

Exécution de Mandats

Table des matières

1	Analyse du mandat par Use Cases et Scénarios	3
1.1	Use Cases (cas d'utilisation)	3
1.2	Scénarios	3
1.2.1	Identification	4
1.2.2	Détails	4
1.3	Modèle conceptuel de données (MCD)	6
2	Planification en mode Agile	8
2.1	Définitions	8
2.2	Principe général	8
2.3	Outils	9
2.4	Structure du Repository	10
2.5	Sprints	10
2.5.1	Premier sprint	10
2.5.2	Sprints suivants	10
2.5.3	Préparation	11
2.6	Projets	12
2.7	Tâches	12
2.7.1	Pièces jointes	13
2.7.2	Checklist	13
2.7.3	Etiquettes	14
2.8	Suivi	14
3	Tests	15
3.1	Niveau	15
3.2	Type	16
3.3	Stratégie	17
3.4	Résultats	17
4	Journal de Travail et Journal de Bord	19
4.1	Le Journal de Travail	19
4.2	Le Journal de Bord	19
5	Documentation de Projet	20
6	Communiquer sur l'avancement des travaux	21
7	Livraison	23
Annexe I.	Planification selon une approche plus « classique »	25
7.1.1	Planification initiale	25
7.1.2	Mise à jour	26

1 Analyse du mandat par Use Cases et Scénarios

« Rien ne sert de commencer à construire un escalier là où une échelle suffit. »

Avant de se mettre au travail, il est capital de s'assurer que ce que nous (mandataires) prévoyons de faire correspond aux attentes du mandant.

Pour valider notre compréhension du mandat, nous utilisons :

- la méthode des Use Cases / Scénarios¹
- le modèle conceptuel de données (MCD) si des données doivent être gérées.

1.1 Use Cases (cas d'utilisation)

Dans cette phase, il s'agit de bien déterminer à quoi va servir le système à réaliser.

Pour ce faire, nous complétons la phrase :

« On utilise [le système] pour ... ».

Exemple : si on nous demande de réaliser un smartphone simplifié, on peut dire :

1. On utilise un SmartPhone pour téléphoner
2. On utilise un SmartPhone pour envoyer et recevoir des SMS
3. On utilise un SmartPhone pour prendre des photos

Nous avons ainsi identifié trois use cases : « Téléphoner », « Envoyer et recevoir des SMS » et « Prendre des photos ». Au passage, nous avons également établi le fait que notre smartphone simplifié ne permettra pas de surfer sur Internet, puisqu'on n'a pas listé ce cas d'utilisation !

La liste des use cases doit être validée avec le mandant du projet !

1.2 Scénarios

Ensuite, nous détaillons chaque use case au moyen de plusieurs scénarios.

A ce stade, on doit s'imaginer en train d'utiliser le système fini. Chaque scénario raconte une histoire : celle d'une variante particulière d'un cas d'utilisation.

Exemple pour le cas d'utilisation « Téléphoner », on pourrait avoir les scénarios :

- « Appeler une personne listée dans mes contact »,
- « Retourner un appel en absence »,
- « Composer un numéro »,
- « Prendre un double appel »,
- etc...

¹ La forme plurielle 'Scénarios' est intentionnelle, la forme 'scénarii' n'ayant plus cours. (voir https://www.larousse.fr/dictionnaires/francais/scenario_scenarios/71355/difficulte)

Nous formulons un scénario au moyen d'une partie identification et d'une partie de détails.

1.2.1 Identification

Il s'agit d'une description courte du scénario.

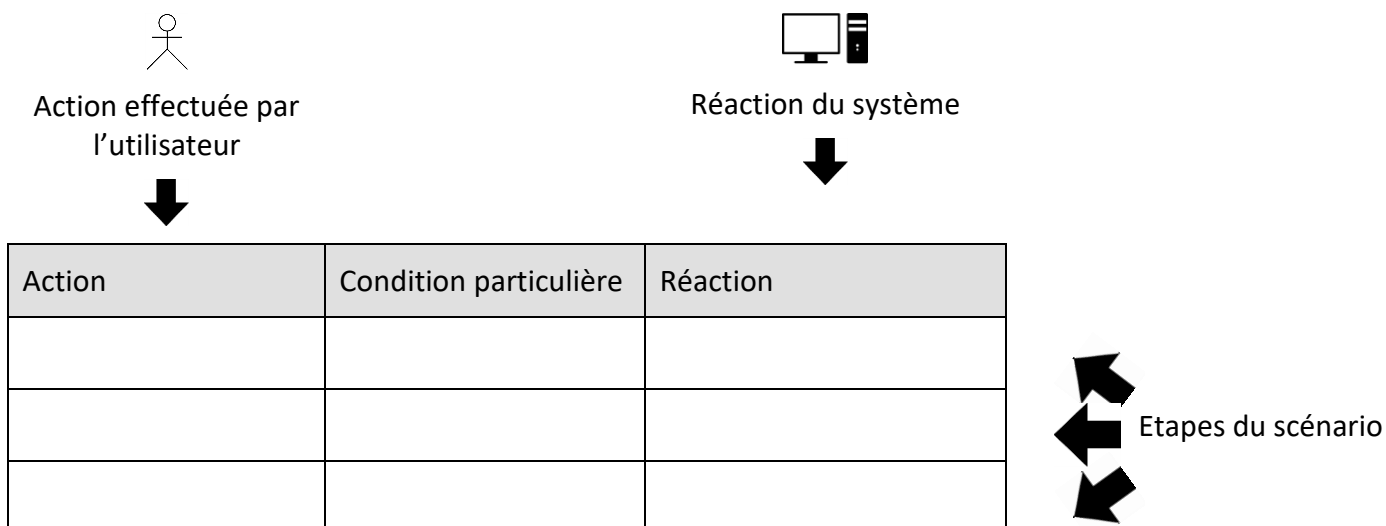
Identifiant + Titre		Un code unique à travers tout le projet + titre court
En tant que		Rôle
Je veux		Quelque chose
Pour		But
Priorité		Selon MoSCoW :

- Must = Vital
- Should = Essentiel
- Could = Confort
- Would = Luxe

1.2.2 Détails

Cette partie raconte étape par étape l'histoire de notre scénario.

Cela se fait dans un tableau à trois colonnes :



Les règles suivantes s'appliquent pour assurer une bonne formulation du scénario :

1. Dans la première colonne, nous ne faisons mention que de l'utilisateur et de l'interface du système, jamais du comportement du système ;
2. Dans la deuxième colonne, nous faisons mention d'une éventuelle condition sur l'utilisateur ou sur l'interface du système ;
3. Dans la troisième colonne, nous ne faisons jamais mention de l'utilisateur ;

4. Rappelez-vous que le scénario raconte une histoire ! Chaque case du tableau décrit quelque chose de précis ; il ne peut pas y avoir – entre autres – de « si » ;
5. Les actions peuvent faire référence à des schémas (maquette d'interface, architecture du système, schéma de réseau, ...) ;
6. On peut décrire une action pour laquelle le système n'a pas de réaction. Ceci afin de mieux raconter notre histoire ;
7. Une action peut engendrer plusieurs réactions.

