Conception, Projet

Cours 11 - Agile : Srum

Table des matières

Introduction au développement Agile

Le développement Agile est une approche de gestion de projet qui favorise des cycles courts de développement, une collaboration étroite entre les membres de l'équipe, l'implication constante des utilisateurs et la capacité à répondre rapidement aux changements.

Principes du développement Agile

L'Agile est guidé par un ensemble de 12 principes fondamentaux, comme énoncés dans le "Agile Manifesto".
Souce :
`https://agilemanifesto.org/iso/fr/manifesto.html`
Ces principes sont les suivants :

Principes du développement Agile

- 1. La priorité de l'équipe est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée.
- 2. L'équipe accueille les demandes de changement, même tard dans le projet. Les processus Agile exploitent le changement comme un avantage compétitif pour le client.
- 3. L'équipe livre des versions fonctionnelles de l'application le plus fréquemment possible, avec une préférence pour une cadence plus courte.
- 4. Les gestionnaires d'affaires et les développeurs travaillent ensemble quotidiennement tout au long du projet.
- 5. Les projets sont construits autour d'individus motivés. Il faut leur donner l'environnement et le soutien dont ils ont besoin, et leur faire confiance pour accomplir la tâche.
- 6. La méthode la plus efficace et efficiente pour transmettre de l'information à une équipe de développement, et au sein de cette équipe, est la conversation en face à face.

Principes du développement Agile

- 7. Un logiciel fonctionnel est la principale mesure de progrès.
- 8. Les processus Agile promeuvent un rythme de développement soutenable. Les parties prenantes, développeurs, et utilisateurs devraient être capables de maintenir ce rythme indéfiniment.
- 9. L'attention portée à l'excellence technique et à la bonne conception améliore l'agilité.
- 10. La simplicité, c'est-à-dire l'art de maximiser la quantité de travail non fait, est essentielle.
- 11. Les meilleures architectures, spécifications, et conceptions émergent toutes d'équipes auto-organisées.
- 12. À intervalles réguliers, l'équipe réfléchit à comment devenir plus efficace, puis s'ajuste et s'adapte en conséquence.

SCRUM

Scrum est une autre méthode Agile qui organise le travail en petits cycles de travail appelés sprints.

Chaque sprint a une durée définie (généralement de 1 à 4 semaines), et à la fin de chaque sprint, l'équipe doit produire un incrément potentiellement livrable.

Principes du SCRUM

- Transparence, inspection, adaptation
- Organisation en équipes auto-organisées et pluridisciplinaires
- Découpage du travail en petits éléments de valeur, priorisés dans un "backlog"
- Itérations courtes avec revue et adaptation (rétroaction et rétrospective) à chaque fin d'itération

SCRUM: Rôles et responsabilités

SCRUM définit trois rôles clés dans le cadre du développement de logiciels :

- Product Owner (Architecte)
- Scrum Master
- L'équipe de développement.

Product Owner (Architecte)

Le Product Owner est responsable de maximiser la valeur du produit résultant du travail de l'équipe de développement.

En théorie, il est la seule personne qui gère le Backlog du produit.

Il est en charge de comprendre les besoins des utilisateurs, des clients, du marché, et de définir les fonctionnalités du produit à réaliser par l'équipe de développement.

Scrum Master

Le Scrum Master est un leader serviteur pour l'équipe de développement.

Il aide tout le monde à comprendre les théories, pratiques, règles et valeurs de Scrum.

Il s'assure que l'équipe respecte et suit les principes et les processus de Scrum.

Le Scrum Master résout les problèmes qui empêchent l'équipe de réaliser ses objectifs.

Équipe de développement

L'équipe de développement est responsable de la réalisation du produit.

Il s'agit d'un groupe de professionnels qui travaillent ensemble pour livrer des versions potentiellement commercialisables du produit à la fin de chaque sprint (cycle de développement).

L'équipe de développement est auto-organisée, ce qui signifie qu'elle décide elle-même comment accomplir son travail.

