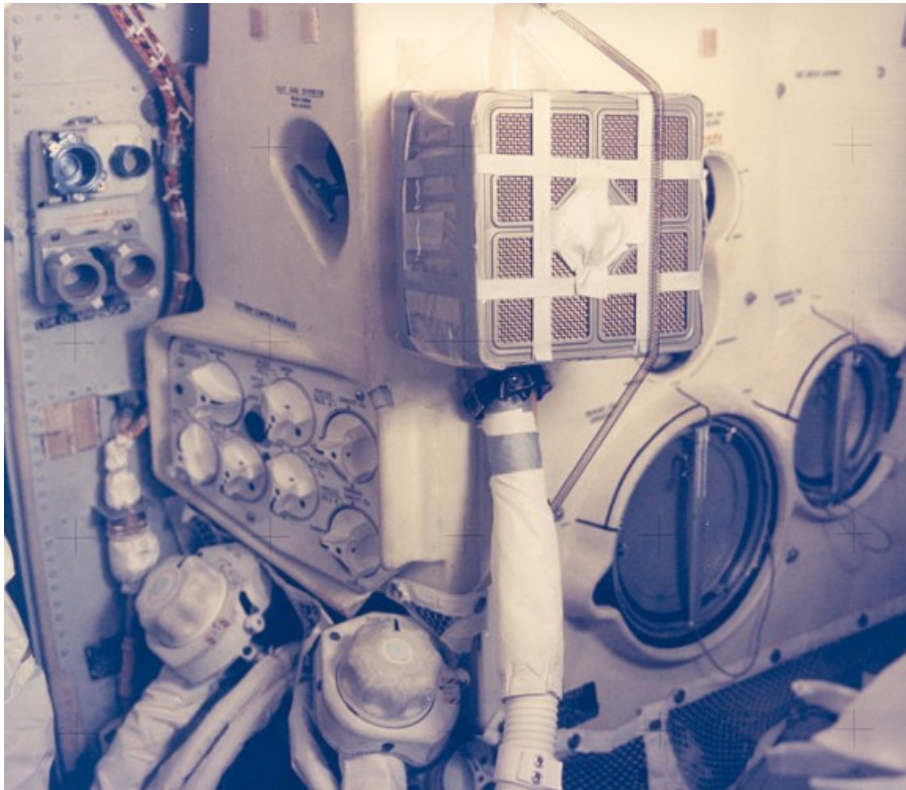


Exécution de Mandats

Table des matières

1	Introduction.....	3
2	Analyse du mandat	4
2.1	Use Cases (cas d'utilisation)	4
2.2	Scénarios	4
2.2.1	Identification	5
2.2.2	Détails	5
2.3	Modèle conceptuel de données (MCD)	7
3	Planification.....	8
3.1	Création des projets et fonctionnalités.....	8
3.1.1	Définition	8
3.1.2	Création du projet	8
3.1.3	Création des tâches	9
3.1.4	Création de sous-projets	10
3.1.5	Tri des projets	11
3.2	Planification initiale	12
3.2.1	Exemple de planification pour la réalisation d'une calculatrice scientifique :	13
3.3	Mise à jour.....	15
3.4	Vue d'ensemble finale	16
3.5	Principe général : résumé.....	16
4	Détails d'implémentation	18
5	Tests	19
5.1	Niveau de test.....	19
5.2	Type de test	20
5.3	Stratégie de test.....	21
5.4	Résultats de test	21
6	Journal de Travail et Journal de Bord	23
6.1	Le Journal de Travail	23
6.2	Le Journal de Bord	23
7	Documentation de Projet.....	24
8	Communiquer sur l'avancement des travaux	25
9	Livraison	27
Annexe I.	Planification selon une approche plus « classique »	29
9.1.1	Planification initiale	29
9.1.2	Mise à jour	30

1 Introduction



Cela vous dit quelque chose ? Non ? Un petit peu d'histoire alors...

Pour le programme Apollo, la NASA avait mis sur pied deux équipes : une pour concevoir le Module Lunaire et l'autre pour le Module de Commande. Lorsque les choses tournèrent mal pour la mission Apollo 13, les trois astronautes se retrouvèrent face à un sérieux problème : ils devaient réparer le module de commande endommagé avec des pièces du module lunaire. Chacun des deux modules disposait d'un système de filtrage/recyclage de l'air, mais pas de chance : les filtres conçus par une équipe avaient une entrée carrée et les bonbonnes choisies par l'autre avaient une sortie ronde ! Un « détail » qui a bien failli coûter la vie des trois membres d'équipage.

Une des leçons à tirer de cette aventure est qu'un minimum de communication et de convention est nécessaire au bon déroulement d'un projet.

Le but de ce document est de décrire les pratiques appliquées au sein de la filière informatique du CPNV, juste au cas où on nous demanderait de construire une fusée.

2 Analyse du mandat

« 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 / Scénarios
- le modèle conceptuel de données (MCD) si des données doivent être gérées.

2.1 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 !

2.2 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...

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

2.2.1 Identification

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

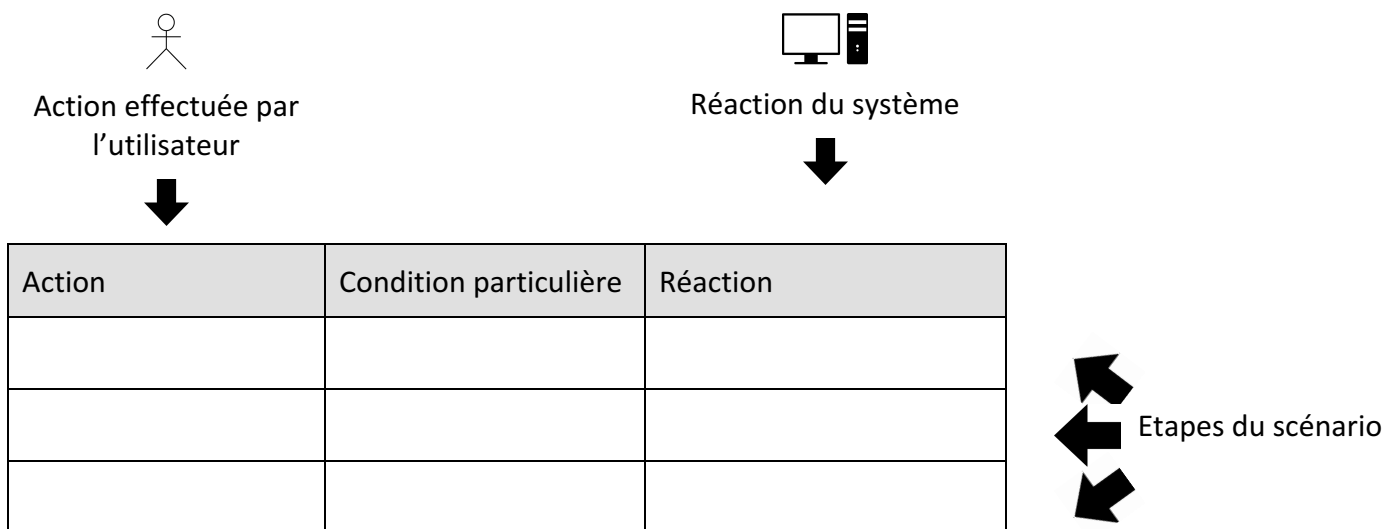
Identifiant + Titre		Un code unique à travers tout le projet + titre court
En tant que		Rôle
Je veux		Quelque chose
Pour		But
Priorité		Selon MoSCoW :

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

2.2.2 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.

