Cycle de vie d'un projet informatique

BUT RT 3

Jonathan Weber

Automne 2024

Contenu du module

- Volume horaire : 12h15
  - Cours : 1h45
  - TP : 10h30
- Objectif :
  - Appréhender toutes les étapes d'un projet informatique
  - En lien avec la SAÉ 5.02 : Piloter un projet informatique
- Contenu
  - Analyse du besoin
  - Mise en place d'un environnement de travail collaboratif
  - Choix des environnements de développement
  - Mise en place des procédures de tests unitaires
  - Production de documentations
  - Mise en production

Qui suis-je ?

- Professeur des Universités à l'UHA
- Enseignant à l'ENSISA :
  - Responsable de la spécialité Informatique et Réseaux
  - IA, Deep Learning, Sécurité, Programmation
- Chercheur à l'IRIMAS
  - Responsable de l'équipe Modélisation et Science des Données (MSD)
  - Deep Learning pour l'analyse de séries temporelles et d'images
- RSSI et FSD de l'UHA

Analyse du besoin

Objectifs

- Identifier et comprendre les besoins et exigences du client
- Cerner précisément ce que doit réaliser le logiciel
- Éviter les malentendus futurs et les changements coûteux en cours de projet
- Permettre une planification réaliste du temps, du budget et des ressources
- Poser les bases du développement et du test du logiciel

Cahier des Charges

- **Objectif :**
  - Définir clairement les attentes et les objectifs du logiciel
  - Servir de référence tout au long du cycle de développement
- **Contenu typique :**
  - Introduction et contexte
  - Objectifs du logiciel
  - Description détaillée des fonctionnalités
  - Contraintes techniques et fonctionnelles
  - Critères d'acceptation et de validation
- **Importance :**
  - Facilite la communication entre les parties prenantes
  - Aide à prévenir les divergences et les malentendus
  - Offre une base pour l'estimation des coûts et des délais

Stratégie de mise en production

- Objectifs :
  - Minimiser les risques associés au déploiement
  - Assurer la continuité des services existants
  - Faciliter une transition fluide vers la nouvelle version
- Stratégies courantes :
  - **Big Bang** : mise en production simultanée sur tous les systèmes
  - **Rolling upgrade** : déploiement progressif pour certaines parties du système
  - **Canary** : déploiement sur un sous-ensemble restreint d'utilisateurs pour évaluer la performance et récolter des feedbacks
  - **Blue-green** : maintien de deux environnements de production, l'un servant de backup pendant que le nouveau est déployé
- Facteurs :
  - Complexité du logiciel
  - Ressources disponibles (temps, budget, personnel)
  - Tolérance au risque de l'organisation

Prise en Compte des Problèmes de Sécurité

- Objectifs :
  - Éviter les brèches de sécurité
  - Assurer la conformité aux réglementations et normes
  - Construire une base solide pour un environnement sécurisé
- Domaines clés à considérer :
  - Authentification et autorisation
  - Gestion des accès et des identités
  - Cryptographie pour la protection des données
  - Sécurité des infrastructures et des réseaux
- Approches communes :
  - Analyses des risques et évaluations de vulnérabilités
  - Développement sécurisé dès le début (Secure by Design)
  - Tests de sécurité réguliers (tests d'intrusion, revues de code)
- Ressources humaines :
  - Inclure des experts en sécurité dans l'équipe de projet
  - Formation et sensibilisation de l'équipe aux bonnes pratiques de sécurité

Mise en place d'un environnement de travail collaboratif

Plateformes collaboratives dédiées au développement

- **Objectif :** Faciliter collaboration et gestion du développement
- **Gestion des branches :**
  - Création/suivi des branches développement et production
  - Fusion des branches après revue et tests appropriés
- **Création de clones :**
  - Facilitation du travail parallèle sur des fonctionnalités distinctes
  - Prévention des conflits grâce à des espaces de travail isolés
- **Gestion des tickets :**
  - Suivi des bogues, des tâches et des demandes de fonctionnalités
  - Amélioration de la communication/collaboration entre équipes
- **Gestion des versions :**
  - Organisation et suivi des différentes versions du logiciel
  - Facilitation des déploiements et gestion des releases
- **Avantages :**
  - Améliore la transparence et la traçabilité
  - Approche collaborative dans le développement de logiciels