L'équipe peut être multidisciplinaire, et peut incorporer plusieurs membres responsables de tâches "post" ou "pre" conception (product owner, vendeur, analyste de qualité, testeur, etc.)

SCRUM - les artefacts

En développement logiciel, le terme artefact fait référence aux informations clés nécessaires durant le développement d'un produit. Le cadre d'applications Scrum se compose de trois artefacts clés :

- Le carnet de produit (Product/Global Backlog)
- Le carnet de sprint (Sprint Backlog)
- L'incrément de produit.

Carnet de produit (Product/Global Backlog)

Le carnet de produit est une liste ordonnée de tout ce qui est connu pour être nécessaire dans le produit/globalement.

Il s'agit de l'unique source des exigences pour tout changement à apporter au produit.

Le Product Owner est responsable du carnet de produit, de son contenu, de la disponibilité et du classement.

Carnet de sprint (Sprint Backlog)

Le carnet de sprint est un ensemble d'éléments du carnet de produit qui est sélectionné pour le sprint, ainsi qu'un plan pour livrer l'incrément de produit et réaliser l'objectif du sprint.

Il est une prévision par l'équipe de développement de ce qui sera nécessaire pour atteindre l'objectif du sprint.

Incrément de produit

L'incrément est la somme de tous les éléments du carnet de produit complétés lors d'un sprint et de la valeur de tous les incréments des sprints précédents.

À la fin d'un sprint, l'incrément doit être en condition d'utilisation, ce qui signifie qu'il doit être "potentiellement livrable", indépendamment du fait que le Product Owner décide de le livrer ou non.

Par contre, un Product Owner qui fait travailler l'équipe de développement pour rien fait perdre des ressources à la compagnie. Ceci-dit, son directeur pourrait être la cause de ses actions s'il lui ordonne de le faire (raison politique, etc.).

SCRUM - les artefacts étendues

En plus des artefacts clés de Scrum (le carnet de produit, le carnet de sprint, et l'incrément de produit), il y a également des "artefacts étendus" qui sont généralement créés pour aider l'équipe à comprendre et gérer le travail.

Bien qu'ils ne soient pas explicitement exigés par Scrum, ils sont souvent considérés comme des pratiques exemplaires.

Les plus courants :

- Definition of Done (DoD)
- Burndown Charts
- Product Goal
- Raffinement du carnet de produit (Backlog Refinement)

Definition of Done (DoD)

La définition de "Fini" (DoD - Definition of Done) est une liste d'exigences que doit remplir un élément du carnet de produit pour être considéré comme terminé.

Cette liste peut varier d'une équipe à l'autre, ou même d'un produit à l'autre, mais elle doit être clairement définie et acceptée par tous les membres de l'équipe.

Burndown Charts

Les graphiques de progression (Burndown Charts) représentent le travail restant à faire par rapport au temps.

Ils sont souvent utilisés dans Scrum pour visualiser la quantité de travail qui reste à faire dans un carnet de produit ou un carnet de sprint, et pour prédire à quel moment le travail sera terminé.

Product Goal

Le But du Produit (Product Goal) est une vision à long terme de ce que l'équipe s'efforce d'atteindre avec le produit.

Il est défini par le Product Ownert et sert de guide pour l'équipe lors de la priorisation des éléments du carnet de produit.

Raffinement du carnet de produit (Backlog Refinement)

Le raffinement du carnet de produit est le processus d'examen et de mise à jour des éléments du carnet de produit.

Il s'agit d'une activité continue qui aide l'équipe à comprendre les exigences et à préparer les éléments du carnet de produit pour les sprints futurs.

Raffinement du carnet de produit (Backlog Refinement)

Le moment de le raffiner est arbitraire à l'équipe et le Product Owner, tant que ceci est fait avant que le nouveau sprint commence.

Il est recommandé de le faire à un moment que l'équipe apprécie et que le Product Owner aussi pour une meilleure collaboration.