Pour aider à la bonne compréhension du scénario, on l'accompagne de maquettes, auxquelles on fait référence dans la description des réactions.

Exemple :

Identifiant	SMP0012 – Appel d'un contact
En tant que	Utilisateur
Je veux	Téléphoner
Pour	Appeler une personne de mes contacts
Priorité	M

Action	Condition	Réaction
J'allume mon smartphone		L'écran s'allume et le système d'exploitation démarre
Je touche l'icône de l'application « téléphone »		L'application démarre (01)
Je touche le bouton « contact »		La liste des contacts personnels s'affiche (02)
Je touche le contact « John »		Les trois numéros de John s'affichent (portable, maison et travail) (03)
Je touche le numéro « maison »		Composition du numéro de domicile de John (04)
		Ça sonne
	John ne décroche pas dans les 30 secondes	L'appel se termine, on retourne aux trois numéros de John (03)
Je touche le numéro « portable »		Composition du numéro de portable de John (04)
		Ça sonne et John décroche
Je discute de mes vacances avec John		

Action	Condition	Réaction
Je touche le bouton « raccrocher »		La communication s'interrompt L'application affiche un résumé de l'appel: Personne appelée, date, heure, durée et coût de l'appel (05)
Je touche « OK »		L'application se referme

(Maquettes, dans un document séparé)

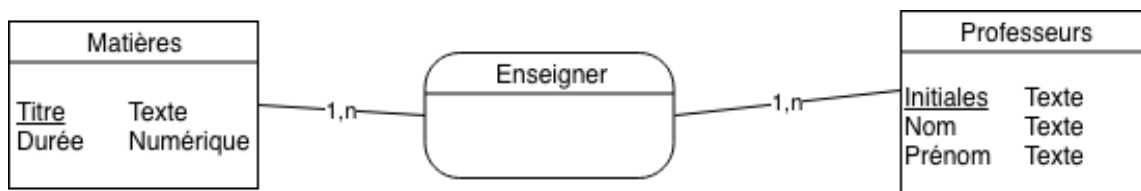


1.3 Modèle conceptuel de données (MCD)

Pour poursuivre notre analyse du projet et vérifier notre bonne compréhension du mandat, un MCD sera établi dans le cas d'un projet incluant la gestion de données. A noter : le fait qu'il y aie des données à gérer ne signifie pas nécessairement qu'il y a une base de données !

Bref rappel sur Le MCD² :

- Il est défini dans la langue du mandant.
- Il ne contient pas d'éléments techniques (clés primaires, tables de liaison, ...)
- Il sert à s'assurer que le système correspondra bien aux attentes du mandant
- La notation que nous utilisons est issue de la méthode Merise:



- Il est fréquemment nécessaire de donner des informations supplémentaires concernant les entités. Cela se fait de manière textuelle dans un document que l'on joint au MCD . Exemple :

² Voir support de cours du module ICT-104 pour plus de détails

Professeurs:

- Les initiales sont toujours formées de trois lettres : la première du prénom, puis la première et la dernière du nom

Matières:

- La durée est exprimée en nombre de périodes d'enseignement prévues

Le MCD – lorsqu'il y en a un – doit impérativement être validé avec le mandant. Le moment de la validation est un événement majeur du projet, qui doit absolument être consigné dans le journal de bord du projet.

2 Planification en mode Agile

Les **méthodes agiles** sont des groupes de pratiques de pilotage et de réalisation de projets. Elles se veulent pragmatiques, impliquant au maximum le demandeur (client) et permettant une grande réactivité à ses demandes. Elles reposent sur un cycle de développement itératif, incrémental et adaptatif et doivent respecter quatre valeurs fondamentales :

1. **Les Individus et leurs interactions** plus que les processus et les outils
2. **Des logiciels opérationnels** plus qu'une documentation exhaustive
3. **La collaboration avec les clients** plus que la négociation contractuelle
4. **L'adaptation au changement** plus que le suivi d'un plan

2.1 Définitions

Sprint (ou itération) : Intervalle de temps court (1 mois maximum, souvent appelé itération), pendant lequel la Dev Team va concevoir, réaliser et tester de nouvelles fonctionnalités. Suivant la nature du projet, le nombre de use case peut être assez grand, ce qui rend difficile le choix des fonctionnalités que l'on va inclure dans un sprint. Une méthode simple qui aide à la décision est celle de la comparaison par paires.

Sprint Review : Réunion de travail consistant à présenter aux parties prenantes les fonctionnalités terminées au cours du Sprint afin de recueillir leurs feedbacks et à faire le point sur l'avancement global du projet.

Kanban : Méthode basée sur le management visuel des tâches. On utilise un tableau des tâches pour suivre visuellement l'activité du Sprint (voir 2.5).

SMART : on dit qu'un objectif ou une tâche est SMART si il/elle est:

- **Spécifique** : l'énoncé est clair et précis. Quand on le lit, on sait ce que l'on doit faire. il ne définit qu'une seule chose à atteindre ou faire ;
- **Mesurable** : il est possible de vérifier de manière non ambiguë que la tâche est réalisée ou non ;
- **Ambitieuse** : la tâche représente un travail et une avancée du projet significatif ;
- **Réaliste** : la personne en charge de la tâche a les moyens et les compétences pour l'effectuer ;
- **Temporelle** : on a pu en estimer la durée, en heures.

Repository : (« dépôt » en français) il s'agit d'un stockage centralisé et organisé de toutes les données relatives à votre projet.

Backlog : l'ensemble des scénarios qui sont en attente de réalisation

2.2 Principe général

1. Il y a un repository par mandat, il suit la structure décrite ci-dessous en section 2.3

³ Le principe SMART est très répandu, mais pas standardisé. La définition donnée ici est propre au CPNV. SMART n'est pas spécifique aux méthodes Agile

2. On crée l'ensemble des tâches identifiables au début du mandat. A ce niveau, les tâches restent assez générales, elles s'affineront au fur et à mesure de l'avancement du projet. On utilise les 'Issues' de Github pour suivre les tâches effectuées.
3. On crée définit le nombre de sprints envisagés dans le projet. On crée un projet Github par sprint, ce qui nous permet d'énoncer la **planification initiale** du projet. Exemple :

