

School ‘ In

Logo

Version

Historique des révisions

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Auteur** |
| 19/10/2015 | 1 | 1er regroupement de doc | Victor |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Sommaire

[Rôles et responsabilités 4](#_Toc415583632)

[Parties prenantes, rôles et coordonnées 4](#_Toc415583633)

[Organigramme de synthèse 4](#_Toc415583634)

[Modalités de déroulement du projet 5](#_Toc415583635)

[Gestion du board agile 5](#_Toc415583636)

[Evaluation des charges et calendrier cible 5](#_Toc415583637)

[Planification initiale 5](#_Toc415583638)

[Gestion du reporting 6](#_Toc415583639)

[Gestion des relations avec les parties prenantes 7](#_Toc415583640)

[Gestion de la documentation 8](#_Toc415583641)

[Description des livrables 8](#_Toc415583642)

[Règles de validation 8](#_Toc415583643)

# Rôles et responsabilités

## Parties prenantes, rôles et coordonnées

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom** | **Rôle au sein du projet** | **Mail** | **Téléphone** |
| Antoine GRANDIERE | Membre | agrandiere@intechinfo.fr | 06 31 07 23 77 |
| Thibaut MIGINIAC | Membre | tmiginiac@intechinfo.fr | 06 15 48 94 38 |
| Victor ANDRE | Membre | vandre@intechinfo.fr | 06 68 49 98 34 |
| Andrian KABOBO | Membre | kabobo@intechinfo.fr | 07 80 84 67 02 |
| Olivier SPINELLI | Suiveur | olivier.spinelli@invenietis.com |  |
| Antoine RAQUILLET | Suiveur | antoine.raquillet@invenietis.com |  |

## Organigramme de synthèse

# Modalités de déroulement du projet

## Gestion du board agile

Le board se trouve sur https://trello.com/schoolin1, la personne charger de le faire évoluer est Victor (administrateur). Le board évolue à selon l’avancer de l’équipe, chaque grande phase du projet est divisée en 4 catégorie Valider/A Valider/En Cours/A Faire qui regroupe les actions à réaliser et leur avancement.

## Evaluation des charges et calendrier cible

         Itération 0 : le 27/10/2015

         Itération 1 : le 23/11/2015

         Itération 2 : le 14/12/2015

         Itération 3 : le 18/01/2016

Jour homme 768h : 4 x 3 x4 x 16

Soit 128 jours (Total).

Soit 24 jours par personne.

## Planification initiale

-Itération (0)  de lancement de projet:

-Livrer des documents avant-projet

-Itération 1 :

-Création d’une école

-Gestion du budget

## Gestion du reporting

Chaque personne devra informer les personnes concernées (cela inclus toujoursle chef de

projet Victor André) de son avancés dans le projet via les outils de communication

tel que :

-**Trello** qui sera tenue à jour à chaque fin de tache (Contiendra toute sorte de fichier)

-Les **Mail** pour les informations importantes (Tout type de fichier seront envoyé)

-**Hugo** qui contiendra tous les fichiers important liés au projet (Documents texte et sources)

-**Git** pour tous les codes source du projet (Code source)

Le suiveur aura bien entendue axer a tout c’est outil de communication il pourra donc être a tout moment informé de l’avancée du projet.

## **Gestion des relations avec les parties prenantes**

L'équipe de projet et le chef de projet communiquent par des réunions physiques une fois par semaine pour faire le point sur l'avancement du projet. Le chef de projet mène la réunion de bout en bout, recueillant les remarques de chacun des membres de l'équipe. Un membre de l'équipe différent à chaque réunion se charge de la mise à jour des documents si besoin est.

L'équipe communique avec son suiveur via des réunions physiques pendant les cours ou par mail, si une difficulté apparaît.

## **Gestion de la documentation**

La documentation d’un projet a une importance primordiale : c’est l’outil de communication entre les membres de l’équipe et les intervenants extérieurs(membre des instances de pilotage, chef de projet, utilisateurs, etc…).Elle assure la pérennité des informations au sein du projet.

Afin d’organiser la gestion de la documentation produit par projet, il convient au préalable d’identifier tous les types de documents relatifs aux diverses étapes d’un projet. Nous nous sommes mis d’accord sur différents règles à respecter.

Ces règles s’articulent autour de plusieurs thèmes. Les plus courants sont :

* La dénomination et l’organisation des fichiers du code source ;
* Le style d’identification ;
* Les conventions de dénomination ou règle de nommage ;
* Les commentaires et documentations du code source ;
* Les recommandations sur la déclaration des variables ;
* Les recommandations sur l’écriture des instructions, des structures de contrôle et l’usage des parenthèses dans les expressions.