Le raffinement du carnet de produit est un aspect essentiel de Scrum qui aide à garantir que le travail est clairement défini et prêt à être entrepris.

Il facilite la planification du sprint et aide l'équipe à maintenir son rythme et sa productivité tout au long du projet.

Raffinement du carnet de produit (Backlog Refinement)

Le raffinement du carnet de produit est un aspect essentiel de Scrum qui aide à garantir que le travail est clairement défini et prêt à être entrepris.

Il facilite la planification du sprint et aide l'équipe à maintenir son rythme et sa productivité tout au long du projet.

Kanban

Kanban est une méthode agile axée sur la gestion visuelle des tâches de travail.

Elle a été initialement développée dans les usines de Toyota dans les années 1940 pour améliorer et maintenir un haut niveau de production et de qualité.

Aujourd'hui, Kanban est largement utilisé dans divers secteurs, notamment le développement de logiciels.

Kanban - principes

Kanban est basé sur trois principes fondamentaux :

- **Visualiser le travail :** En utilisant un tableau Kanban, vous pouvez visualiser l'ensemble du flux de travail, ce qui facilite l'identification des goulots d'étranglement et des problèmes.
- **Limiter le travail en cours :** Chaque étape du flux de travail a une limite fixe de tâches qui peuvent être en cours à un moment donné. Cela aide à maintenir un flux constant de travail et à éviter les surcharges.
- **Améliorer continuellement :** L'équipe doit toujours chercher des moyens d'améliorer le flux de travail et d'augmenter l'efficacité.

Le tableau Kanban

Le tableau Kanban est l'outil principal de la méthode Kanban.

Il est généralement divisé en différentes colonnes qui représentent les différentes étapes du flux de travail.

Chaque tâche est représentée par une carte qui se déplace de gauche à droite sur le tableau à mesure qu'elle progresse à travers le flux de travail.

Les tableaux Kanban peuvent être personnalisés pour s'adapter à différents types de flux de travail et peuvent également inclure des sous-colonnes pour plus de détails.

Utilisation de Kanban dans le développement de logiciels

Dans le développement de logiciels, Kanban peut être utilisé pour gérer le flux de travail de diverses activités, comme la rédaction de code, les tests, l'intégration continue et la livraison.
Il peut également être utilisé pour gérer le carnet de produit et suivre le progrès des fonctionnalités ou des histoires d'utilisateurs (user stories).

Utilisation de Kanban dans le développement de logiciels

Il est important de noter que, bien que Kanban et Scrum soient tous deux des méthodes agiles, ils diffèrent en plusieurs points clés.

Par exemple, contrairement à Scrum, Kanban ne fonctionne pas en sprints mais permet un flux de travail continu.

Cela peut rendre Kanban plus flexible et adaptable à certaines équipes ou situations.

SCRUM vs. Kanban

Il est important de noter que, bien que Kanban et Scrum soient tous deux des méthodes agiles, ils diffèrent en plusieurs points clés.

Par exemple, contrairement à Scrum, Kanban ne fonctionne pas en sprints mais permet **un flux de travail continu**. Cela peut rendre Kanban plus flexible et adaptable à certaines équipes ou situations.

SCRUM vs. Kanban

Scrum et Kanban sont tous deux des cadres de développement Agile, mais ils diffèrent dans leur approche de la gestion de projet.

Les équipes Agile peuvent choisir d'utiliser l'un ou l'autre, ou même une combinaison des deux, en fonction de leurs besoins spécifiques.

Similitudes entre Scrum et Kanban

- **Les deux sont des méthodologies Agile :** Scrum et Kanban sont tous deux basés sur les principes du Manifeste Agile, qui mettent l'accent sur les personnes, la collaboration, le client et la flexibilité.
- **Les deux utilisent des tableaux visuels :** Scrum utilise un tableau Scrum et Kanban utilise un tableau Kanban pour visualiser le flux de travail et le progrès de l'équipe.
- **Les deux se concentrent sur la livraison continue :** Scrum et Kanban visent tous deux à livrer des produits de qualité de manière continue et efficace.