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
Je touche le bouton « contact »		La liste des contacts personnels s'affiche
Je touche le contact « John »		Les trois numéros de John s'affichent (portable, maison et travail)
Je touche le numéro « maison »		Composition du numéro de domicile de John
		Ça sonne
	John ne décroche pas dans les 30 secondes	L'appel se termine, on retourne aux trois numéros de John
Je touche le numéro « portable »		Composition du numéro de portable de John
		Ça sonne et John décroche
Je discute de mes vacances avec John		

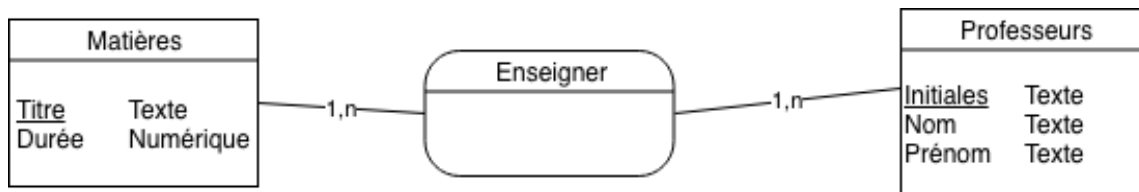
Action	Condition	Réaction
Je touche le bouton « raccrocher »		La communication s'interrompt L'application affiche un résumé de l'appel: Personne appelée, date, heure, durée et coût de l'appel
Je touche « OK »		L'application se referme

2.3 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 ait des données à gérer ne signifie pas nécessairement qu'il y a une base de données !

Bref rappel sur Le MCD¹ :

- 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 :

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.

¹ Voir support de cours du module ICT-104 pour plus de détails

3 Planification

3.1 Création des projets et fonctionnalités

Nous gérons la planification des projets avec Github selon le mode Agile. Au préalable, vous devrez avoir créé un compte dans github, avec votre adresse email du CPNV et un nom d'utilisateur qui permette de vous reconnaître (« BillGates » 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 ».

3.1.1 Définition

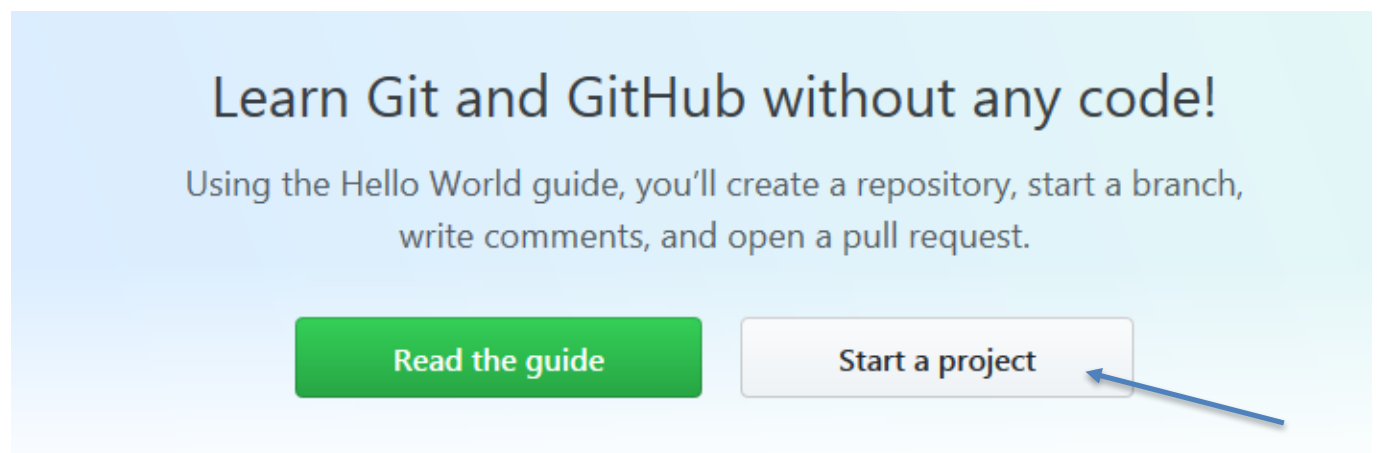
Sprint (ou itération) : Intervalle de temps court (1 mois maximum, souvent appelé itération), pendant lequel la Dev Team va concevoir, réaliser et tester de nouvelles fonctionnalités. Suivant la nature du projet, le nombre de use case peut être assez grand, ce qui rend difficile le choix des fonctionnalités 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 §3.2.1.3).

3.1.2 Création du projet

Sur le site www.github.com, créer un nouveau projet.



Y créer un nouveau repository :

Indiquer un nom (court et parlant, représentatif du contenu du projet) et une description.

Bien que GitHub soit destiné à la gestion de code uniquement, nous utiliserons également nos repositories pour contenir les documents de projet (documentation, journal de travail, ...). Vous veillerez donc à ce qu'il y ait deux répertoires à la base de votre repository :


1. « Files » qui contiendra le résultat de votre travail (code source, scripts, données, ...)
2. « Documentation » qui contiendra les fichiers de projet

Garder les options telles qu'indiquées dans la copie d'écran ci-dessous. Cliquer sur « Create repository ».

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

 fandolfatto ▾

Repository name

/ GestionRecettes ✓

Great repository names are short and memorable. Need inspiration? How about **cautious-meme**.

Description (optional)

Ce projet gère les recettes de l'école professionnelle de cuisine



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.



Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None ▾

Add a license: None ▾



Create repository

3.1.3 Création des tâches

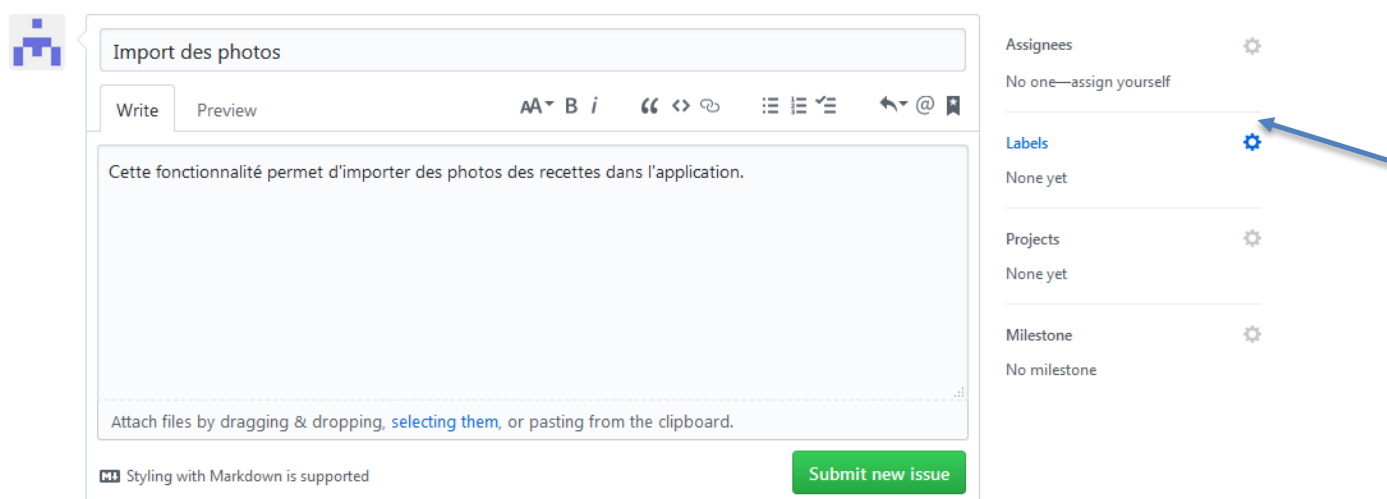
Aller sur l'onglet « Issues ». Les « issues » créées pour le projet regroupent l'ensemble des tâches à accomplir dans le projet.



Dans l'onglet « Issues », pour créer une nouvelle « issue » / tâche, cliquer sur le bouton « New Issue ».

Indiquer un titre et une description. 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... Une fois les informations indiquées, cliquer sur « Submit new issue ».

Dans « Labels », indiquer le type de l'issue (amélioration, bug, nouvelle fonctionnalité...), indiquer le milestone (date à laquelle le cas doit être traité). Le milestone **doit** être précisé lorsqu'il s'agit de la revue / démo d'un sprint, il est facultatif autrement.



Remarque : Pour créer des labels et des milestones personnalisés, aller dans l'onglet « Issues ». Cliquer sur le bouton « Labels » / « Milestones ».

Dans « Milestones », cliquer sur « New milestone ». Entrer un titre parlant suivi d'une date d'échéance, par exemple « démo sprint1 (10.07.2018) », indiquer aussi la date d'échéance dans le champ prévu à cet effet et une description. Il est conseillé de mettre la date d'échéance dans le titre de la milestone pour qu'elle soit visible rapidement au niveau de l'issue.

3.1.4 Création de sous-projets

Dans l'onglet « Projects », cliquer sur le bouton « Create a project ». Indiquer un nom de projet, par exemple « sprint 1 ». Indiquer **obligatoirement** une description pour décrire le projet. Dans la description, y indiquer la date d'échéance (exemple : Echéance : 12.09.2018). Choisir le template « Basic kanban ». Cliquer sur « Create Project ».

Important : Comme nous travaillons en mode « Agile », nous nommerons nos projets de la manière suivante :

« sprint » suivi du n° du sprint

Create a new project

Coordinate, track, and update your work in one place, so projects stay transparent and on schedule.

Project board name

Sprint 1

Description (optional)

- Analyse globale et conception globale du projet
- Modélisation de la base de données de l'application
- Echéance : 08.09.2018

Project template

Save yourself time with a pre-configured project board template.

Template: **Basic kanban** ▼

Create project

Un nouveau projet vide apparaît.

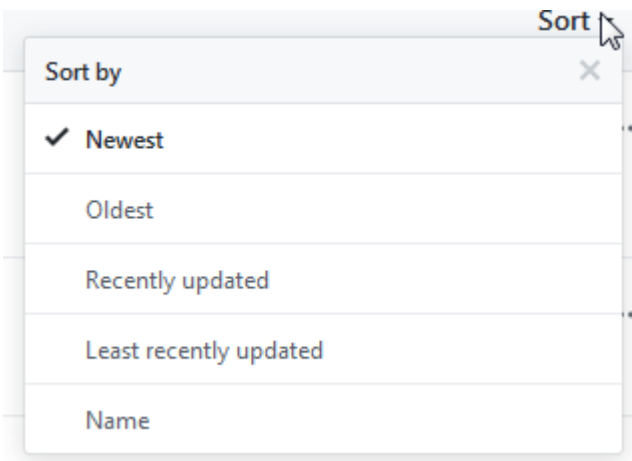
Les issues / cartes définies pour le projet apparaissent à droite. Les drag and dropper lorsque vous souhaitez les traiter dans le projet nouvellement créé.

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 ».

3.1.5 Tri des projets

Dans l'onglet « Projects », vous pouvez voir l'ensemble des projets et les trier de différentes manières.

4 Open ✓ 0 Closed		Sort ▾
Planning général Updated 21 days ago	<ul style="list-style-type: none"> Sprint 1 : Analyse - conception globale du projet Sprint 2 : Import des recettes et photos Sprint 3 : Affichage des recettes et photos 	...
Sprint 1 Updated just now	<ul style="list-style-type: none"> Analyse et conception globale du projet Modélisation de la base de données de l'application Use cases Echéance : 08.09.2018 	...
Sprint 2 Updated 27 seconds ago	<ul style="list-style-type: none"> Import des recettes et des photos de recettes Echéance : 18.09.2018 	...
Sprint 3 Updated on 25 Jun	<ul style="list-style-type: none"> Affichage des recettes et des photos Echéance : 28.09.2018 	...



3.2 Planification initiale

Au début de votre projet, votre planification contient des tâches assez générales car on ne dispose que de peu d'informations à ce stade du projet. Elle n'est typiquement constituée que de phases générales et de jalons majeurs tels que :

- Début/Fin
- Création des Use cases/scénarios avec le mandant
- Validation des Use cases / scénarios avec le mandant
- Réalisation d'un Use case choisi
- Livraison finale
- ...

Le sprint 1 est défini de manière précise. Le contenu des sprints suivants pourra évoluer en fonction de l'avancement et du résultat du sprint 1. Les « issues » seront toutes définies et pourront également évoluer en fonction des tests utilisateurs et de la demande du client.

Si vous devez faire une modélisation de base de données, celle-ci pourra évoluer au fur et à mesure des sprints.

Chaque sprint aura une durée de 1 à 3 semaines maximum et pourra varier d'un sprint à l'autre.