Découpage du travail et affectation des tâches 1/2

- Objectif :
  - Structurer le travail
  - Faciliter la répartition équitable et efficace des tâches
- Stratégies de découpage :
  - Décomposition des fonctionnalités en tâches granulaires
  - Identification des dépendances entre les tâches
- Affectation des tâches :
  - En fonction des compétences et expertises de chacun
  - Prise en compte de la charge de travail actuelle de chaque membre
- Outils et techniques :
  - Utilisation de plateformes de gestion de projet (Jira, **Trello**, Asana)
  - Méthodes Agile et **Scrum** pour une affectation dynamique et itérative

Découpage du travail et affectation des tâches 2/2

- Suivi et ajustements :
  - Réunions régulières pour le suivi de l'avancement
  - Ajustements basés sur le feedback et les changements de priorités
- Avantages :
  - Favorise une réalisation du projet en temps et en heure
  - Permet de tirer pleinement parti des compétences de l'équipe

Choix des environnements de développement

Choix des Environnements de Développement 1/2

- Objectifs :
  - Sélectionner des environnements de développement adaptés
  - Faciliter la réalisation des différentes parties du projet
- Critères de choix :
  - Compatibilité avec les technologies utilisées
  - Performance et stabilité de l'environnement
  - Facilité d'utilisation et de configuration
  - Support communautaire et documentation
  - Coût
- Environnements courants :
  - IDEs : Visual Studio, **PyCharm**, Eclipse
  - Environnements virtuels : **Docker**, Vagrant
  - Gestionnaires de paquets : npm, pip, **conda**

Choix des Environnements de Développement 2/2

- Considérations spécifiques :
  - Environnements pour le frontend et le backend
  - Outils pour la base de données et les services web
  - Solutions pour le développement mobile (ex. React Native, Swift)
- Intégration et automatisation :
  - Intégration continue (CI) et déploiement continu (CD)
  - Outils d'intégration comme Jenkins, GitLab CI
- Avantages :
  - Facilitation du développement et du debugging
  - Création d'un environnement de travail harmonieux et productif

Mise en place des procédures de tests unitaires et de qualification

Tests unitaires et de qualification

- Objectifs :
  - **Tests unitaires** : validation individuelle des composants du logiciel
  - **Tests de qualification** : validation du système dans son ensemble
- Phase de développement :
  - Développement parallèle des tests et du code source
  - Intégration continue et automatisation des tests
- Outils et bonnes pratiques :
  - Utilisation d'outils spécialisés : JUnit, **PyTest**, Selenium
  - Maintien d'une documentation à jour et formation des équipes
- Avantages :
  - Détection précoce des anomalies (erreurs, régressions, ...)
  - Assurance d'un produit final de qualité

Production de documentations

Documentation technique pour la maintenance et l'évolution

- **Objectif :** Fournir une base solide pour la maintenance future et l'évolution de l'application
- **Éléments clés :**
  - Manuel d'utilisation et d'administration
  - Guide d'installation et de configuration
  - Spécifications techniques détaillées
  - Journal des changements (Changelog)
- **Bonnes pratiques :**
  - Maintien régulier de la documentation à jour
  - Utiliser schémas/illustrations pour une meilleure compréhension
- **Outils populaires :**
  - Wiki d'équipe (Confluence, MediaWiki)
  - Générateurs de documentation (Doxygen, **Sphinx**)
- **Avantages :**
  - Facilite la maintenance et le débogage
  - Accélère le processus d'intégration des nouveaux membres

