DEVTeamX

DESCRIPTION DU SYSTÈME KANBAN

Tchèssi Pre

05-10-2023

I. DESCRIPTION DU TABLEAU DE KANBAN

Plateforme : Notre tableau Kanban virtuel est hébergé sur Trello, spécifiquement adapté pour répondre aux besoins complexes de la **DevTeamX**. >> Lien vers le tableau Kanban

Objectif : Ce tableau est un outil central pour assurer la **transparence**, **la collaboration** et **le suivi** des tâches de l'équipe. Il est conçu pour :

- Visualiser en temps réel l'avancement des travaux.
- Identifier rapidement les goulets d'étranglement et les tâches bloquées.
- Assurer une communication fluide entre les membres de l'équipe et les autres parties prenantes.
- Faciliter une livraison continue, rapide, et efficace des tâches tout en maintenant une haute qualité.

Intégrations : Notre tableau Kanban sur **Trello** est intégré avec les autres outils que nous utilisons, tels que **Slack** pour les notifications, Confluence pour la documentation, et **Git** pour le suivi des versions.

Accessibilité : Tous les membres de DevTeamX ont accès au tableau. De plus, des droits d'accès spécifiques sont configurés pour permettre à certaines parties prenantes (par exemple le Product Owners) de visualiser, commenter ou modifier certaines tâches selon leurs rôles.

II. STRUCTURE DU TABLEAU DE KANBAN

Le tableau est soigneusement structuré pour refléter notre flux de travail unique. Il est divisé en plusieurs colonnes, chacune représentant une étape spécifique :

- **Backlog:**
  - Objectif: Stocker toutes les tâches et fonctionnalités envisagées pour le futur.
  - Détails:
    - Chaque tâche est ajoutée avec une brève description, une étiquette de catégorie, et un membre responsable pour la supervision.
    - Les tâches peuvent également avoir une priorité basée sur des étiquettes de couleur. Les discussions initiales et les suggestions d'équipe sont ajoutées sous forme de commentaires.
    - Des pièces jointes, telles que des maquettes ou des documents de spécification, peuvent être ajoutées si disponibles.
- **À Faire (Cette semaine):**
  - Objectif: Organiser le travail à exécuter pendant la semaine en cours.
  - Détails:
    - Tâches transférées du Backlog basées sur la priorité et la capacité de l'équipe.
    - Des dates d'échéance précises sont attribuées.
    - Une liste de contrôle de pré-développement est ajoutée pour s'assurer que tous les prérequis sont en place.
- **En Cours:**
  - Objectif: Suivre le travail que les développeurs ont actuellement en main.
  - Détails:

        - Les développeurs ajoutent leur nom à une tâche lorsqu'ils commencent à travailler dessus.
        - L'avancement est indiqué via une liste de contrôle intégrée.
        - Les blocages ou les questions sont ajoutés sous forme de commentaires pour faciliter la communication.
- **En Test:**
    - Objectif: Assurer que chaque fonctionnalité ou correction fonctionne comme prévu.
    - Détails:
        - Une fois le développement d'une tâche terminé, elle est déplacée ici.
        - Les testeurs ajoutent leurs retours et leurs rapports de bugs sous forme de commentaires.
        - Si des bugs sont découverts, la tâche peut être renvoyée en développement ou un nouveau ticket est créé dans le backlog.
- **En Revue (Code review):**
    - Objectif: Garantir la qualité et la maintenabilité du code.
    - Détails:
        - Les pairs examinent le code et fournissent des commentaires.
        - Les outils d'intégration continue peuvent être utilisés pour signaler les problèmes automatiquement.
        - Une fois la revue terminée et les éventuelles corrections apportées, la tâche est déplacée vers la colonne "Fait" ou renvoyée en développement si nécessaire.
- **Fait (Terminé):**
    - Objectif: Archiver les tâches achevées avec succès.
    - Détails:
        - Une fois qu'une tâche a passé toutes les étapes précédentes avec succès, elle est placée ici.
        - C'est également un espace pour réfléchir lors des rétrospectives, pour comprendre ce qui a bien fonctionné et ce qui pourrait être amélioré.