D’où, ces règles se déclinent comme suit :

* **Les règles facilitant la maintenance du code :**
* Les identificateurs des constantes doivent être composés de MAJUSCULES ou de caractères soulignés.
* Les identificateurs de classes doivent être écrits en CamelCase ( (littéralement : « casse de chameau »). C’est un terme anglais utilisé dans l'informatique. Elle est une pratique qui consiste à écrire un ensemble de mots en mettant en majuscule la première lettre des mots liés.
* Les identificateurs des méthodes doivent être écrites en Camelcase en commençant par une lettre minuscule.

Après avoir réalisé les livrables attendus tel que les aspects d’utilisation, les aspects conceptuels et méthodologiques qui seront mieux détaillés un peu plus bas   
 Nous mettrons les différentes taches réalisées sur Trello qui est basé sur une organisation des projets en planches listant des cartes, chacune représentant des tâches. Les cartes sont assignables à des utilisateurs et sont mobiles d'une planche à l'autre, traduisant leur avancement. Pour que l’ensemble de l’équipe ait une idée sur le travail de chaque membre du groupe.

A chaque fin de tache, nous mettrons les documents sur le serveur Hugo.Nous utiliserons git qui nous permettra de mieux gérer l’évolution du projet. Pour ce faire, Git possède deux structures de données : une base d'objets et un cache de répertoires. Il existe quatre types d'objets :

l'objet blob, qui représente le contenu d'un fichier (l'origine de cette dénomination est probablement à chercher dans les binary large objects des bases de données) ;

l'objet tree (mot anglais signifiant « arbre »), qui est une liste d'objets de type blobs et des informations associées à chaque blob, tel que le nom du fichier et les permissions. Cet objet décrit l'arborescence des sources à un instant donné ;

l'objet commit, résultant de l'opération du même nom (mot anglais signifiant « valider une transaction »3) et qui donne accès à l'historique d'une arborescence de source. Il contient un message de log, un objet arbre et pointe vers un ou plusieurs objets commit parents ;

l'objet tag (étiquette) qui est une manière de représenter un commit spécifique. Il est en général utilisé pour marquer certains commits, par exemple par un numéro ou un nom de version (2.1 ou bien Lucid Lynx).

La base des objets peut contenir n'importe quel type d'objets. Une couche intermédiaire, utilisant des index (les sommes de contrôle), établit un lien entre les objets de la base et l'arborescence des fichiers.

Chaque objet est identifié par une somme de contrôle SHA-1 de son contenu. Git calcule la somme de contrôle et utilise cette valeur pour déterminer le nom de fichier de l'objet. L'objet est placé dans un répertoire dont le nom correspond aux deux premières lettres de la somme de contrôle. Le reste de la somme de contrôle constitue alors le nom du fichier pour cet objet.

Git enregistre chaque révision dans un fichier en tant qu'objet blob unique. Les relations entre les objets blobs sont déterminées en examinant les objets commit. En général, les objets blobs sont stockés dans leur intégralité en utilisant la compression de la zlib. Ce principe peut rapidement consommer une grande quantité de place disque ; de ce fait, les objets peuvent être combinés dans des archives, qui utilisent la compression différentielle (c'est-à-dire que les blobs sont enregistrés sous la forme de différences par rapport aux autres blobs).

## Description des livrables

L’objectif est de documenter et tracer le travail réalisé (outils, méthodes…) qu’il s’agisse de traitement de données, de développement, ou d’hébergement.

* Concernant les aspects ‘conceptuels et méthodologiques’ :
* La description des données contenus ou traitées,
* Le dossier de spécification fonctionnelle et technique, les diagrammes de classes, les modèles de séquences décrivant et explicitant la conception et le fonctionnement de l’application.
* Les descriptions des différents cas d’utilisations pour les spécifications fonctionnelles.
* Concernant les aspects ‘installation et hébergement’ :  
   - Une notice d’installation
* Un mode opératoire pour tester l’applicatif
* Eventuellement un jeu de données pour test
* Concernant les aspects ‘utilisation’ :
* Une notice d’utilisation
* Une documentation pour les administrateurs de l’outil (l’administration de l’école)
* Concernant l’outil ou l’application:

Le ou les codes, avec code commenté ou API de documentation du code

## 

## Règles de validation

Les livrables doivent être clairement définis pour un résultat conforme à des normes de qualités.

* Pour les aspects ‘conceptuels et méthodologiques’ : les différents besoins fonctionnelles et non fonctionnelles doivent tous être réalisables, les classes doivent être correctement créées et les différents digrammes doivent être clairement explicite pour qu’on comprenne comment l’API fonctionne.
* Les notices expliquant l’installation et l’hébergement de l’application devront être faciles à comprendre de sorte qu’un client (une école) utilisant l’application ne rencontre pas de difficulté durant l’installation de l’API.
* L’utilisation de l’API devra être simple et ne pas poser de problèmes lors de son exécution.

* L’application devra tourner dans toute son intégralité, tous les cas d’utilisations devront être mis en place.