Documentation utilisateur pour la formation des usagers

- **Objectif :** Faciliter l'adoption du logiciel par les utilisateurs
- **Composants essentiels :**
  - Guides d'utilisation détaillés
  - Tutoriels et exemples de cas d'usage
  - FAQ pour résoudre les problèmes communs
- **Bonnes pratiques :**
  - Langage simple et accessible
  - Inclusion de visuels (captures d'écran, vidéos)
  - Mises à jour régulières avec les nouvelles fonctionnalités
- **Moyens de diffusion :**
  - Portail d'aide en ligne
  - Webinaires et ateliers
  - Manuels imprimés et PDF
- **Avantages :**
  - Augmentation de la satisfaction utilisateur
  - Réduction du nombre de tickets d'assistance

Outils pour la génération de documentation 1/2

- **Générateurs à partir du code :** Automatisent la création de la documentation à partir des commentaires dans le code
  - **Sphinx** : Utilisé principalement pour les projets Python, supporte également d'autres langages
  - Javadoc : Spécifique pour Java, génère des API docs à partir des commentaires Java
  - Doxygen : Compatible avec plusieurs langages, dont C++, C, Java et Python
- **Langages pour la documentation :** Facilitent la rédaction de la documentation avec des syntaxes simples
  - **Markdown** : Syntaxe légère et facile à utiliser, largement adopté
  - Asciidoc : Plus riche que le Markdown, permet une personnalisation avancée
  - RestructuredText : Utilisé principalement avec Sphinx, offre des fonctionnalités avancées

Outils pour la génération de documentation 2/2

- Importance :
  - Favorise la maintenance et l'évolution du logiciel
  - Facilite la collaboration et la communication au sein de l'équipe
- Bonnes pratiques :
  - Documenter le code dès le début du projet
  - Mise à jour régulière de la documentation

Mise en production

Mise en prod. des différentes versions de l'application 1/2

- **Objectif :** Déployer des versions stables et mises à jour de l'application de manière fluide et sécurisée
- **Étapes clés :**
  - Planification des releases avec des roadmaps claires
  - Tests approfondis avant le déploiement
  - Validation des fonctionnalités avec les clients
- **Stratégies de déploiement :**
  - Déploiements progressifs (canary releases, blue-green deployments)
  - Déploiements continuels pour des mises à jour transparentes
  - Utilisation de feature toggles pour activer/désactiver des fonctionnalités
  - Utilisation de conteneurs et d'orchestrateurs pour faciliter la mise à jour

Mise en prod. des différentes versions de l'application 2/2

- Gestion des risques :
  - Préparation de plans de rollback en cas d'échec
  - Monitoring constant pour identifier et résoudre rapidement les problèmes
- Outils de déploiement :
  - Systèmes de gestion des configurations (Ansible, Puppet, etc.)
  - Outils d'intégration et de déploiement continu (Jenkins, GitLab CI, etc.)

Mise en production et sécurité

- **Objectif :** Assurer une mise en production sécurisée des différentes versions de l'application
- **Aspects clés :**
  - Gestion sécurisée des mots de passe (utilisation de gestionnaires de mots de passe, hashage, etc.)
  - Sécurisation de la base de données de production (chiffrement, contrôles d'accès, etc.)
  - Protection contre les attaques courantes (SQL injection, Cross-Site Scripting, etc.)
- **Bonnes pratiques :**
  - Tests de pénétration réguliers
  - Formation de l'équipe sur les meilleures pratiques de sécurité
  - Mises à jour régulières et patchs de sécurité
- **Outils et frameworks :**
  - Outils d'analyse statique de code pour identifier les vulnérabilités
  - Frameworks de sécurité intégrés au cycle de développement (DevSecOps)

Focus sur quelques notions

DevOps

- **Définition :** Unifier le développement (Dev) et l'administration des infrastructures et systèmes (Ops).
- Objectifs principaux :
  - Amélioration continue
  - Livraison rapide et fiable de fonctionnalités
  - Collaboration étroite entre les équipes
- Principaux composants :
  - Intégration continue (CI) / Déploiement continu (CD)
  - Automatisation des tests et des déploiements
- Avantages :
  - Réduction des cycles de développement
  - Moins de bugs et de problèmes de déploiement
- Outils populaires :
  - Jenkins, Travis CI pour l'intégration et le déploiement continus
  - Docker pour la containerisation
  - Kubernetes pour l'orchestration de conteneurs