“Ce système, lorsqu'il est mis en place et suivi régulièrement, permet à DevTeamX de travailler de manière fluide et transparente, tout en s'assurant que rien n'est négligé.”

III. WIP (Work In Progress) Limit pour DevTeamX

- - - - X

- **Backlog:**
  - Limite: Pas de limite.
  - Rationale:
    - Le Backlog est essentiellement une banque d'idées, de fonctionnalités et de bugs à traiter. Il est donc logique de ne pas avoir de limite ici.
    - Cela permet une planification et une priorisation flexibles pour les sprints futurs.
- **À Faire (Cette semaine):**
  - Limite: 10 tâches.
  - Rationale:
    - Cette limite assure que l'équipe ne surcharge pas son pipeline et se concentre sur les tâches les plus cruciales.
    - Elle aide à garder une vue claire des priorités immédiates et empêche l'éparpillement des efforts.
- **En Cours:**
  - Limite: 5 tâches.
  - Rationale:
    - Limiter le nombre de tâches en développement assure que les développeurs ne sont pas débordés et peuvent se concentrer pleinement sur chaque tâche.
    - Cela garantit également une meilleure qualité du code, car moins de multitâche signifie moins d'erreurs.

- **En Test:**
    - Limite: 3 tâches.
    - Rationale:
        - Le test est une étape cruciale qui nécessite une attention méticuleuse. Une limite aide à s'assurer que les testeurs peuvent se concentrer sur un nombre limité de tâches à la fois.
        - Cela favorise une meilleure qualité d'assurance et une détection rapide des bugs.
- **En Revue (Code review):**
    - Limite: 4 tâches.
    - Rationale:
        - La revue de code demande du temps et une attention aux détails. En limitant le nombre de tâches, on s'assure que les revues sont complètes et minutieuses.
        - Cela favorise un code de meilleure qualité et une cohérence dans les standards de codage.
- **Fait (Terminé):**
    - Limite: Pas de limite.
    - Rationale:
        - Une fois qu'une tâche est complétée, elle est archivée ici. Il n'est pas nécessaire d'imposer une limite, car cela ne représente pas le travail en cours.

“ L'utilisation de ces limites WIP permet à DevTeamX de gérer efficacement son flux de travail, d'éviter les surcharges et de garantir que chaque étape du processus reçoit l'attention appropriée.”

IV. RÈGLES DE FLUX

- **Assignation:**
  - **Détail:** Chaque tâche, avant d’être déplacée de Backlog" à "À Faire", doit être assignée à un membre spécifique de l’équipe.
  - **Rationale:**
    - Assure que chaque tâche ait un responsable clairement défini.
    - Permet un suivi efficace et réduit la confusion concernant les responsabilités.
  - **Exception:** Si une tâche nécessite une expertise ou une décision spéciale, elle peut être assignée à un groupe ou un rôle plutôt qu’à un individu jusqu'à ce que le rôle approprié soit déterminé.

- **Documentation:**
  - **Détail:** Tout changement majeur dans une tâche, que ce soit une progression, un obstacle ou une mise à jour, doit être noté dans les commentaires de la carte. Tout document ou lien pertinent doit y être joint.
  - **Rationale:**
    - Assure une traçabilité complète des événements et décisions liés à la tâche.
    - Aide à la communication au sein de l'équipe et fournit un historique pour les futurs référencements.
  - **Fréquence:** Au minimum, une mise à jour hebdomadaire pour chaque carte, sauf si elle est déplacée vers une nouvelle colonne ou si un événement majeur survient.

- **Blocage:**
    - **Détail:** Une tâche bloquée est identifiée avec un label rouge ou une icône spéciale. Elle doit également avoir une description claire de la raison du blocage dans les commentaires.
    - **Rationale:**
        - Les blocages peuvent ralentir ou arrêter le flux de travail. Les identifier rapidement permet d'y remédier efficacement.
        - La visibilité est essentielle pour que toute l'équipe soit au courant et puisse collaborer pour trouver une solution.
    - **Escalade:** Si une tâche est bloquée pendant plus de 48 heures sans résolution, elle doit être escaladée au gestionnaire ou à l'équipe de direction.