3.2.1 Exemple de planification pour la réalisation d'une calculatrice scientifique :

Grâce à cette calculatrice, l'utilisateur pourra effectuer des calculs (opérations de base, calculs trigonométriques, utilisation de constantes comme pi...) mais aussi indiquer une fonction et en demander le graphe.

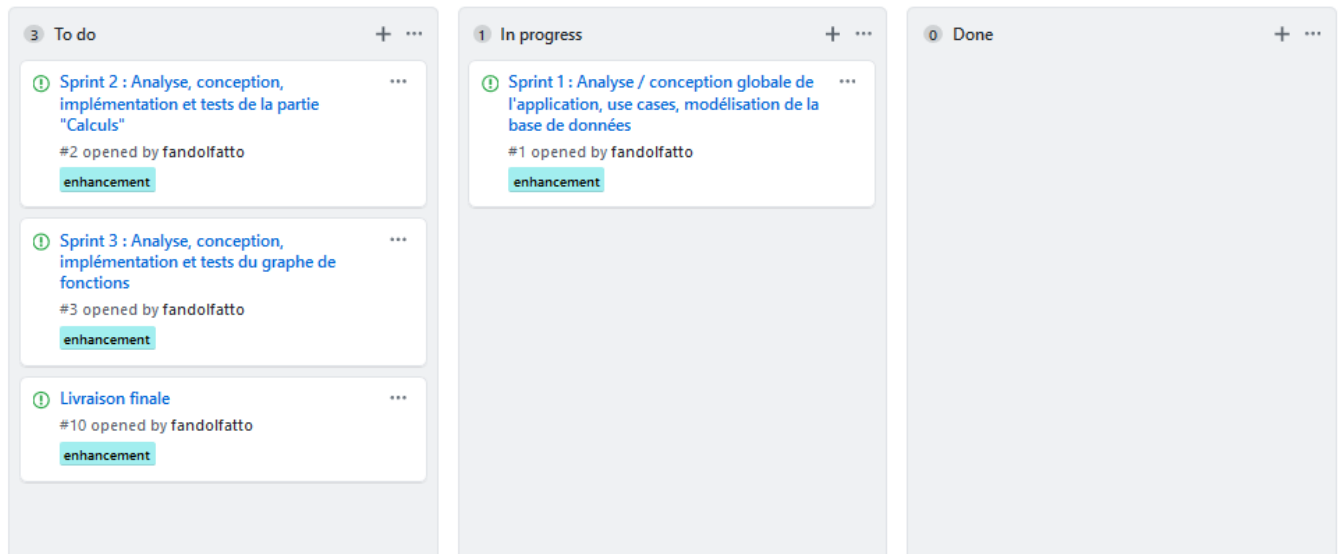
3.2.1.1 Vue d'ensemble du projet

Au début du projet, seul le sprint 1 sera défini de manière précise. Les sprints suivants seront définis au fur et à mesure que le projet avance. Un planning général sera établi, de manière assez grossière.

<div>Planning général</div> <div> Updated 5 hours ago</div> <div><div></div></div>	<ul style="list-style-type: none">• Sprint 1 : Analyse - conception globale du projet + modélisation de la base de données• Sprint 2 : Conception et implémentation de la partie "opérations"• Sprint 3 : Conception et implémentation de la partie "graphe de fonctions"• Echéance : 10.07.2018	...
<div>Sprint 1</div> <div> Updated 40 seconds ago</div> <div><div></div></div>	<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• Echéance : 20.06.2018	...
<div>Sprint 2</div> <div> Updated 21 seconds ago</div> <div><div></div></div>	<div>Réalisation de la partie "Calculs" :</div> <ul style="list-style-type: none">• Création des scenarii• Mise à jour du modèle de données• Implémentation et tests• Echéance : 29.06.2018	...
<div>Sprint 3</div> <div> Updated just now</div> <div><div></div></div>	<div>Réalisation de la partie "graphe de fonctions"</div> <ul style="list-style-type: none">• Création des scenarii• Mise à jour du modèle de données• Implémentation et tests• Echéance : 10.07.2018	...