Sprint 1  <small>Updated on 20 Nov 2018</small>	<ul style="list-style-type: none"> Analyse / Conception globale de l'application Création des use cases de l'application Modélisation de la base de données Début de la documentation Echéance : 20.06.2018
Sprint 2  <small>Updated on 1 Oct 2018</small>	<ul style="list-style-type: none"> Création des scénarii liés à la partie "calculs" Mise à jour du modèle de données Création des maquettes graphiques pour les calculs Création de la partie graphique pour les calculs Gestion des opérations Documentation liée à la gestion des opérations Echéance : 29.06.2018
Sprint 3  <small>Updated on 1 Oct 2018</small>	<ul style="list-style-type: none"> Création des scénarii pour le graphe de fonctions Mise à jour du modèle de données Création des maquettes graphiques pour le graphe de fonctions Création de la partie graphique pour le graphe de fonctions Gestion du graphe de fonctions Documentation liée à la gestion du graphe de fonctions Echéance : 10.07.2018
Sprint 4  <small>Updated on 3 Oct 2018</small>	Réalisation de la partie "graphe de fonctions" <ul style="list-style-type: none"> Création des scénarii Mise à jour du modèle de données Implémentation et tests Documentation Echéance : 10.07.2018
Sprint 5  <small>Updated on 3 Oct 2018</small>	Réalisation de la partie "Calculs" : <ul style="list-style-type: none"> Création des scénarii Mise à jour du modèle de données Implémentation et tests Documentation Echéance : 29.06.2018

4. On détaille le premier sprint (voir 0)
5. On se lance dans les itérations, durant lesquelles on maintient notre planning à jour (voir 2.8)

2.3 Outils

Nous gérons la planification dans Github selon le mode Agile. Vous devrez donc avoir créé un compte sur la plateforme Github, avec votre adresse email du CPNV. Choisissez un nom d'utilisateur qui permette de vous reconnaître (« JeanDujardin » par exemple, mais pas « RobinHood239 »).

Nous utiliserons ici un mode « Agile » simplifié, nous définirons uniquement des sprints, des sprints reviews, des issues et la méthode « Kanban ».

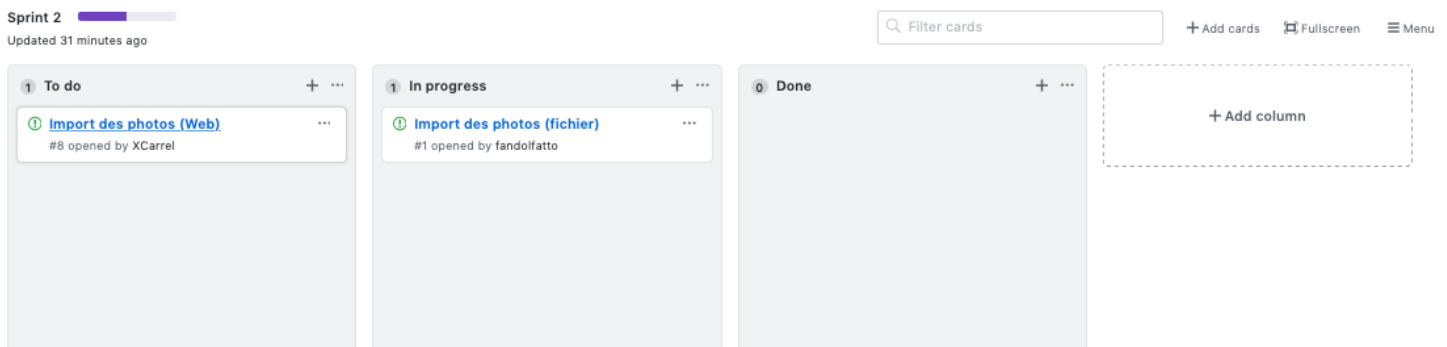
2.4 Structure du Repository

Le repository contient à sa base deux dossiers et fichier 'Readme.md' :

- Un dossier contenant résultat du travail demandé. Il peut s'agir de code source, de scripts, de fichiers de données, etc... Bref : ce que le mandant espère recevoir.
- Un dossier nommé 'doc' qui contient tous les fichiers qui ne servent qu'à documenter le projet : cahier des charges, document de projet, documentation de produit tiers utilisé, exemples, etc...
- Le fichier Readme.md qui :
 - Présente brièvement le projet
 - Explique comment remettre en place l'environnement nécessaire pour continuer le projet
 - Explique comment exploiter le résultat

2.5 Sprints

Chaque sprint (projet Github) se présente sous la forme d'un Kanban simple:



Cliquer sur 'Add cards' en haut à droite pour faire apparaître la liste des issues et pouvoir drag-dropper les tâches dans les colonnes.

2.5.1 Premier sprint

Le sprint 1 contient au minimum :

- L'établissement avec le client de la liste des Use Cases et scénarios du projet. ATTENTION : à ce stade on ne détaille pas encore les scénarios, on ne fait qu'en établir la liste (la table des matières en quelque sorte).
- La création du backlog, en créant une tâche d'implémentation pour chaque scénario identifié.

2.5.2 Sprints suivants

En règle générale, chaque sprint doit contenir les tâches suivantes :

- Création des scénarios et maquettes du sprint (et leur validation naturellement)
- Exécution et documentation des tests
- Création de la release
- Préparation du sprint suivant
- Sprint review. Dans le commentaire de cette tâche, on ajoute une checklist qui contiendra la liste des points qui seront formellement abordés et vérifiés. Le chef de projet les coche durant la review.

2.5.3 Préparation

Avant de pouvoir terminer le sprint X, il faut préparer le sprint X+1 (sauf bien sûr si le sprint X est le dernier du projet). Préparer le sprint X+1 consiste à :

- Sélectionner dans le backlog les prochaines tâches à effectuer et à les placer dans la colonne « ToDo » du sprint X+1
- Créer les issues pour les tâches 'habituelles' d'un sprint et les mettre dans la colonne « ToDo » du sprint X+1
 - Analyse détaillée
 - Faire la release sur Github
 - Exécution de tests
 - Préparation du sprint X+2
 - Sprint review
- Décomposer une tâche existante en deux ou plusieurs tâches SMART au besoin.
- Reporter des tâches du sprint en court qui n'auraient pas été réalisées
- Ajouter des bugs découverts dans le sprint en court

Exemple : On a le projet suivant

Sprint 1 🕒 Updated 22 seconds ago 	<ul style="list-style-type: none">• Analyse et conception globale du projet• Modélisation des données de l'application• Use cases• Echéance : 08.09.2018	...
Sprint 2 🕒 Updated 12 days ago 	<ul style="list-style-type: none">• Import des recettes et des photos de recettes• Echéance : 18.09.2018	...
Sprint 3 🕒 Updated 5 minutes ago 	<ul style="list-style-type: none">• Affichage des recettes et des photos• Echéance : 28.09.2018	...

On est donc en cours de sprint 2. La préparation du sprint 3 donne quelque chose comme ceci :

The screenshot shows a GitHub project board titled 'Sprint 3' with a subtitle 'Updated 6 minutes ago'. At the top right is a search bar labeled 'Filter cards'. The board is divided into three columns: 'To do' (8 items), 'In progress' (0 items), and 'Done' (0 items). The 'To do' column contains the following items:

- Finaliser le scénario 'Affichage de la recette pas-à-pas' (#16 opened by XCarrel)
- Finaliser le scénario 'Affichage de la fiche de recette' (#13 opened by XCarrel)
- Erreur d'importation de fichiers de plus 1Mb (#17 opened by XCarrel) with a red 'bug' label
- Gestion des recettes (#11 opened by XCarrel)
- Faire la release sur Github (#15 opened by XCarrel)
- Exécution de tests (#18 opened by XCarrel)
- Préparer le Sprint 4 (#14 opened by XCarrel)
- Sprint 3 review (#10 opened by XCarrel)

2.6 Projets

The screenshot shows the GitHub navigation bar with the following tabs: '<> Code', 'Issues 5', 'Pull requests 0', 'Projects 4' (which is highlighted with an orange bar), 'Wiki', and 'Insights'.

