Ticketing avec Gitlab

1. Qu'est-ce qu'un ticket ?

Lors du développement ou de la maintenance d'une application, il est généralement utile d'avoir un moyen de communication entre les testeurs ou utilisateur de l'application et les développeurs, par exemple pour permettre la remontée de bugs ou des propositions d'améliorations de l'application, des besoins ou problèmes rencontrés, etc.

On utilise généralement des outils de ticketing ou outils de remontée des bugs pour établir cette communication.
Ces outils peuvent être intégrés à l'environnement de développement (*c'est le cas avec Gitlab ou Github*) ou peuvent être externes (*Redmine, Bugzilla, Mantis, Jira, …*)

Le processus suivi est assez similaire au processus de gestion des incidents vu en cours :

a. Un utilisateur ou un testeur remonte une erreur ou une demande d'amélioration
b. Un développeur de l'équipe effectue le support de niveau 1 :
   - Si c'est un bug, il le qualifie comme bug en lui affectant un label/une étiquette, et l'affecte à un membre de l'équipe spécialisé sur le code affecté par le bug (*s'il y en a un*)
   - Si c'est une demande d'amélioration, il la qualifie comme telle en lui affectant un label

   Il existe différentes façons de s'organiser. Parfois l'affectation des tickets sera faite directement au niveau 1, parfois cette affectation se fera en début de semaine avec toute l'équipe, …

   Dans tous les cas, le ticket sera ensuite affecté à une **milestone**, ou étape. Les milestones sont généralement les versions de l'application. Cette affectation permet à l'utilisateur de savoir quand le problème sera corrigé, et quelle mise à jour il devra appliquer pour disposer du correctif ou de la nouvelle fonctionnalité.

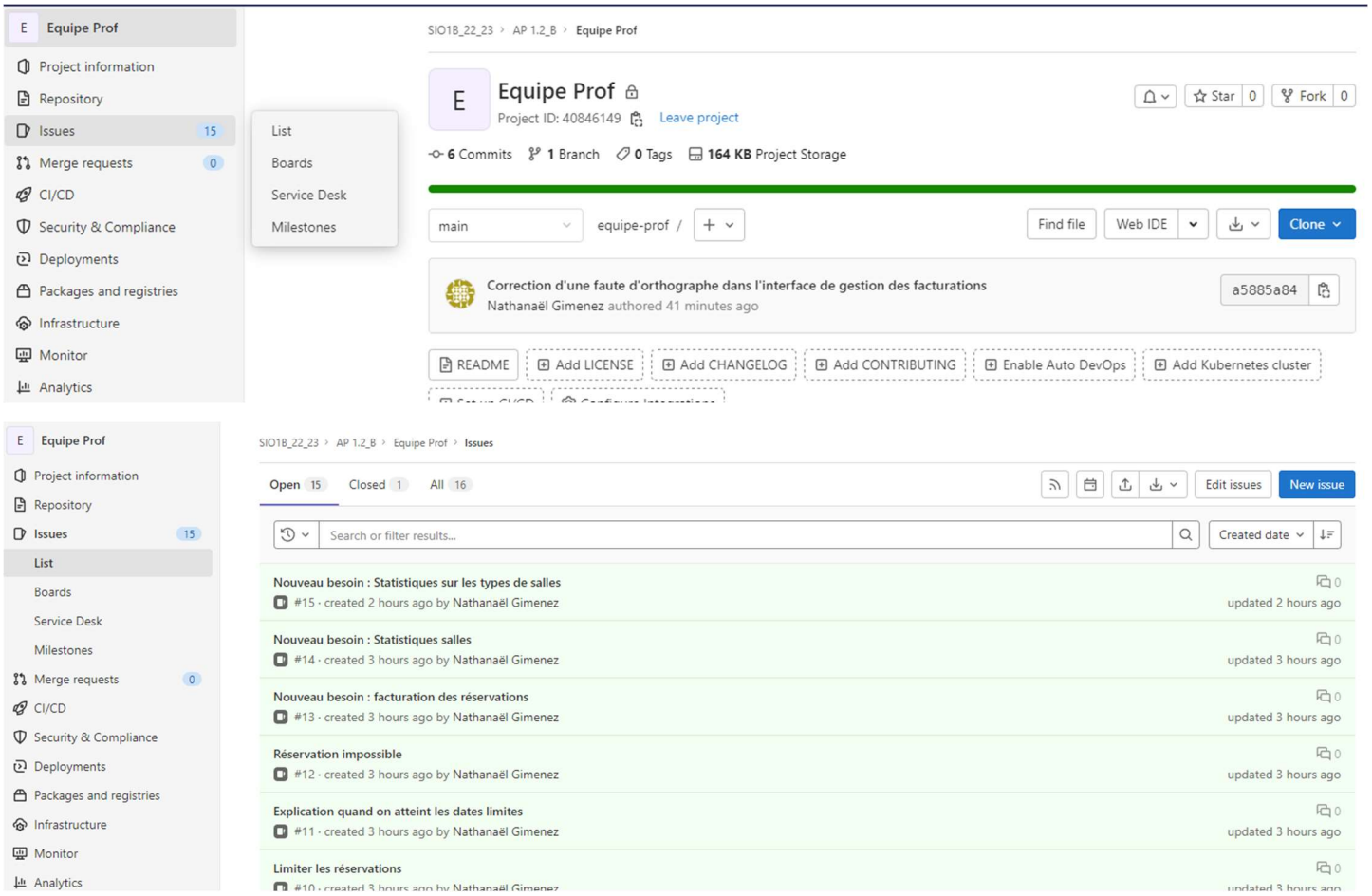
   Durant le support de niveau 1, la personne réalisant ce support devra communiquer au moins brièvement avec la personne ayant créé le ticket pour la prévenir que la demande est prise en compte et ce qui va être réalisé.