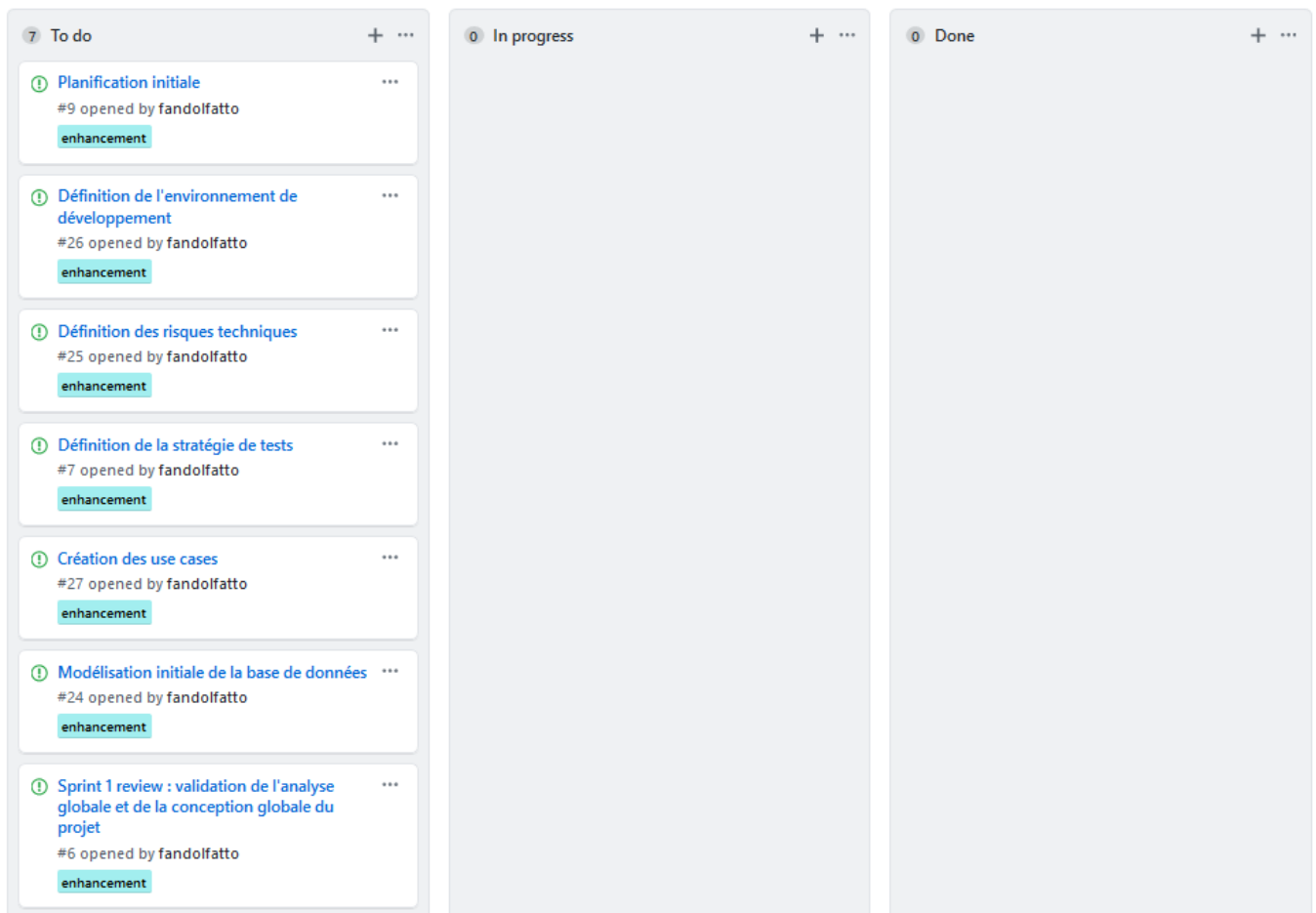
3.2.1.2 Planning général

Au clic sur « Planning général », la fenêtre suivante apparaît :



3.2.1.3 Sprint 1

Au clic sur « Sprint 1 », la fenêtre suivante apparaît :



Les tâches à traiter dans le sprint 2 seront définies quelques jours avant la fin du sprint 1.

3.2.1.4 Liste des tâches

Au début du projet, dans « issues » se trouvera l'ensemble des tâches générales pour l'application. La liste des tâches sera affinée et complétée au fur et à mesure du projet.

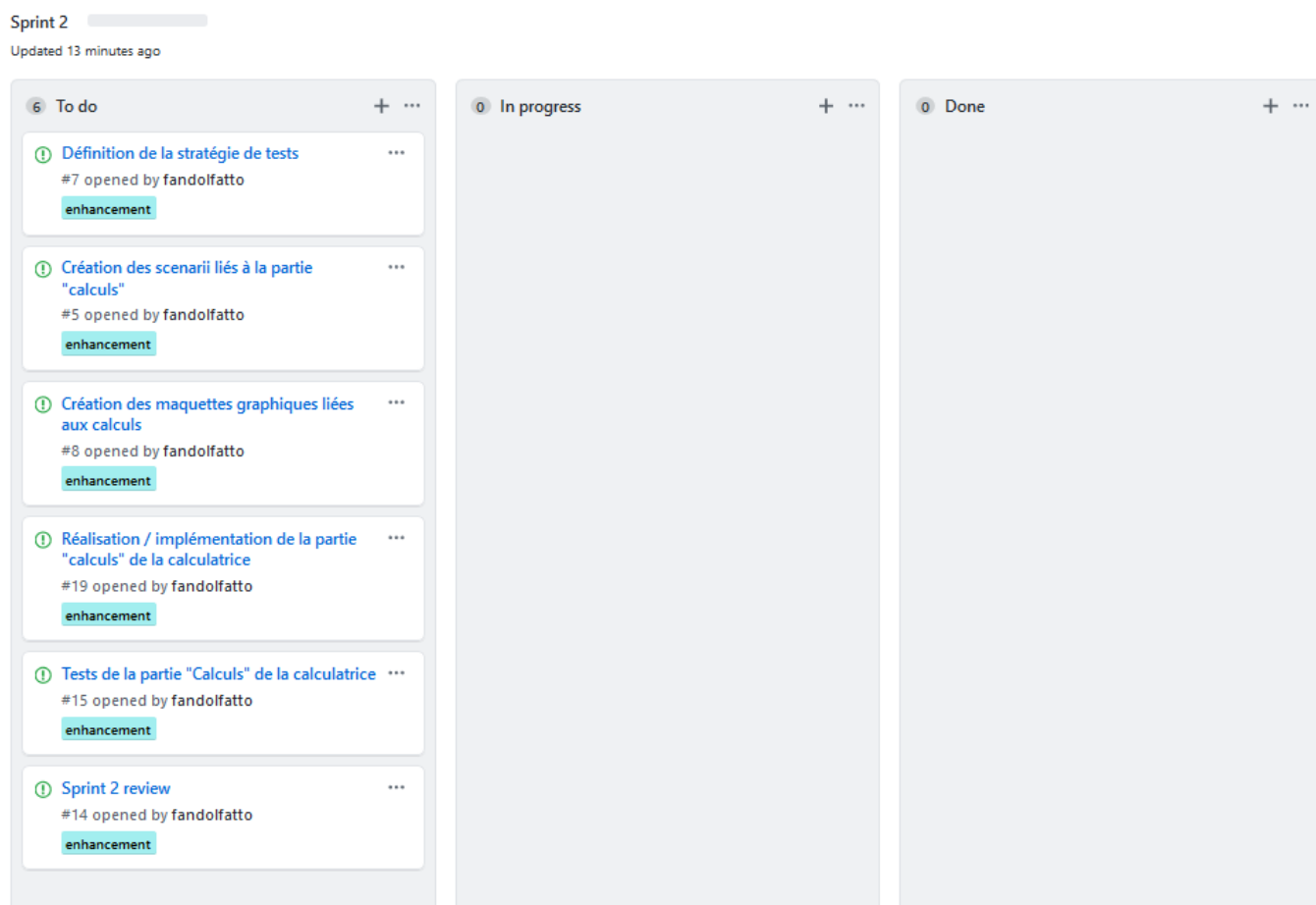
3.3 Mise à jour

Une semaine avant le début du prochain sprint, son contenu est défini.

Cela consiste à :

- Décider des tâches qui feront partie du sprint
- Décomposer une tâche existante en deux ou plusieurs tâches SMART (voir section suivante).
- Reporter des tâches du sprint en court qui n'auraient pas été réalisées
- Ajouter des bugs découverts dans le sprint en court
- Consigner les éventuels changements majeurs dans le journal de bord.

Par exemple, le sprint 2 sera défini quelques jours avant la fin du sprint 1 et contiendra les tâches suivantes :



La stratégie de tests prévue lors du sprint 1 n'étant pas satisfaisante, elle est reportée au sprint 2, d'autres fonctionnalités y sont ajoutées.