DevSecOps

- **Définition :** Intégrer la sécurité (Sec) dans le DevOps
- **Objectifs principaux :**
  - Intégration de la sécurité dès le début du développement
  - Automatisation des tests de sécurité
  - Réponse rapide aux incidents de sécurité
- **Principaux composants :**
  - Scans de sécurité automatisés
  - Revue de code sécurisé
  - Formation continue en sécurité
- **Avantages :**
  - Réduction des vulnérabilités de sécurité
  - Conformité réglementaire facilitée
  - Collaboration entre les équipes de développement et de sécurité
- **Outils populaires :**
  - Outils d'analyse statique de code (SAST)
  - Outils de détection dynamique des vulnérabilités (DAST)
  - Plateformes de gestion des identités et des accès (IAM)

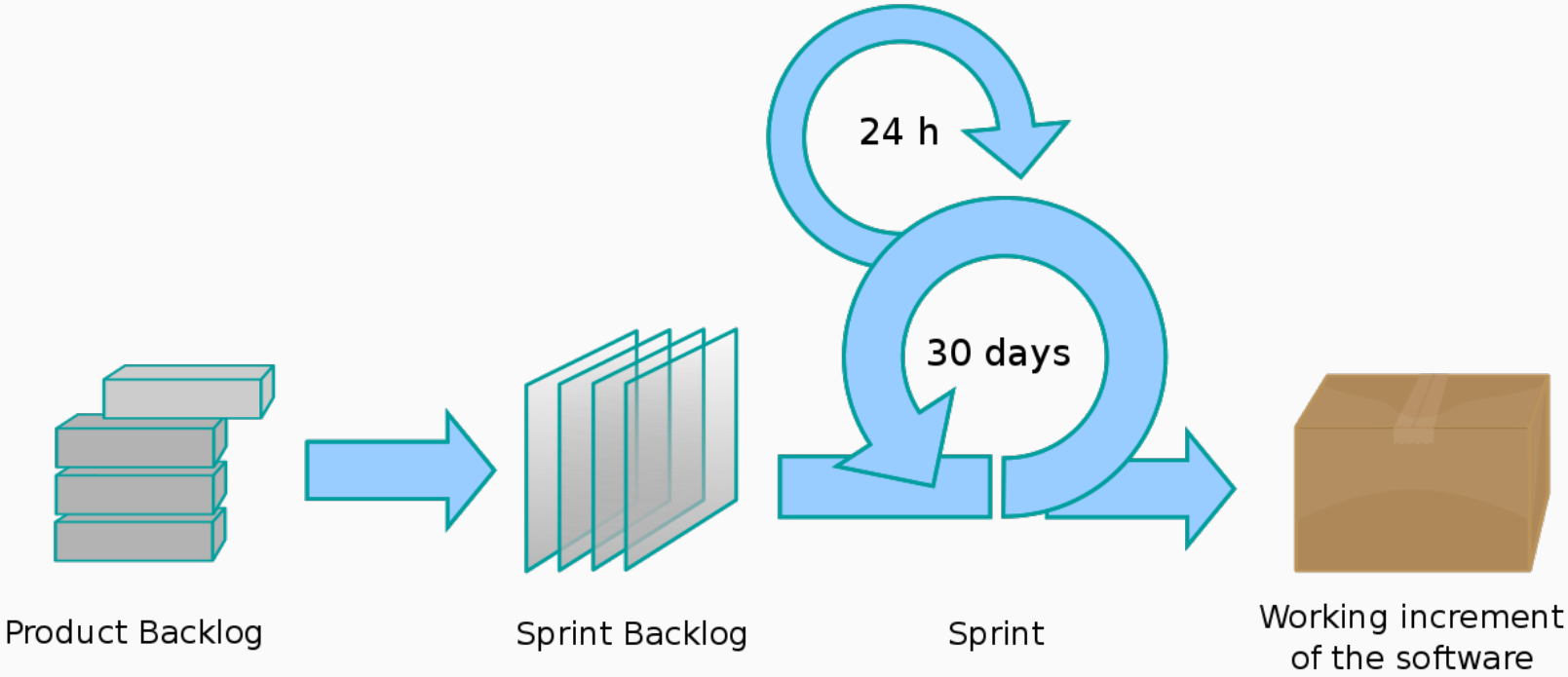
Intégration Continue

- **Définition :** Intégrer automatiquement et continuellement les modifications apportées au code source dans un projet.
- **Principes clés :**
  - Intégration régulière de code source
  - Automatisation des tests
  - Retour rapide aux développeurs
- **Avantages :**
  - Détecter et corriger les bugs plus rapidement
  - Réduction des risques d'intégration
  - Livraison plus rapide de fonctionnalités et de correctifs
- **Outils populaires :**
  - Jenkins, un serveur d'automatisation open source
  - Travis CI, un service d'intégration continue hébergé
  - GitLab CI/CD, un service d'intégration et de déploiement continus
- **Bonnes pratiques :**
  - Maintenir une suite de tests complète
  - Automatiser le build et les déploiements

SCRUM 1/2

- **Définition :** framework agile mettant l'accent sur la collaboration, l'adaptabilité et les retours rapides.
- **Éléments clés :**
  - Product Backlog : Liste classée de fonctionnalités/tâches à faire
  - Sprints : Périodes de travail courtes pour accomplir un ensemble défini de tâches
  - Daily Stand-up : Réunions quotidiennes pour discuter des progrès et des obstacles
- **Rôles :**
  - Product Owner : Représente les clients et les utilisateurs
  - Scrum Master : Facilite le processus SCRUM et s'assure que l'équipe suit les principes et pratiques de SCRUM
  - Équipe de Développement : Réalise le développement et les tests
- **Avantages :**
  - Adaptabilité face aux changements
  - Collaboration étroite entre les membres de l'équipe
  - Livraison continue de valeur pour les utilisateurs