Dans la description, on y indique le ou les but(s), ainsi que la date d'échéance. On y ajoute les tâches que l'on s'engage à réaliser dans le sprint.

Dans l'onglet « Projects », cliquer sur le bouton « Create a project ». Indiquer un nom de projet, qui doit être « **Sprint** » suivi du **n° du sprint** et éventuellement d'un titre. Par exemple « Sprint 4 - Backups ». Indiquer **obligatoirement** une description pour décrire le sprint. Dans la description, y indiquer la date d'échéance (exemple : Echéance : 12.09.2018). Choisir le template « Basic kanban », donc à trois colonnes ('ToDo', 'In Progress', 'Done')

Les issues / cartes définies pour le projet apparaissent à droite. Les drag and dropper lorsque vous souhaitez les traiter dans le projet nouvellement créé.

Au fur et à mesure qu'elles sont en cours de traitement, déplacer-les dans la colonne « In progress ». Lorsqu'elles sont traitées, déplacer-les dans la colonne « Done ».

2.7 Tâches

Nous définissons les tâches au moyen d' « issues » de Github.

The screenshot shows the GitHub navigation bar with the following tabs: '<> Code', 'Issues 5' (which is highlighted with an orange bar), 'Pull requests 0', 'Projects 4', 'Wiki', and 'Insights'.

Les tâches détaillées suivent les recommandations suivantes :

1. Elles sont SMAR, sans 'T' ! La composante temporelle est fournie par le sprint dans lequel la tâche est placée
2. Les plus courtes possibles (max 4-5 heures si possible)
3. Inutile de préciser les tâches faites en parallèle à petite dose (tenue du journal de bord, documentation...), mais en tenir compte dans l'estimation des autres tâches.
4. Prévoir, en revanche, une tâche de finalisation de la documentation (relecture, correction des fautes, brochage), qui prend souvent quelques heures.

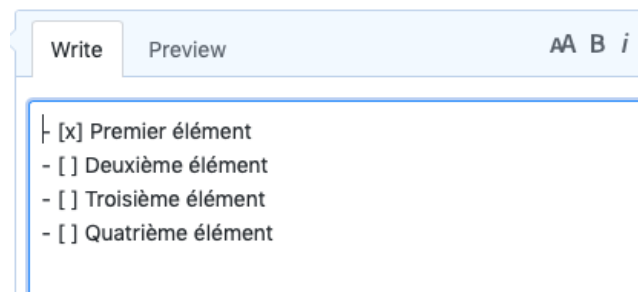
Indiquer un titre et une description.

2.7.1 Pièces jointes

Vous pouvez ajouter des documents (word, pdf, images) en les « drag and droppant ». Ces documents peuvent être, par exemple, les maquettes de votre projet, un diagramme de classes, un schéma de votre réseau, un modèle de données...

2.7.2 Checklist

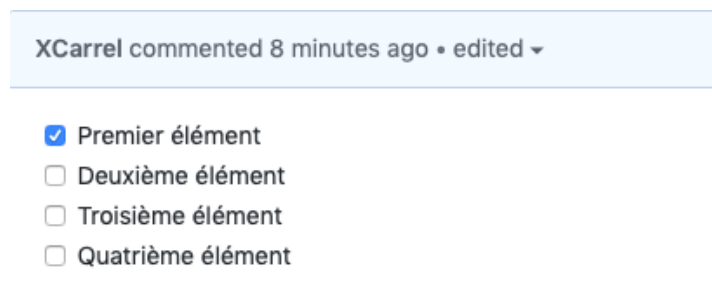
Il est parfois utile (nécessaire dans le cas du sprint review) d'avoir une checklist dans une issue. Cela peut se faire avec du markdown dans Github en débutant la ligne avec « - [] ». Exemple :



The screenshot shows a GitHub issue editor with a 'Write' tab selected. The text area contains a checklist in markdown format:

- [x] Premier élément
- [] Deuxième élément
- [] Troisième élément
- [] Quatrième élément

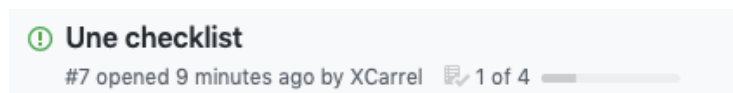
L'issue présentera la liste ainsi :



The screenshot shows a GitHub issue comment from 'XCarrel' with a rendered checklist:

- ☒ Premier élément
- ☐ Deuxième élément
- ☐ Troisième élément
- ☐ Quatrième élément

Et dans la liste des tâches, on aura une vue résumée :

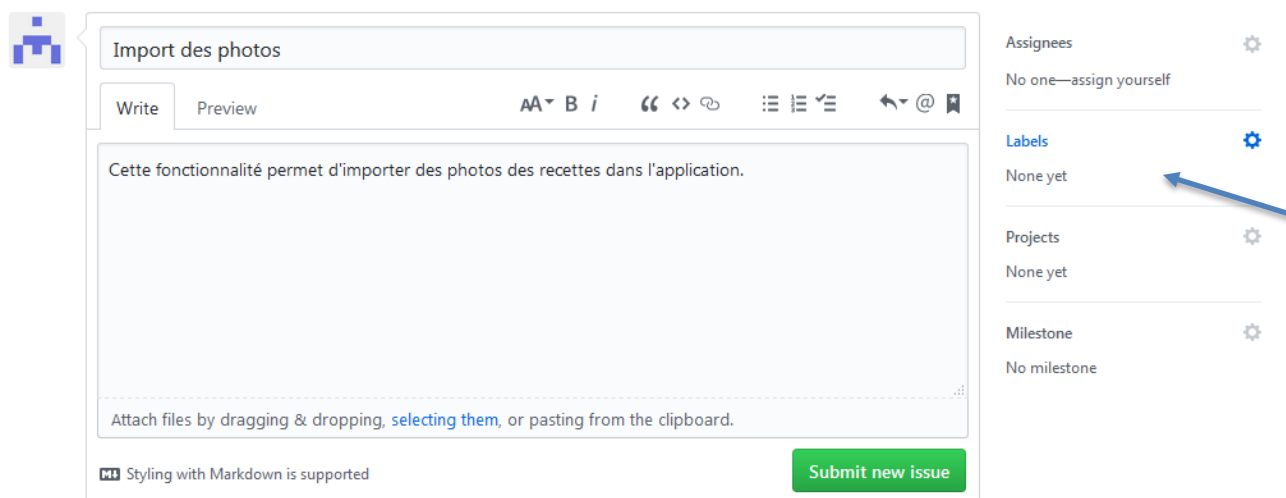


The screenshot shows a GitHub task list entry with a summary view of a checklist:

! Une checklist
#7 opened 9 minutes ago by XCarrel 1 of 4

2.7.3 Etiquettes

Les étiquettes (« Labels » en anglais) permettent de catégoriser les tâches de manière très flexible. Dans « Labels », indiquer le type de l'issue (amélioration, bug, nouvelle fonctionnalité)



Remarque : Il est possible de créer des labels personnalisés : cliquer sur le bouton « Labels »

2.8 Suivi

Tout au long du mandat :

- Faites de fréquents 'commits' dans Github :
 - Dès qu'une tâche est terminée et que sa carte passe dans la colonne 'Done' du Kanban
 - En fin de journée de travail. Si vous devez laisser votre travail 'en cours', nommez le commit 'WIP' (Work In Progress)
- Déplacez et fermez les issues dans les colonnes du Kanban du sprint en cours
- Sélectionnez dans 'ToDo' la ou les tâches suivante(s) à faire passer dans 'In Progress'
- Consignez les événements majeurs dans le journal de bord.
- Remplissez votre journal de travail au quotidien

3 Tests

Nous fournissons deux éléments concernant les tests :

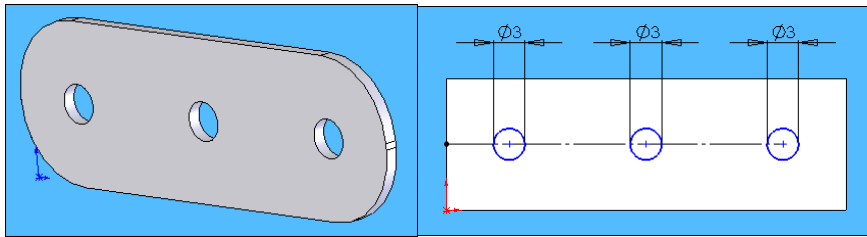
1. La Stratégie,
2. Les Résultats.

Ces deux points sont détaillés plus loin, mais commençons par quelques définitions que nous utiliserons dans la description de la stratégie de test.

3.1 Niveau

Nous appliquons jusqu'à trois niveaux de test :

- **Le Test unitaire** : Il vérifie le bon fonctionnement d'un composant réalisé par une personne.



Exemple : on vérifie qu'une pièce de Mécano respecte les dimensions voulues

- **Le Test d'intégration** : Il vérifie que deux ou plusieurs composants testés unitairement fonctionnent correctement ensemble.



Exemple : on vérifie que diverses pièces de Mécano peuvent bien être assemblées au moyen des vis et écrous.

- **Le Test Système** : Il vérifie le bon fonctionnement du système complet dans l'environnement où il va être vraiment utilisé.



Exemple : on vérifie le bon fonctionnement de notre construction complète.

3.2 Type

Nous distinguons également trois types de tests :

- Le Test fonctionnel, qui vérifie que l'élément testé fait ce qu'on attend de lui ;
- Le Test de performance, qui vérifie que l'élément testé peut traiter la quantité d'information voulue dans le temps voulu ;
- Le Test de robustesse, qui vérifie que l'élément testé peut reprendre un fonctionnement normal après avoir passé par une situation anormale.



Exemple : considérons la machine ci-dessus.

Cette machine sert à imprimer des flyers, les plier, les insérer dans une enveloppe, fermer l'enveloppe et faire des paquets de 100 enveloppes fermées.

Un test fonctionnel vérifiera que les flyers sont bien pliés en trois volets égaux et qu'il y a bien 100 enveloppes par paquet au final

Un test de performance vérifiera que la machine est capable de produire 250 paquets de 100 enveloppes en une heure

Un test de robustesse vérifiera que si on fournit des enveloppes qui sont déjà remplies, le système les écarte et reprend son fonctionnement dès que les enveloppes fournies sont vides

3.3 Stratégie

Pour tout mandat/projet, nous définissons une stratégie de test. Celle-ci consiste à décrire :

1. Le matériel et logiciel tiers.
2. Les données de test.
3. Les personnes qui vont participer aux tests : camarades de classe, amis, famille, profs, ...
4. Le timing des activités de test
5. Les types et niveaux de tests effectués

Exemple (site web pour un yearbook):

Pour le développement du site web, le serveur sera un wamp installé sur le même PC que celui sur lequel j'écrirai le code. Sur ce poste, je ne ferai que des tests fonctionnels.

Je préparerai

- Un script SQL qui créera 60 élèves répartis dans 4 classes.
- Un ZIP contenant des photos fictives pour les 60 élèves

J'utiliserai ensuite une machine virtuelle VMWare avec Debian 9 installé dessus et un serveur LAMP. Cette machine virtuelle sera également sur mon PC de développement. Elle ne servira qu'à faire des tests fonctionnels. Elle sera visible sur le réseau de l'école. Mon chef de projet ainsi que deux de mes camarades de classe l'utiliseront pour faire quelques tests.

Je déploierai mon site du l'hébergeur 'Swisscenter' pour faire les tests fonctionnels finaux. Avec l'aide d'autres camarades, nous effectuerons également des tests de robustesse et de performance en se connectant à plusieurs simultanément.

Les tests fonctionnels seront faits avec les navigateurs Google Chrome et Firefox sur Windows, Google Chrome, Firefox et Safari sur Mac. Aucun test n'est prévu sur Edge.

3.4 Résultats

Les résultats des tests s'inscrivent au fur et à mesure du projet dans un tableau récapitulatif dont :

- Chaque colonne représente les résultats d'un moment d'activité de test. Elle donne le résultat du test d'un ou plusieurs scénarios. L'entête de la colonne contient :
 - La date
 - La personne qui a fait le test
 - L'environnement
 - Eventuellement : les données de test
- Chaque ligne représente chacune des fois où un scénario a été testé. L'entête de ligne contient l'identifiant et le titre du scénario
- Une cellule du tableau représente le résultat du test :
 - OK sur fond vert si le test est réussi
 - KO sur fond rouge si le test est raté, avec une information relative au problème
 - Rien si ce scénario n'y pas été testé ce jour-là

L'exemple qui suit est tiré d'un rapport de TPI dont le sujet était le développement d'un jeu pédagogique (apprentissage de vocabulaire) sur Android :

Scénario	17.5 Développeur Galaxy 3 Voc 1	24.5 Développeur Galaxy 3 Voc 1+3	1.6 Chef Proj Samsung S7 Voc1+3	7.6 Développeur Galaxy 3 Voc 1+3
1.1 Lancement d'une partie	OK	OK		OK
1.2 Quitter le jeu	KO	KO		KO
2.1 Le joueur choisit le français...	OK	OK		OK
2.2 Changement de langue pour le prof		KO Choix ignoré		KO Choix ignoré
2.3 Changement de langue pour l'élève		KO Choix ignoré		KO Choix ignoré
2.4 Pas de vocabulaire pour la langue sélectionnée	OK	OK		OK
2.5 Le joueur change de vocabulaire	OK	OK		OK
3.1.1 Le prof se déplace de gauche à droite		OK	OK	OK
3.1.2 le prof choisit un mot		OK	OK	OK
3.1.3 Le prof lance un avion en papier		OK	OK	OK
3.1.4 Le prof a envoyé tous les élèves en pause			KO Ça continue	KO Ça continue
3.1.5 L'élève atteint le prof			OK	OK
3.2.1 L'élève reçoit un mot correct			OK	OK
3.2.2 l'élève reçoit un mot incorrect			OK	

4 Journal de Travail et Journal de Bord

Tout au long du projet, le mandataire maintient à jour deux journaux distincts.