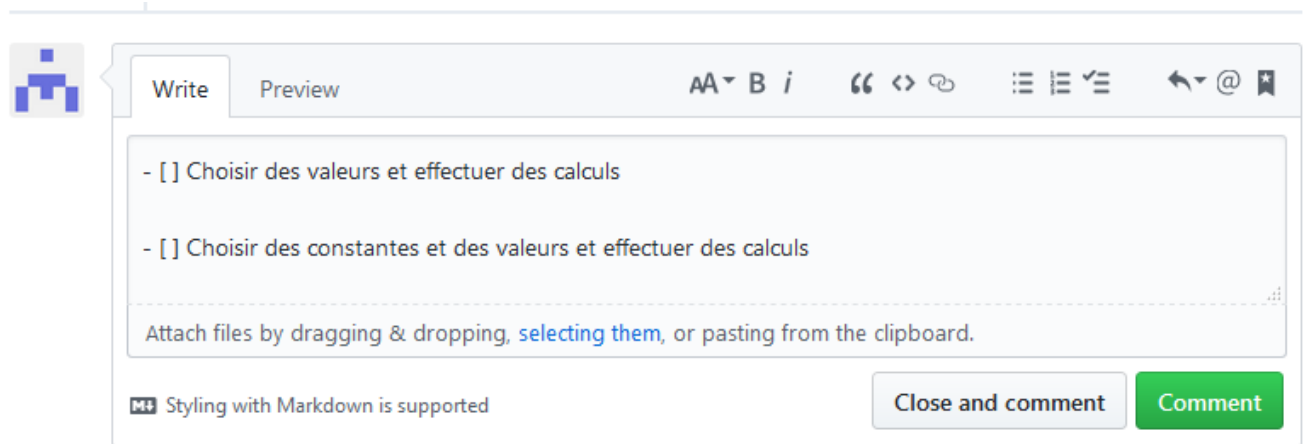
3.4 Vue d'ensemble finale

Planning général 🕒 Updated 20 days ago <div><div></div></div>	<ul style="list-style-type: none">• Sprint 1 : Analyse - conception globale du projet + modélisation de la base de données• Sprint 2 : Conception et implémentation de la partie "opérations"• Sprint 3 : Conception et implémentation de la partie "graphe de fonctions"• Echéance : 10.07.2018	...
Sprint 1 🕒 Updated 7 minutes ago <div><div></div></div>	<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• Echéance : 20.06.2018	...
Sprint 2 🕒 Updated 2 minutes ago <div><div></div></div>	<ul style="list-style-type: none">• Création des scenarii 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• Echéance : 29.06.2018	...
Sprint 3 🕒 Updated a minute ago <div><div></div></div>	<ul style="list-style-type: none">• Création des scenarii 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• Echéance : 10.07.2018	...

3.5 Principe général : résumé

1. Créer l'ensemble des tâches / « issues ». A ce niveau, les tâches restent assez générales, elles s'affineront au fur et à mesure de l'avancement de votre projet.
2. Créer un premier projet contenant le planning général i.e. l'ensemble des sprints et leur contenu global.

3. Créer ensuite un projet par sprint. Dans la description, y indiquer la date d'échéance. Y ajouter les tâches que vous vous engagez à réaliser à la fin du sprint. Commencer par créer le sprint 1 puis quelques jours avant la fin du sprint en cours, créer le sprint suivant avec les tâches à réaliser.
4. Chaque sprint contient une démo ou « sprint review ». Y préciser un milestone. Dans le commentaire, y ajouter une « task list » qui contiendra la liste des scénarios présents dans votre documentation, votre chef de projet coche les éléments de la checklist au moment de la démo et lorsque le test est réussi.



The screenshot shows a Jira comment editor interface. On the left is a small icon of a person. The editor has two tabs: 'Write' (active) and 'Preview'. The 'Write' tab contains a rich text editor with a toolbar at the top (font size, bold, italic, quote, link, unlink, list, link, unlink, undo, redo, mention, bookmark). The text area contains a task list:

- [] Choisir des valeurs et effectuer des calculs
- [] Choisir des constantes et des valeurs et effectuer des calculs

Below the text area is a dashed line and the text: 'Attach files by dragging & dropping, selecting them, or pasting from the clipboard.' At the bottom left, it says 'Styling with Markdown is supported'. At the bottom right, there are two buttons: 'Close and comment' and 'Comment' (green).

Les tâches détaillées suivent les recommandations suivantes :

1. Elles sont SMART ²:
 - **S**pécifique : l'énoncé est clair et précis. Quand on le lit, on sait ce que l'on doit faire. La tâche ne définit qu'une seule chose à faire ;
 - **M**esurable : il est possible de vérifier de manière non ambiguë que la tâche est réalisée ou non ;
 - **A**mbitieuse : la tâche représente un travail significatif ;
 - **R**éaliste : la personne en charge de la tâche a les moyens et les compétences pour l'effectuer ;
 - **T**emporelle : on a pu en estimer la durée, en heures.
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.

² Le principe SMART est très répandu, mais pas standardisé. La définition donnée ici est propre au CPNV.

4 Détails d'implémentation

Il est rare qu'un projet débouche du premier coup sur un produit fini. Il est donc primordial de fournir avec le travail rendu toutes les informations nécessaires pour qu'une autre personne puisse reprendre et continuer le projet.

Cela consiste au minimum à décrire :

- Les choix technologiques effectués, s'ils ne sont pas évidents ou imposés ;
- Le fonctionnement général du système, ainsi que les algorithmes spécifiques s'il y en a ;
- L'arborescence du répertoire « Résultat » de la livraison. Nous listons tous les fichiers que nous avons créés ou modifiés, avec une rapide description de leur contenu (des noms qui parlent !) ;
- Les librairies ou outils logiciels tiers utilisés (version, utilité, référence à l'éditeur/fabriquant) ;
- La description exacte du matériel ;
- La marche à suivre pour reconstruire le résultat final à partir de ce qui a été rendu ;
- Les éventuels login/password nécessaires pour utiliser le système dans son environnement de développement.
- Le modèle logique de données (MLD). Il contient les tables, les champs nommés de manière exhaustive avec leurs types et leur longueur ainsi que les relations entre les tables et leurs cardinalités.

5 Tests

Nous fournissons deux éléments concernant les tests :

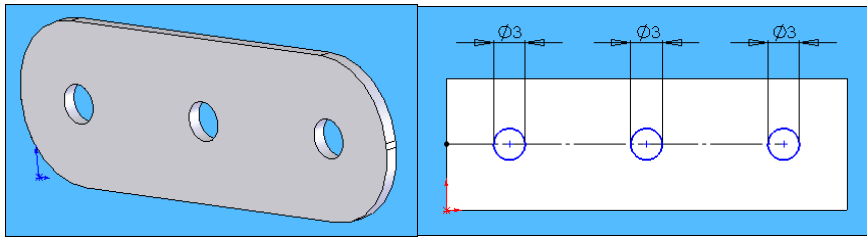
1. La Stratégie,
2. Les Résultats.

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

5.1 Niveau de test

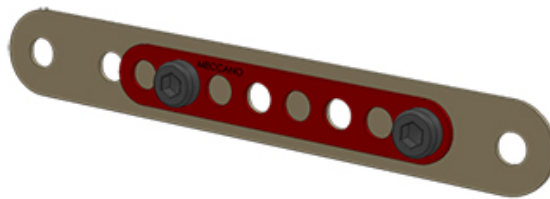
Nous appliquons jusqu'à trois niveaux de test :

- **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 individuellement 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.