Différences entre Scrum et Kanban - Les cycles de travail

Les cycles de travail : Scrum fonctionne en sprints, qui sont des cycles de travail fixes généralement de deux semaines. Kanban, en revanche, n'a pas de cycles fixes et se concentre sur la livraison continue.

Différences entre Scrum et Kanban - Les rôles dans l'équipe

Les rôles dans l'équipe : Dans Scrum, il y a des rôles clairement définis (Product Owner, Scrum Master, Team). En Kanban, il n'y a pas de rôles spécifiques.

Différences entre Scrum et Kanban - Le travail en cours

Le travail en cours : Scrum limite le travail en cours par sprint, tandis que Kanban limite le travail en cours pour chaque état de travail.

Différences entre Scrum et Kanban - Les changements durant le cycle

Les changements durant le cycle : Dans Scrum, le backlog du sprint est supposé rester constant pendant le sprint. Dans Kanban, les nouvelles tâches peuvent être ajoutées à tout moment tant que la limite du travail en cours n'est pas dépassée.

Exemple : Mise en situation

Imaginons que vous soyez un chef de projet dans une entreprise de développement de logiciels. Votre entreprise vient de décrocher un contrat pour développer une nouvelle application mobile.

Pour gérer ce projet, vous décidez de former une équipe Agile.

Voici comment cela va se dérouler.

Création d'une équipe Agile - Réunion de lancement du projet

Vous commencez par rassembler votre équipe, composée de membres aux compétences variées - développeurs, concepteurs, testeurs, etc. Lors de cette première réunion, plusieurs choses se produisent :

- **Présentations :** Chaque membre de l'équipe se présente et partage son rôle et ses compétences.
- **Présentation du projet :** Vous présentez le contexte et les objectifs du projet à l'équipe.
- **Définition de l'équipe :** L'équipe choisit un nom, définit ses valeurs et sa mission.
- **Alignement sur les définitions :** L'équipe s'accorde sur certaines définition et critère, (i.e. def. of done au sprint 0).
- **Présentation des outils :** backlog, stand-ups, outils de suivi... Le chargé de projet s'assure que tous le monde soit au courant du contexte/outils à la dispostion du projet.

Création d'une équipe Agile - Sprint 0

Votre équipe Agile est prête à commencer son travail sur le nouveau projet d'application mobile.

Cependant, avant de commencer à développer des fonctionnalités, il y a un certain nombre de tâches à accomplir pour se préparer. C'est ce que l'on appelle le "sprint 0".

Création d'une équipe Agile - Sprint 0

Voici comment cela pourrait se dérouler :

- Définir la vision du produit
- Créer le backlog du produit
- Estimer les histoires d'utilisateurs
- Définir la définition de "done"
- Mise en place de l'environnement de développement
- Planification de l'architecture

Création d'une équipe Agile - Sprint 0

Une fois le sprint 0 terminé, l'équipe est prête à commencer le développement en sprint 1.

Les tâches effectuées pendant le sprint 0 ne produisent généralement pas de valeur client directe, mais elles sont essentielles pour garantir que l'équipe est bien préparée et que le projet est sur la bonne voie dès le départ (et de fin).

Les prochaines diapositives vont décrire chaque point :

Création d'une équipe Agile - Sprint 0 - Définir la vision du produit

Définir la vision du produit :

L'équipe se réunit avec le client pour comprendre la vision du produit.

Quel est l'objectif de l'application ?

Qui sont les utilisateurs cibles ?

Quels sont les besoins de ces utilisateurs que l'application devra répondre ?

Cette vision guidera toutes les décisions futures de l'équipe.

Création d'une équipe Agile - Sprint 0 - Créer le backlog du produit

Créer le backlog du produit :

Une fois la vision du produit établie, l'équipe commence à décomposer cette vision en fonctionnalités spécifiques que l'application devra avoir.