c. La développeuse affectée s'empare du ticket : c'est maintenant elle qui en est responsable. Elle va travailler sur le code pour corriger le bug ou réaliser la demande d'amélioration. Il sera peut-être nécessaire de communiquer avec le demandeur pendant ce processus : avoir plus d'information sur comment le bug s'est produit, avoir des détails sur la fonctionnalité demandée, valider des interfaces, etc. Ces échanges se passent directement dans l'outil de ticketing. Ils doivent être professionnels, précis, compréhensibles, avec une bonne grammaire et une bonne orthographe.
d. Une fois que le ticket est résolu, selon la méthode adoptée, il pourra être passé par des étape d'attente de validation puis validation, ou étape de test, etc.
Chaque équipe à sa propre méthode.
e. Une fois que le ticket est considéré comme totalement résolu par la méthode de gestion de projet adoptée par l'équipe, le ticket est clôturé.

2. Visualiser des tickets avec Gitlab

Dans Gitlab, les tickets sont appelés **Issues**. Pour les visualiser, il faut :

a. Aller dans le projet
b. Cliquer sur Issues dans le menu de gauche



Cette interface permet de visualiser les tickets ouverts (*Open*), ou tickets en cours, et les tickets fermés (*Closed*).

Vous avez un exemple de ticket fermé dans l'onglet Closed, avec toute sa gestion.