- **Transition entre colonnes:**
    - **Détail:** Avant qu'une tâche ne soit déplacée d'une colonne à l'autre, elle doit répondre aux critères définis pour la colonne cible. Par exemple, une tâche ne peut être déplacée vers "En Test" que si le développement est complètement terminé.
    - **Rationale:**
        - Assure la qualité et l’intégrité du processus de développement.
        - Évite les retours en arrière et les confusions concernant l'état de la tâche.

“ En suivant ces règles de flux, **DevTeamX** s'assure que le processus de travail est transparent, efficace et que toutes les tâches sont traitées avec le plus grand soin et la plus grande attention.”

V. INDICATEURS DE PERFORMANCE

- **Lead Time:**
  - Définition: Durée totale entre la création d'une tâche (lorsqu'elle est ajoutée au backlog) et son achèvement (lorsqu'elle est déplacée vers "Fait").
  - Importance:
    - Indique la rapidité globale avec laquelle l'équipe est capable de livrer des fonctionnalités ou des corrections.
    - Un long Lead Time pourrait indiquer des goulots d'étranglement dans le processus ou des retards dans les phases initiales du cycle de développement.
  - Méthode de mesure: Utiliser la date/horodatage de la création de la tâche et la comparer à la date/horodatage de son achèvement.

- **Cycle Time:**
  - Définition: Durée entre le moment où le travail commence réellement sur une tâche (lorsqu'elle est déplacée vers "En Développement") et son achèvement.
  - Importance:
    - Reflète l'efficacité du processus de développement et de test.
    - Un Cycle Time plus court suggère une livraison plus rapide, tandis qu'un Cycle Time plus long pourrait indiquer des problèmes ou des inefficacités dans ces phases.
  - Méthode de mesure: Utiliser la date/horodatage de quand la tâche est déplacée vers "En Développement" et la comparer à la date/horodatage de son achèvement.

- **Taux de bugs:**
  - **Définition:** Proportion du nombre de bugs identifiés par rapport au nombre total de tâches (bugs et fonctionnalités) terminées dans une période donnée.
  - **Importance:**
    - Un indicateur de la qualité du code produit par l'équipe.
    - Un taux de bugs élevé pourrait indiquer des problèmes avec le processus de développement, des tests insuffisants ou des spécifications mal définies.
  - **Méthode de mesure:**
    - **Numérateur:** Nombre total de bugs identifiés pendant une période (par exemple, un mois).
    - **Dénominateur:** Nombre total de tâches (bugs + fonctionnalités) terminées pendant cette période.
    - **Formule:**

$$Taux\ de\ bugs\ =\ (Nombre\ de\ bugs\ identifiés\ /\ Nombre\ total\ de\ tâches\ terminées)\ *\ 100$$

Ces indicateurs, lorsqu'ils sont suivis régulièrement, fournissent une image claire des performances de l'équipe et des domaines potentiels d'amélioration. Il est essentiel pour DevTeamX de les réviser périodiquement et d'ajuster les processus en conséquence pour assurer une livraison de haute qualité et efficace.

CONCLUSION

“ Le tableau Kanban est un outil essentiel pour visualiser et optimiser le flux de travail de DevTeamX. Grâce à une structure bien définie, des limites de WIP claires et des règles de flux, l'équipe peut s'assurer que les tâches sont traitées efficacement tout en évitant le surmenage. De plus, les indicateurs de performance tels que le Lead Time, le Cycle Time et le taux de bugs permettent une évaluation continue de la qualité et de l'efficacité du processus. En adoptant et en adaptant régulièrement ces méthodologies, DevTeamX est bien placée pour améliorer constamment ses livrables tout en garantissant la satisfaction des parties prenantes. ”