Ces fonctionnalités sont écrites sous forme d'histoires d'utilisateurs et ajoutées au backlog du produit.

Création d'une équipe Agile - Sprint 0 - Estimer les histoires d'utilisateurs

Estimer les histoires d'utilisateurs :

L'équipe doit estimer le temps qu'il faudra pour développer chaque histoire d'utilisateur.

Ces estimations aideront à la planification des sprints futurs.

Création d'une équipe Agile - Sprint 0 - Définir la définition de "done"

Définir la définition de "done" :

Qu'est-ce que cela signifie pour une histoire d'utilisateur d'être "terminée" ?

Est-ce que cela signifie simplement que le code a été écrit, ou cela signifie-t-il que le code a été testé, documenté, et accepté par le client ?

L'équipe doit se mettre d'accord sur cette définition.

Création d'une équipe Agile - Sprint 0 - Mise en place de l'environnement de développement

Mise en place de l'environnement de développement :

Pendant le sprint 0, l'équipe mettra en place l'environnement de développement nécessaire pour le projet.

Cela peut inclure des outils de développement, des serveurs, des bases de données, etc.

NB : Normalement, l'architecte (ou le directeur) ont déjà décidé les technologies.

Création d'une équipe Agile - Sprint 0 - Planification de l'architecture

Planification de l'architecture : En fonction de la vision du produit et du backlog, l'équipe planifie l'architecture générale du système.

NB : Normalement, l'architecte (ou le directeur) ont déjà décidé comment l'architecture de l'application est fondée.

Déroulement d'un sprint

Maintenant que votre équipe Agile est prête à commencer son premier sprint, après avoir terminé le sprint 0, voici comment la manière dont tout va se dérouler :

- Début
  - Planification du sprint (Sprint Planning)
- Déroulement
  - Développement
  - Mêlées quotidiennes (Daily Scrum)
- Fin
  - Revue du sprint (Sprint Review)
  - Rétrospective du sprint (Sprint Retrospective)

Déroulement d'un sprint - Début

Planification du sprint (Sprint Planning) :

L'équipe se réunit pour planifier le travail à accomplir lors du prochain sprint. Les histoires d'utilisateurs du backlog du produit sont discutées, estimées, et priorisées.

L'équipe s'engage alors à livrer un certain nombre de ces histoires d'utilisateurs à la fin du sprint.

Déroulement d'un sprint - Déroulement

Développement :

L'équipe commence à travailler sur les histoires d'utilisateurs.

Cela peut impliquer la conception, le codage, le test et la documentation de la fonctionnalité.

Déroulement d'un sprint - Déroulement

Mêlées quotidiennes (Daily Scrum) :

Chaque jour (ou 2-3 fois par semaines), l'équipe se réunit brièvement pour discuter de ce qui a été fait la veille, de ce qui sera fait aujourd'hui et de tous les obstacles éventuels.

Déroulement d'un sprint - Fin

Revue du sprint (Sprint Review) :

À la fin du sprint, l'équipe présente le travail accompli au client lors de la revue du sprint.

C'est l'occasion pour l'équipe de recevoir des commentaires sur le travail qu'elle a réalisé.

Déroulement d'un sprint - Fin

Rétrospective du sprint (Sprint Retrospective) :

Après la revue du sprint, l'équipe se réunit pour discuter de ce qui a bien fonctionné, de ce qui pourrait être amélioré et de la manière dont ils peuvent mettre en œuvre ces améliorations au cours du prochain sprint.

Déroulement d'un sprint - Ensuite ?

Une fois la rétrospective terminée, l'équipe commence le processus à nouveau avec la planification du prochain sprint.

Chaque sprint est une opportunité pour l'équipe de livrer de la valeur au client et d'apprendre de l'expérience pour s'améliorer.