4.1 Le Journal de Travail

Il a pour but de savoir qui a passé du temps (et combien) sur quelle tâche. Il sert typiquement à :

- Introduire le nombre d'heures effectives dans le fichier MS-Project durant la rétro-planification ;
- Facturer le travail (s'il y a lieu).

Exemple :

Personne	Date	Tâche	Durée(h)	Commentaire
Julien	7.9.16	Mise en place de l'environnement de travail	3h	Réinstallation complète Windows 7
Julien	7.9.16	Rédaction Use Cases	1.5	25 minutes d'installation de logiciel (Visio)
Julien	7.9.16	Création maquettes	2.5	
Julien	8.9.16	Rédaction Use Cases	2	
Julien	8.9.16	Séance de revue des UC	1	
Julien	9.9.16	Ajustements Use Cases	0.5	
Julien	9.9.26	Séance de revue des UC	0.5	
Julien	9.9.16	Préparation planning	1	
Julien	9.9.16	Codage UC 1	4	30 minutes perdues à cause d'un problème de compatibilité de librairies

Le journal de travail est typiquement tenu dans un fichier Excel se trouvant dans le dossier doc du repository. Vous en trouverez un canevas de base sous

<https://github.com/fandolfatto/GestionRecettes/tree/master/doc/Projet>

4.2 Le Journal de Bord

Il a pour but de retracer l'histoire du projet au travers de ses faits marquants

Exemple :

Date	Evénement
8.9.16	Revue des Use cases avec M. Gates : des ajustements sont demandés au niveau de la gestion des membres
9.9.16	Revue des Use cases avec M. Gates : ils sont validés
9.9.16	Rétro-planification de planning : aucun problème en vue
14.9.16	M. Gates demande (par mail) l'ajout d'une page d'historique des achats

Le journal de bord se trouve dans le document de projet

5 Documentation de Projet

Il est primordial de fournir avec le travail rendu toutes les informations nécessaires pour qu'une autre personne puisse reprendre et continuer le projet.

La documentation d'un projet réalisé dans la filière informatique du CPNV consiste au final en un document (livré en format PDF) contenant :

1. Une introduction avec :
 - Une description générale du projet
 - Les informations sur le mandant du projet
 - Les informations sur le mandataire (nous)
 - Des références (pas de copie !) aux documents fournis en guise d'énoncé (cahier des charges, projet précédent, emails explicatifs, ...)
2. L'analyse, telle que décrite au chapitre 1 de ce document
3. La planification initiale, telle que décrite au chapitre 2.2 de ce document
4. Les détails de réalisation qui méritent explications
5. Les détails de la livraison, telle que décrite au chapitre 7 de ce document
6. Le Journal de Bord, tel que décrit au chapitre 4.2 de ce document
7. Une conclusion, contenant entre autres :
 - Un commentaire critique de comparaison entre ce qui était demandé et ce qui a été réalisé
 - Le « planning réel », ainsi qu'une comparaison de celui-ci avec le planning initial
 - La liste des problèmes connus
 - Un commentaire personnel sur l'ensemble du projet

6 Communiquer sur l'avancement des travaux

Pour tenir le mandant informé de l'avancement des travaux, nous lui communiquons à intervalles réguliers :

- Le Journal de Bord (inclus dans le document de projet),
- Le Journal de Travail,
- Le Document de Projet.
- Le contenu actuel du projet

Sauf cas exceptionnel demandé spécifiquement par le mandant, tous les documents sont toujours transmis en format PDF.

Dans les journaux, on ne fait que rajouter des lignes. Il est donc facile pour le mandant de détecter les changements intervenus entre deux envois : il suffit d'aller consulter le bas du tableau, qui est organisé par ordre chronologique.

Il n'en va pas de même pour le Document de Projet, dans lequel des modifications difficilement décelables peuvent se situer dans des endroits que le mandant a déjà lu plusieurs jours auparavant.

Pour guider le lecteur, nous fournissons le document de projet en deux exemplaires :

1. Le document lui-même
2. Un exemplaire qui met en évidence les modifications intervenues depuis la version précédente.

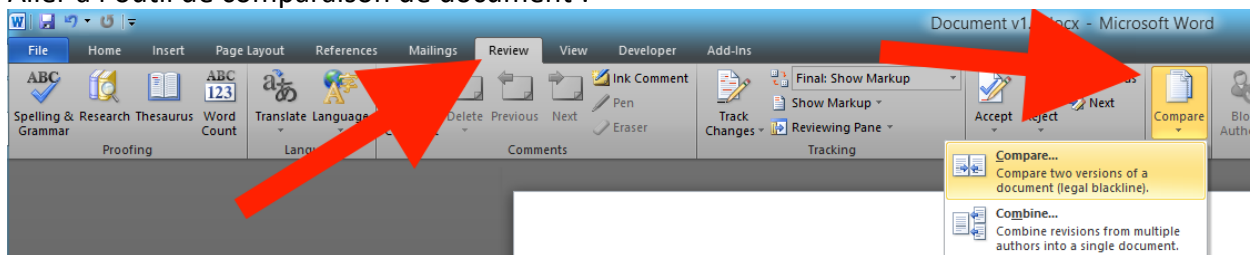
Exemple : si l'on travaille sur un document « Projet Alpha.docx » et que l'on publie la version 1.3, on fournira :

1. « Projet Alpha v1.3.pdf »
2. « Projet Alpha v1.3 – différences.pdf »