SCRUM 2/2



Lakeworks - `https://commons.wikimedia.org/wiki/File:Scrum_process.svg`

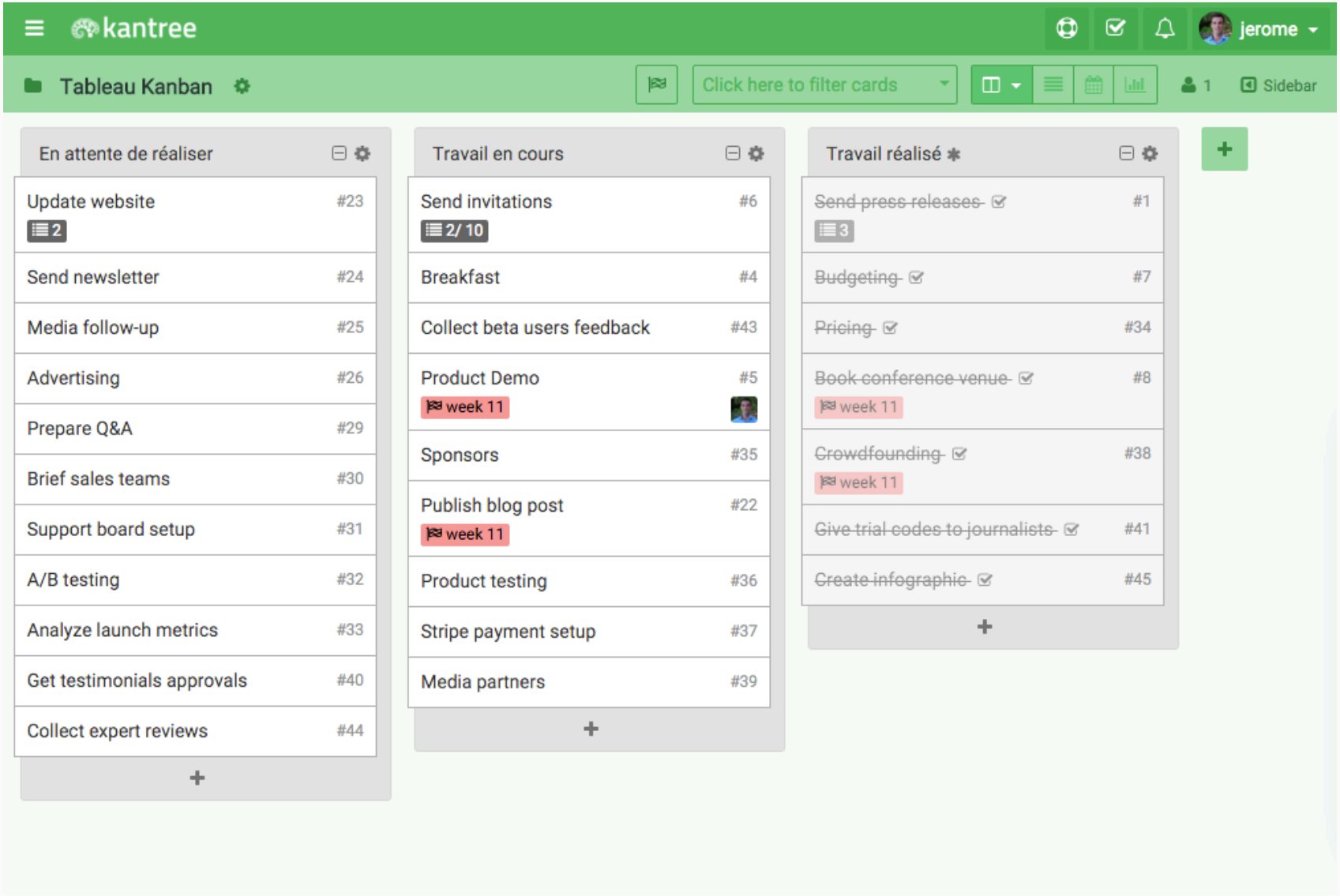
Les daily stand-up dans SCRUM

- **Définition :** Réunion quotidienne et brève (15 minutes)
- **Objectifs :**
  - Favoriser la communication au sein de l'équipe
  - Identifier rapidement les obstacles et les défis
  - Faciliter la résolution proactive des problèmes
  - Synchroniser ses activités et de planifier le travail de la journée
- **Format typique :** Chacun répond à trois questions :
  - Qu'ai-je fait hier ?
  - Que vais-je faire aujourd'hui ?
  - Y a-t-il des obstacles sur mon chemin ?
- **Bonnes pratiques :**
  - Se tenir debout pour encourager la brièveté
  - Commencer à l'heure, quelle que soit l'assistance
  - Se concentrer sur les objectifs du sprint en cours
- **Avantages :**
  - Favorise la transparence et l'ouverture
  - Crée un rythme régulier et prévisible pour l'équipe
  - Aide à bâtir une culture d'équipe collaborative et autonome

Les tableaux Kanban 1/2

- **Définition :** Outil de gestion visuelle du workflow
- **Caractéristiques clés :**
  - Visualisation du workflow : Répartition des tâches dans différentes colonnes représentant les étapes du processus.
  - Limitation du travail en cours (WIP) : Définition d'un nombre maximum de tâches qui peuvent être traitées à chaque étape.
  - Gestion des blocages : Identification et résolution rapide des obstacles empêchant la progression des tâches.
- **Structure typique :**
  - À faire : Liste des tâches à accomplir.
  - En cours : Tâches actuellement en traitement.
  - Fait : Tâches complétées avec succès.
- **Avantages :**
  - Améliore la visibilité du workflow.
  - Identifie les goulets d'étranglement.
  - Permet une adaptation rapide aux changements.

Les tableaux Kanban 2/2



Kantree - `https://commons.wikimedia.org/wiki/File:Tableau_kanban.png`