GitHub Project

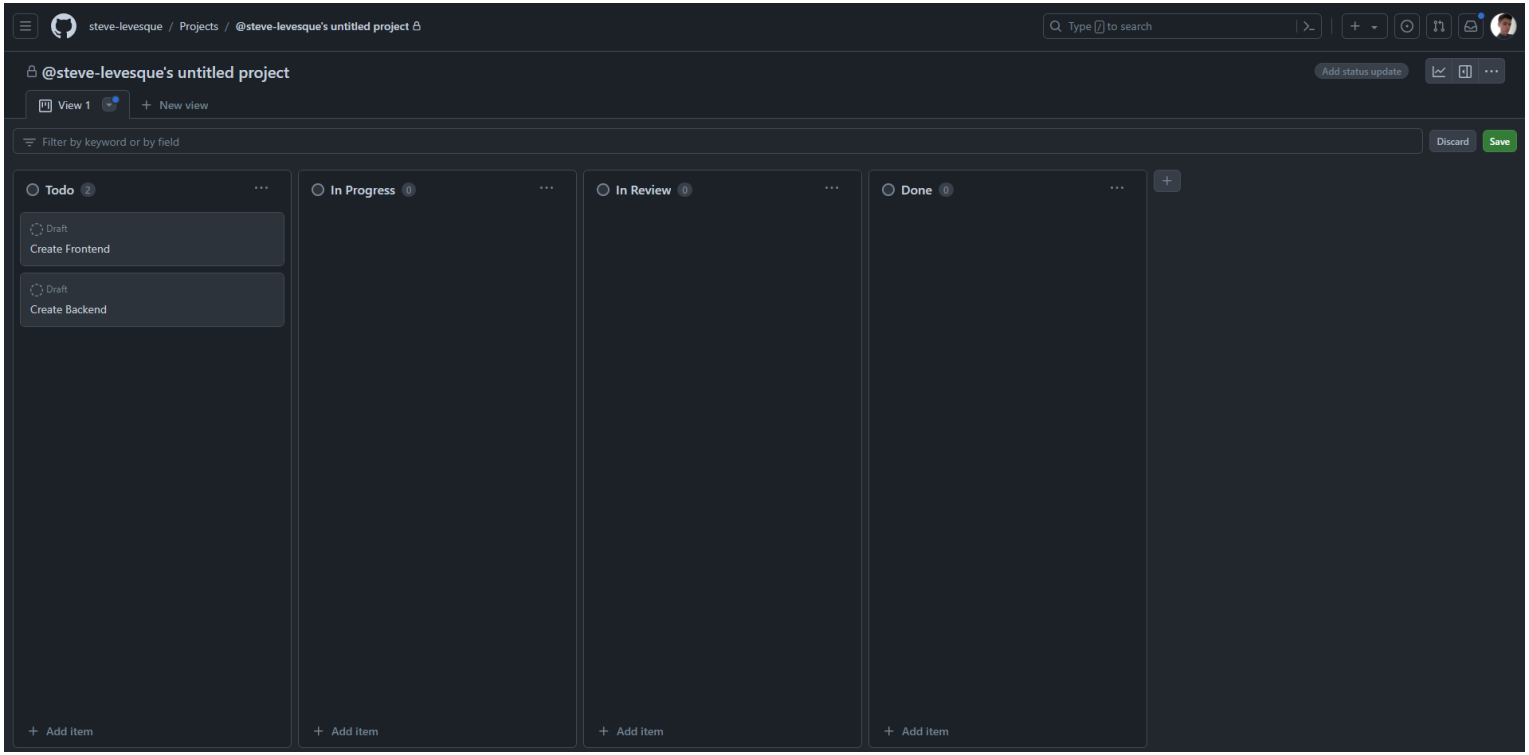
GitHub Projects est un outil de gestion de projet intégré à la plateforme GitHub.

Il s'inspire largement du système Kanban pour aider les équipes à organiser et à suivre leur travail.