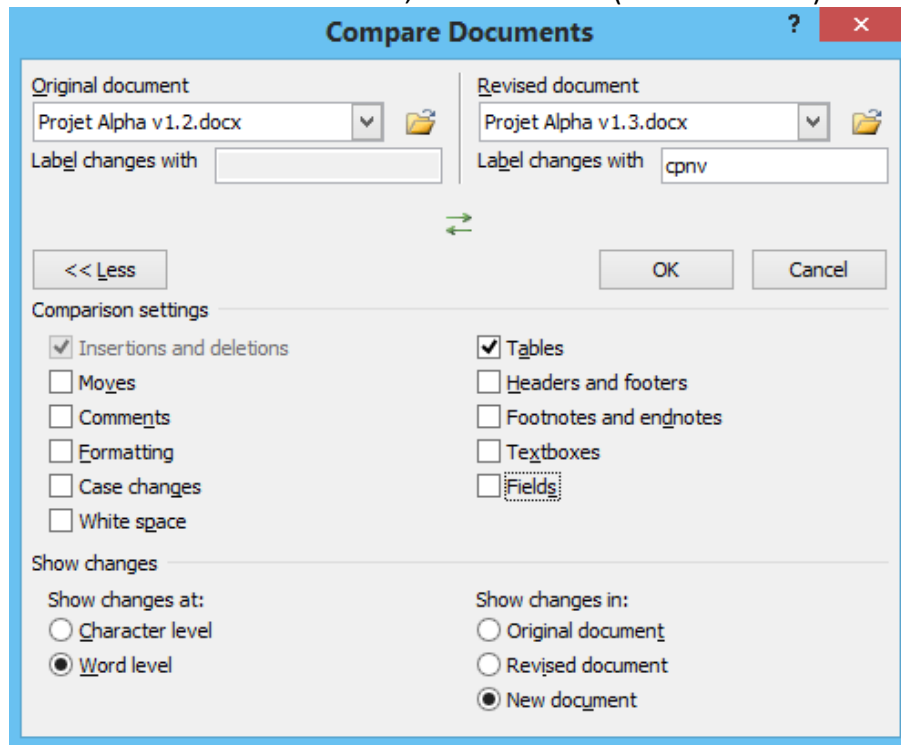
Remarque : ces fichiers n'ont pas besoin d'être enregistrés dans le repository Git, ce dernier contenant les originaux.

Pour créer la version « ... - différences » :

1. Aller à l'outil de comparaison de document :



2. Sélectionner les deux versions, tout décocher (sauf « tables ») dans les options :



3. Effectuer la comparaison (OK) et sauver le résultat au format pdf.

En ce qui concerne le contenu, nous informons le mandant du fait que nous avons mis le repository à jour sur Github. Si le mandant n'a pas accès à Github, nous lui faisons parvenir le .zip

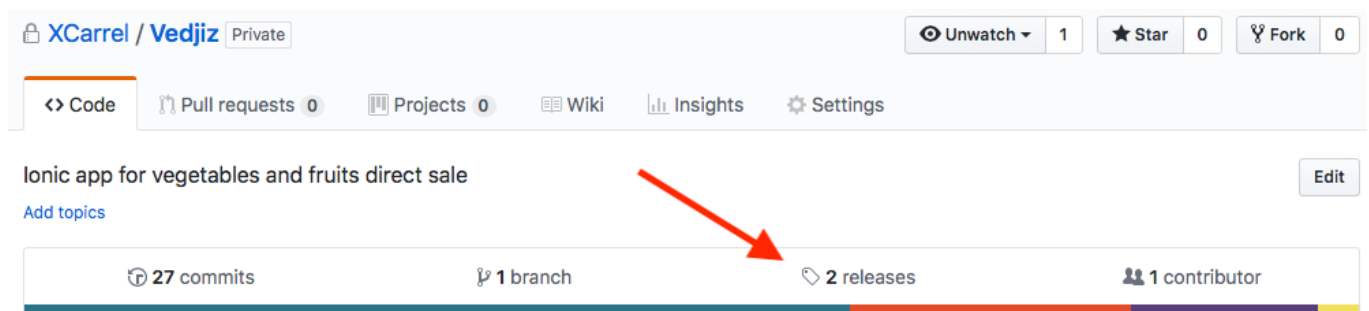
7 Livraison

Nous livrons le résultat de notre travail au moyen « release » dans Github.

Une release est un commit du repository que l'on distingue des autres par des informations supplémentaires :

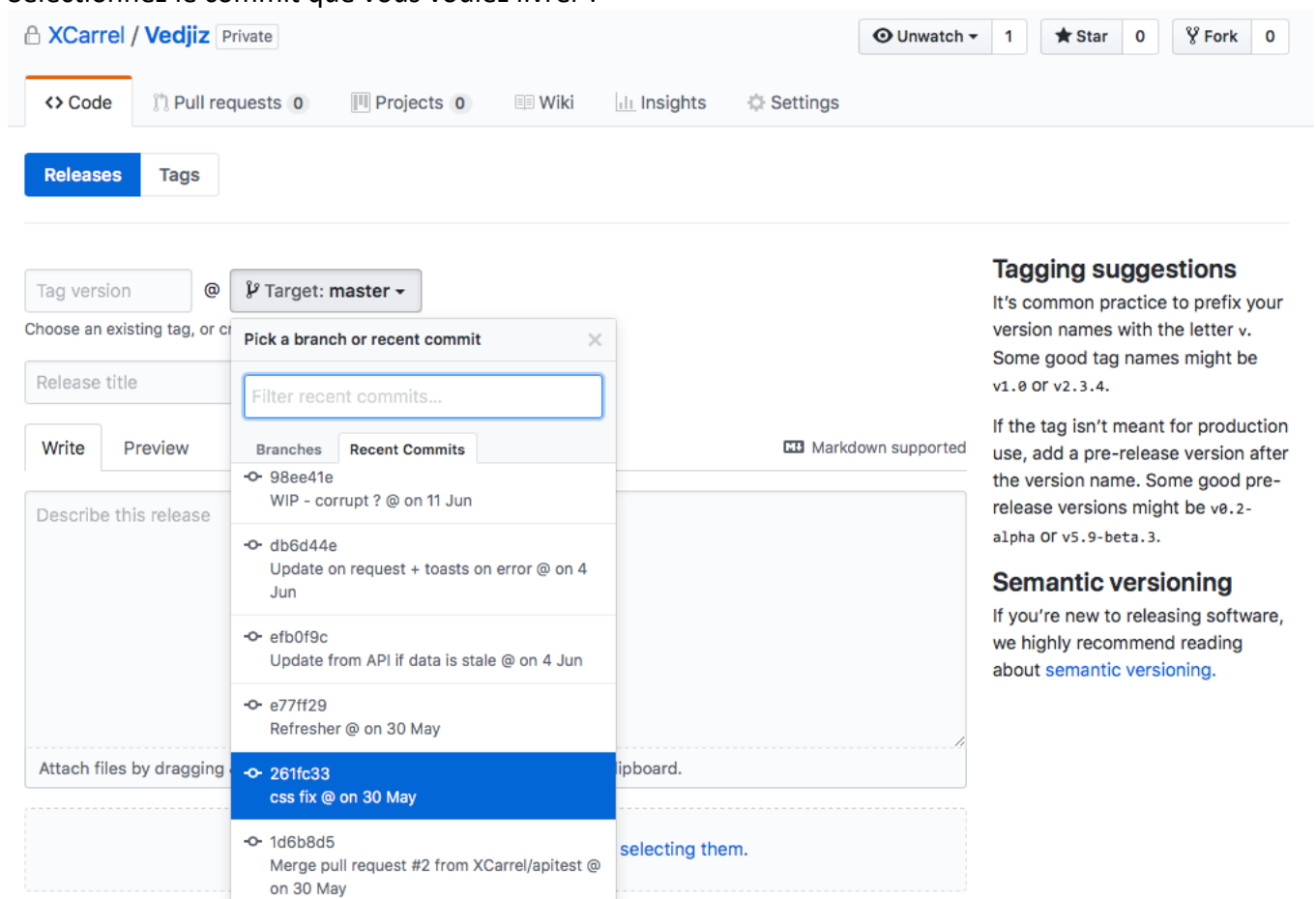
- Un numéro de version
- Un titre
- Une description

Pour créer une release, allez dans l'onglet « release » du repository



et cliquez sur « Draft a new release ».