5.2 Type de test

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

5.3 Stratégie de test

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.

5.4 Résultats de test

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. Chaque colonne donnera 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	

6 Journal de Travail et Journal de Bord

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

6.1 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

6.2 Le Journal de Bord

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

Exemple :

Date	Événement
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

7 Documentation de 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 2 de ce document
3. La planification initiale, telle que décrite au chapitre 3.2 de ce document
4. Les détails de réalisation, tels que décrits au chapitre 4 de ce document
5. Les détails de la livraison, telle que décrite au chapitre 9 de ce document
6. Le Journal de Bord, tel que décrit au chapitre 6.2 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 » résultant des rétro-planifications, ainsi qu'une comparaison de celui-ci avec le planning initial
 - La liste des problèmes connus
 - Un commentaire personnel sur l'ensemble du projet

8 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.

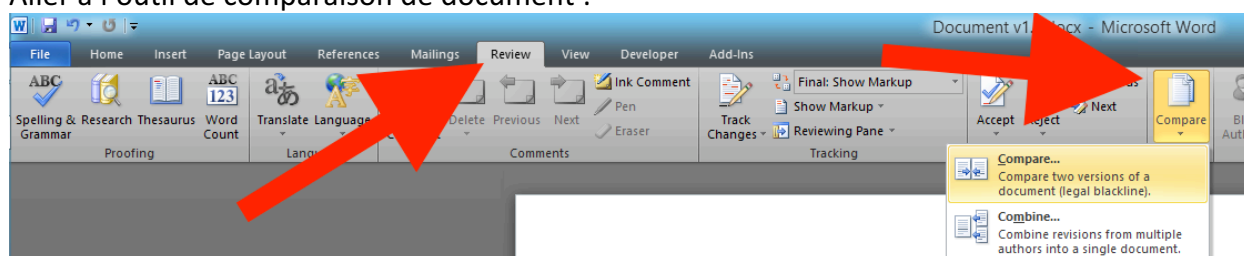
Exemple : si l'on travaille sur un document « Projet Alpha.docx » et que l'on publie la version 1.3, on fournira :

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

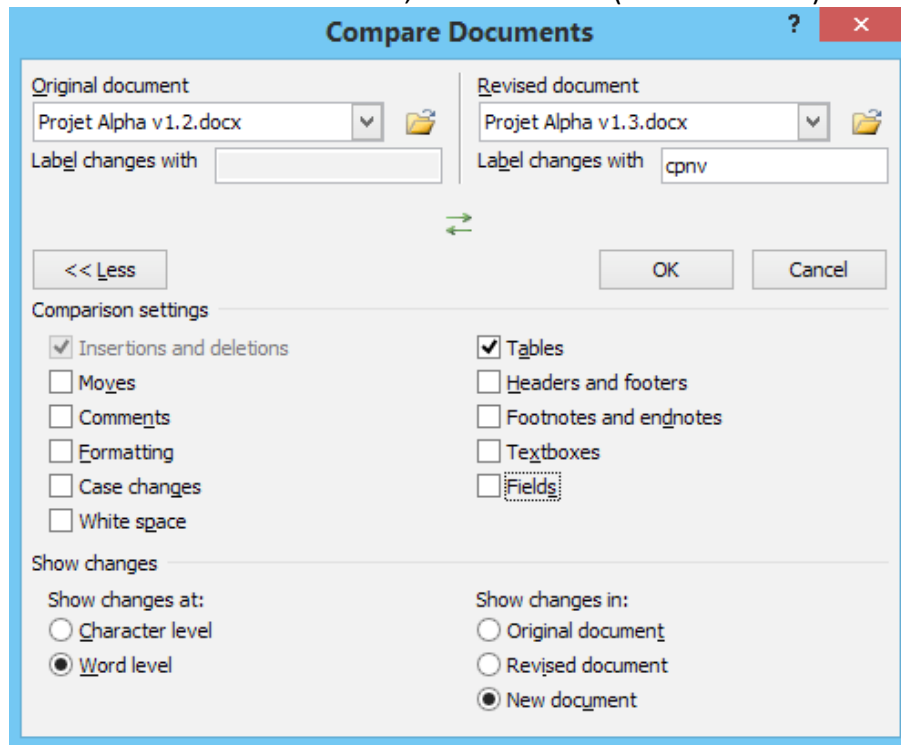
Remarque : ces fichiers n'ont pas besoin d'être enregistrés dans le repository Git, ce dernier contenant les originaux.

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

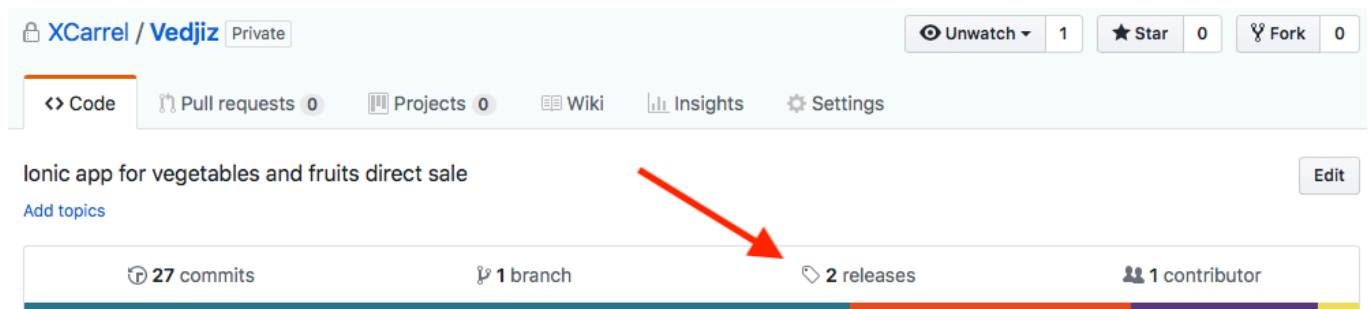
9 Livraison

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 :