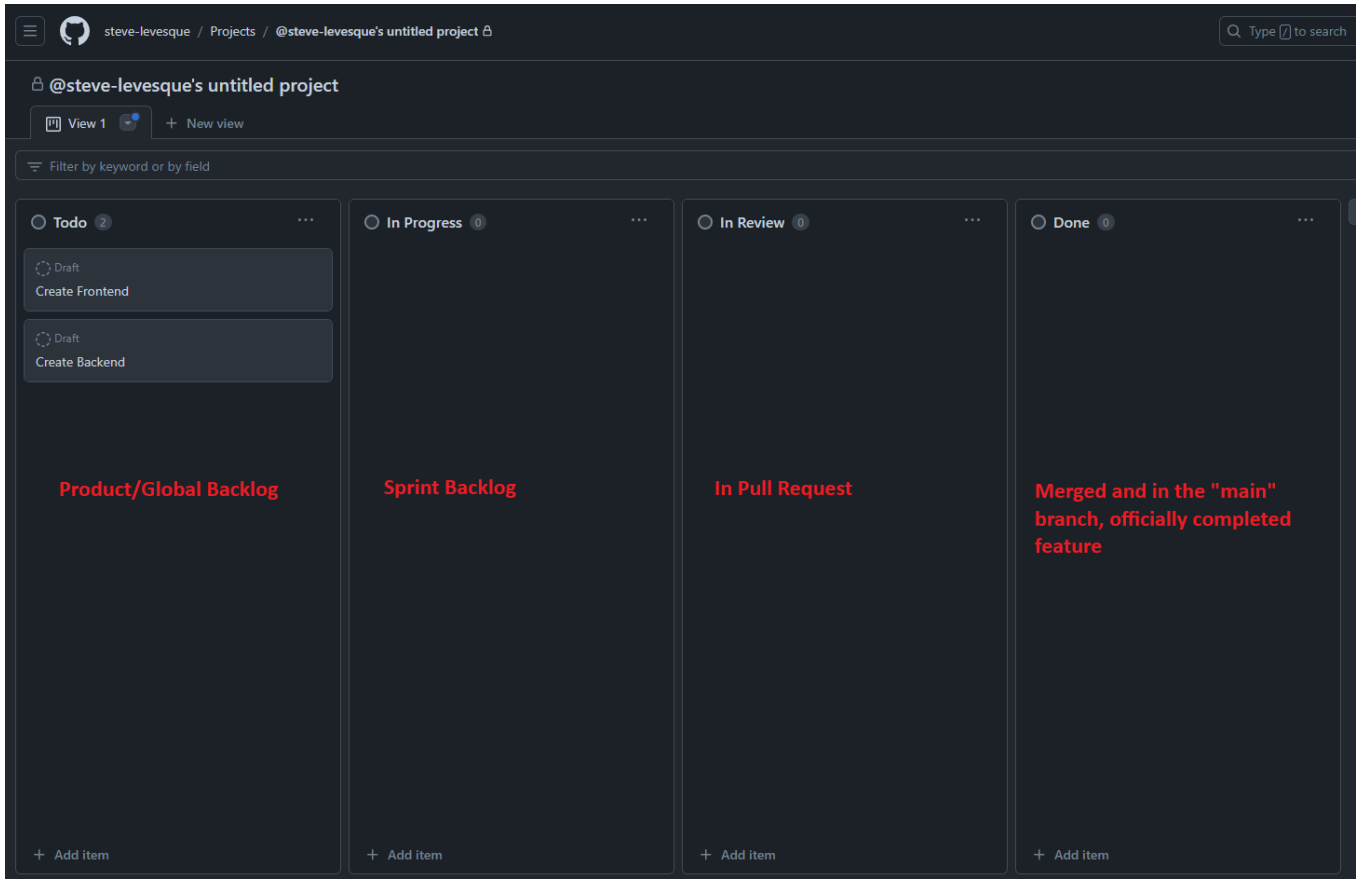
Nous allons maintenant mettre en pratique ce qu'on a apprit.

Veuillez suivre les isntructions du professeur.