Sélectionnez le commit que vous voulez livrer :



Remplissez le reste du formulaire et publiez la version :

XCarel / Vedjiz

Private

Unwatch

1

Star

0

Fork

0

<> Code

Pull requests 0

Projects 0

Wiki

Insights

Settings

Releases

Tags


Edit release

Delete


V0.6-Beta


261fc33

Version for external testers

 XCarel released this a minute ago

Assets

 [Source code \(zip\)](#)

 [Source code \(tar.gz\)](#)

Functions that are to be tested:

- Login/Logout
- List of products

Il ne vous reste plus qu'à communiquer au mandant le tag et le titre de la release.
Si le mandant n'a pas accès à Github, faites-lui parvenir le .zip

Annexe I. Planification selon une approche plus « classique »

Dans le cas où votre chef de projet ou votre expert refuserait le mode Agile simplifié proposé, vous avez la possibilité d'utiliser github selon une approche plus « classique » (modèle en V, modèle en cascade). Vous planifierez votre projet en suivant les étapes ci-dessous :



Remarque : Cela revient à ne faire qu'un seul sprint à la planification selon le mode Agile proposé en §0.

7.1.1 Planification initiale

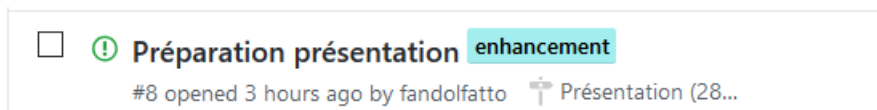
La planification initiale du projet est grossière car on ne dispose que de peu d'informations à ce stade du projet, comme cela était le cas aussi dans le mode Agile. Elle n'est typiquement constituée que de phases générales et de jalons majeurs tels que :

- Début/Fin
- Validation des use cases/scénarios avec le mandant
- Réalisation d'un use case choisi
- Version Beta
- ...

Les « issues » seront créées comme expliqué ci-dessus. Plusieurs projets pourront être créés et contiendront les différentes issues, comme ci-dessous par exemple (ne pas oublier dans la description de chaque projet de mettre les différentes tâches ainsi que l'échéance du projet) :

3 Open ✓ 0 Closed		Sort ▼
1. Analyse / conception Updated a minute ago	<ul style="list-style-type: none">• Analyse et conception du projet• Echéance : 02.06.2018	...
2. Réalisation / tests Updated 10 seconds ago	<ul style="list-style-type: none">• Création des maquettes graphiques• Implémentation de la partie "calculs" et "graphes" de la calculatrice et tests unitaires• Ajustements• Tests d'intégration, tests de masse• Echéance : 22.06.2018	...
3. Livraison / présentation Updated 9 minutes ago	<ul style="list-style-type: none">• Démo• Guide d'installation• Préparation de la présentation• Présentation• Echéance : 29.06.2018	...

Les tâches / « issues » suivies d'un jalon contiendront un milestone, comme par exemple « Préparation présentation » qui amène à la « Présentation », « Réalisation / tests » qui conduit à la « Version Beta » du projet.

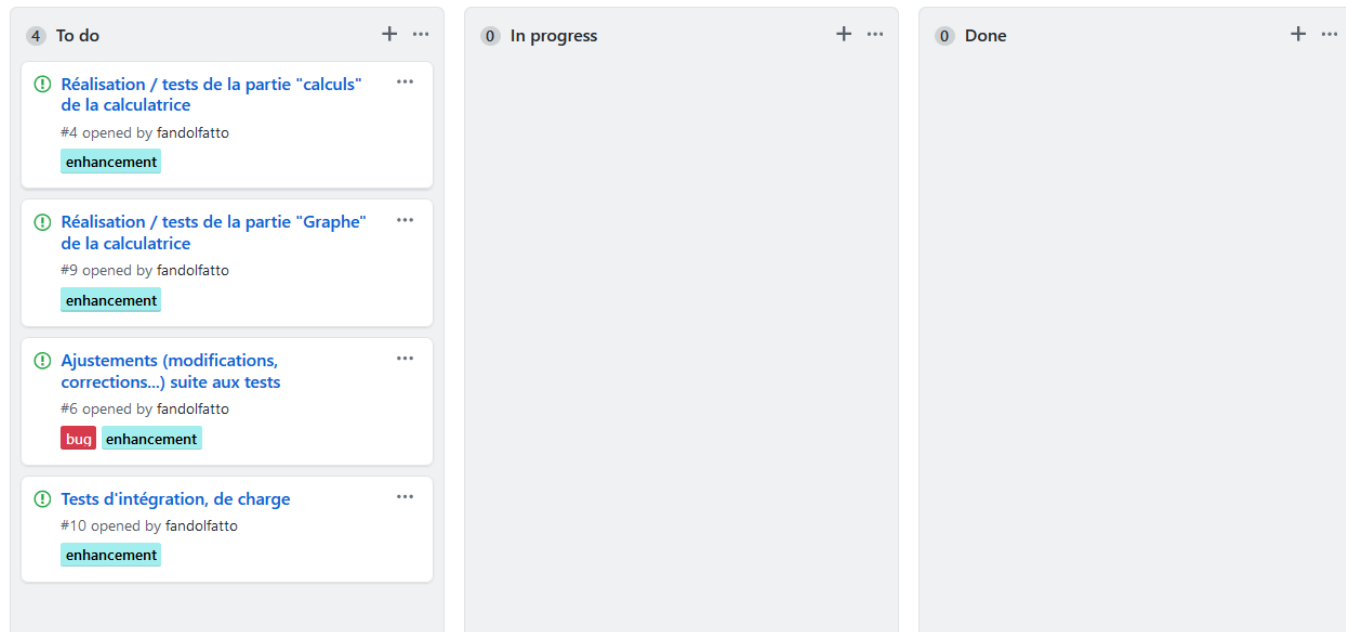


Chaque projet contient les « issues » qui le composent.

Exemple :

2. Réalisation / tests

Updated 8 minutes ago



7.1.2 Mise à jour

A partir de là, une rétro-planification sera réalisée au minimum 1 fois par semaine.

Une rétro-planification consiste à :

- Reporter dans le planning le travail effectué depuis la dernière rétro-planification ;
- Décomposer une tâche existante en deux ou plusieurs tâches SMART (voir ci-dessus) ;
- Décider des tâches qui seront exécutées dans les jours qui viennent ;
- Consigner les éventuels changements majeurs dans le journal de bord.

Typiquement, une première rétro-planification effectuée sur le planning de la section précédente pourrait mener à ajouter dans le projet « Analyse / conception » les issues :

- « Rédaction des use cases et scenarii »
- « Revue préliminaire »
- « Ajustement des use cases et scenarii ».