Une fois tous les tests passés, la version est mise en place sur le serveur de production à disposition du client. Ainsi, il peut lui aussi faire des tests et vérifier la correspondance avec ces attentes. En cas, de bug à remonter, il suivra le processus décrit dans le document concernant les « Retours de Bugs ».

Il important de noter que à n’importe quel moment du processus, on peux devoir reprendre depuis le début car on des bugs à gérer. Ou encore car on a mal découpés les tâches dans un premier temps et qu’il faut donc décomposer.