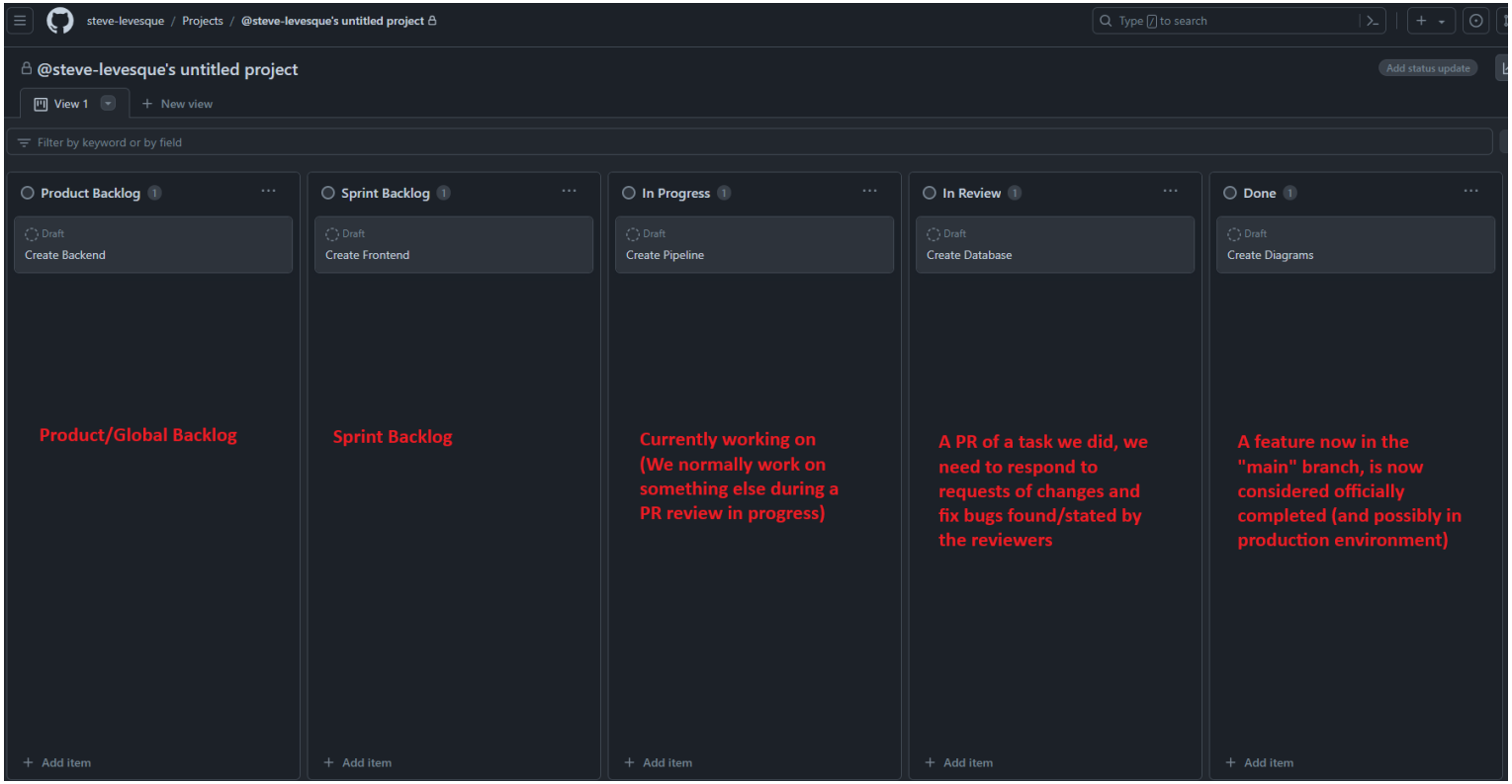
GitHub Project - Détails



GitHub Project - Détails - Exemple simple



GitHub Project - Détails - Exemple avec distinctions claires



Bibliographie

- https://github.com/