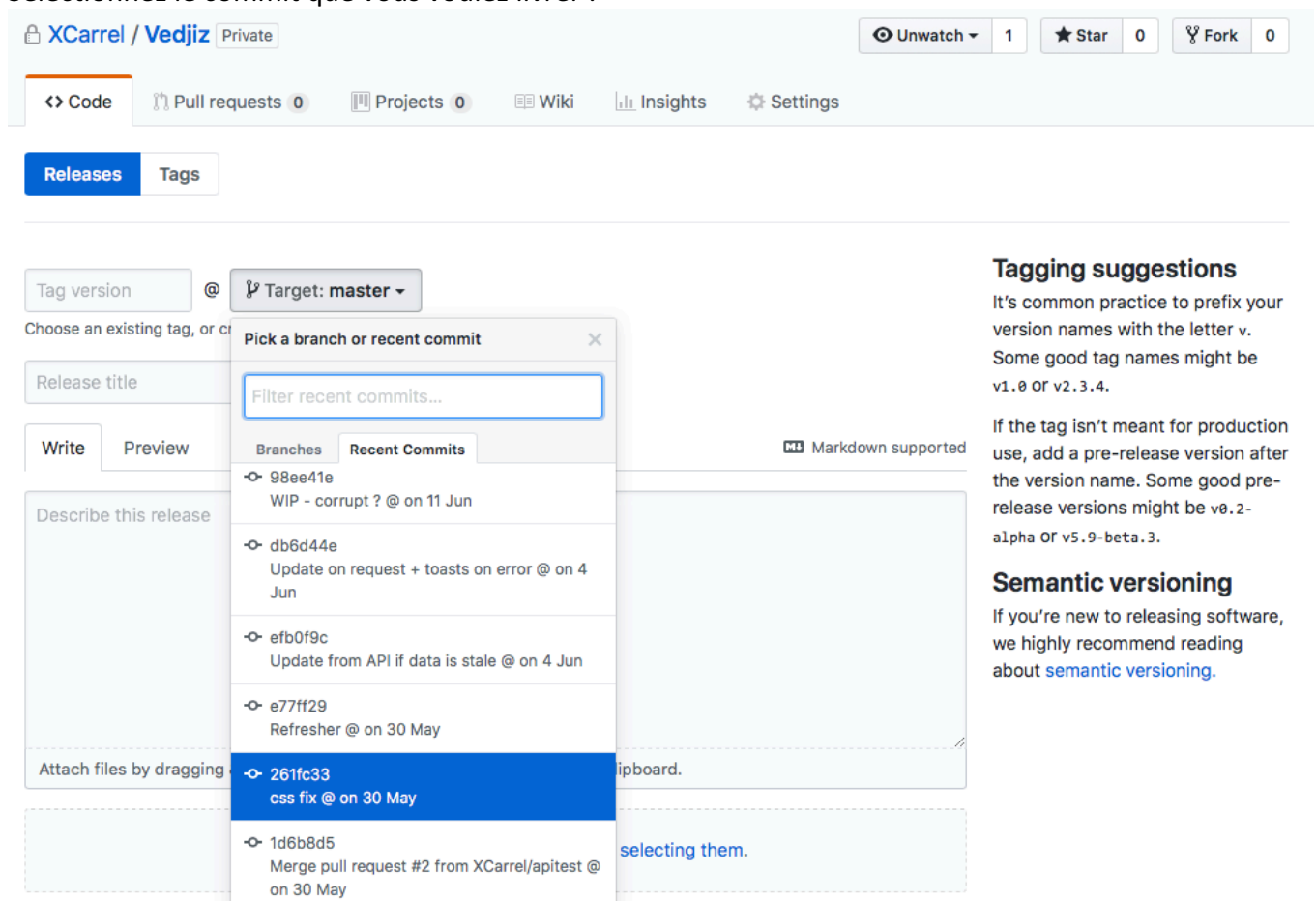
- Un numéro de version
- Un titre
- Une description

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 :

XCarrel / 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

XCarrel 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

- Page 28 -

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.

9.1.1 Planification initiale

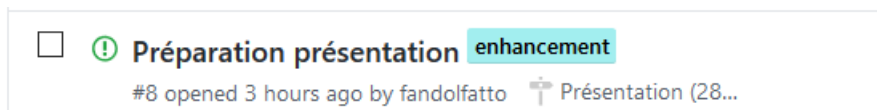
La planification initiale du projet est grossière car on ne dispose que de peu d'informations à ce stade du projet, comme cela était le cas aussi dans le mode Agile. Elle n'est typiquement constituée que de phases générales et de jalons majeurs tels que :

- 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• Échéance : 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• Échéance : 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• Échéance : 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.

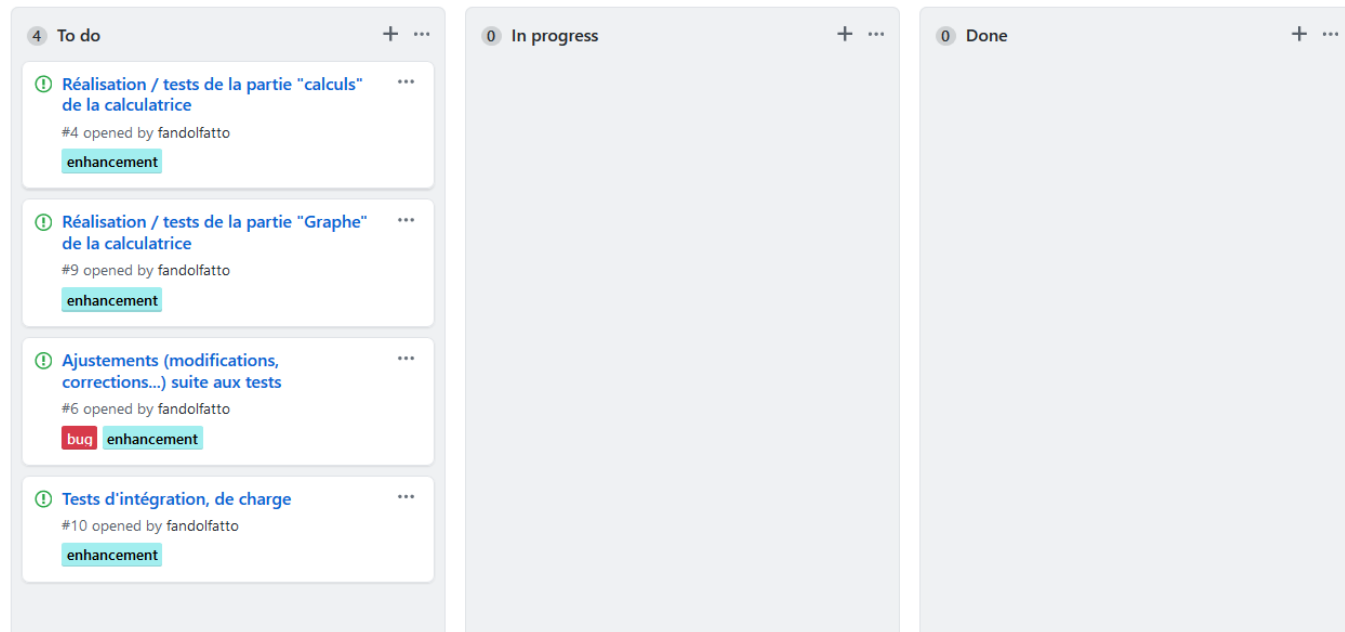


Chaque projet contient les « issues » qui le composent.

Exemple :

2. Réalisation / tests

Updated 8 minutes ago



9.1.2 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 ».