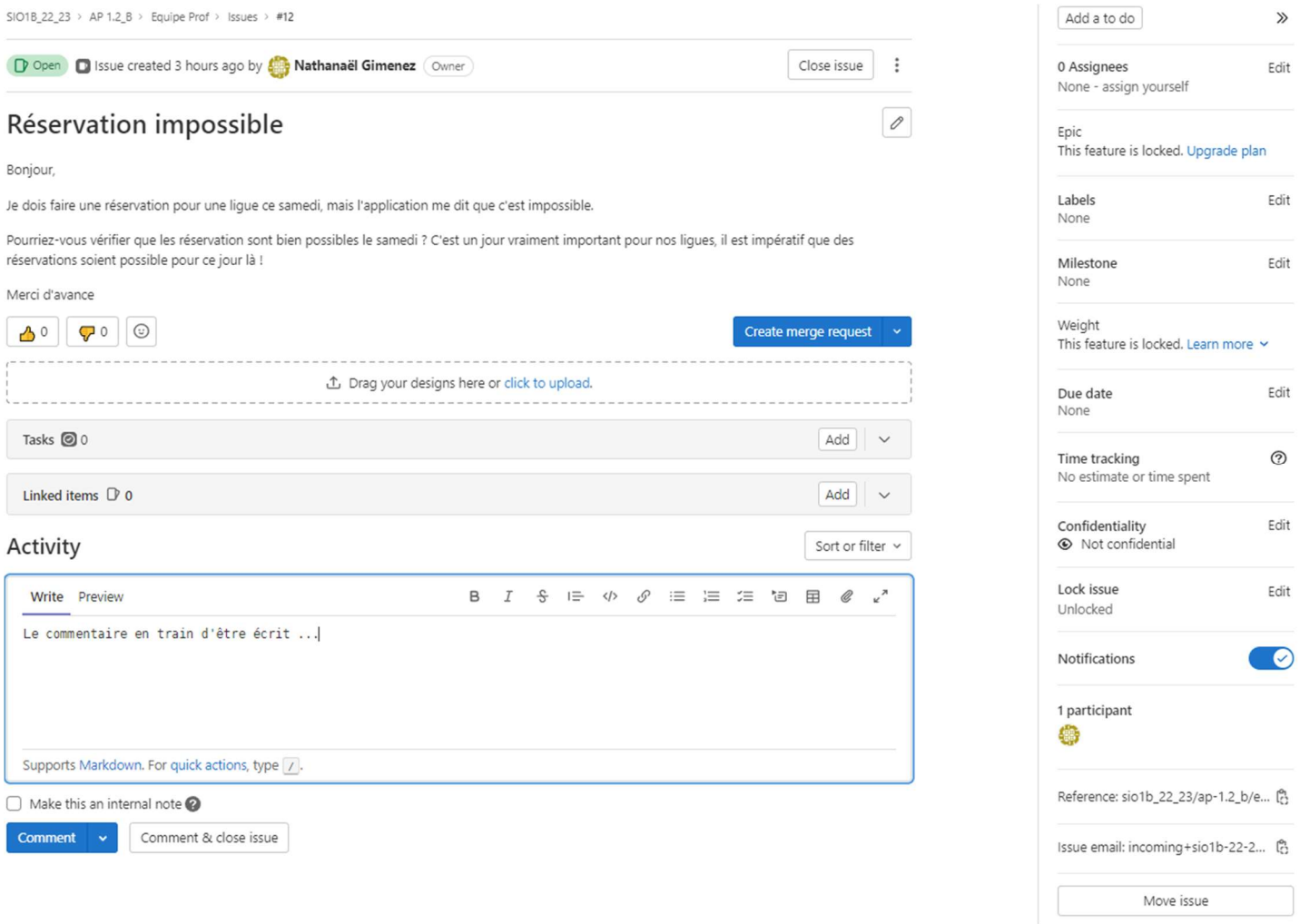
3. Gérer des tickets avec Gitlab

Créer un nouveau ticket :
Il faut cliquer sur le bouton bleu ***New Issue*** puis remplir les champs adéquats.

Si c'est un développeur qui crée le nouveau ticket, il peut directement préciser des informations comme le label, la milestone, ou le développeur affecté.

Gérer un ticket :

a. Cliquez sur le ticket qu'on souhaite gérer. La fenêtre suivante s'affiche :



b. Le menu de droite permet de :

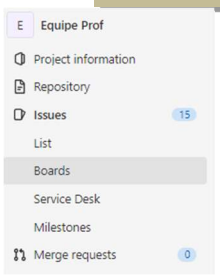
- Affecter un développeur à une tâche : *Assignees > Edit* puis choisir la personne à affecter
- Associer un label au ticket : *Labels > Edit* puis sélectionner un ou plusieurs labels. Pour supprimer un label, passer sa souris dessus puis cliquer sur la croix qui s'affiche.
- Affecter le ticket à une milestone : *Milestone > Edit* puis choisir la bonne milestone
- Affecter une deadline à un ticket : *Due Date > Edit*
- Choisir de recevoir ou non les notifications relatives au ticket : *Notification* puis utiliser le bouton slider.
- Communiquer avec le client : écrire dans la textbox de commentaires puis cliquer sur le bouton bleu *Comment*
- Fermer un ticket : ecrire un commentaire puis utiliser le bouton *Comment and close issue* **ou** en haut de la page du ticket, cliquez sur le bouton *Close issue*

Créer des tâches :

Parfois, la résolution d'un ticket peut demander un temps de travail assez long, avec plusieurs tâches différentes à réaliser. Il faut parfois même affecter certaines de ces tâches à différentes développeuses.

Gitlab propose dans ce cas là d'utiliser la section ***Tasks*** dans l'interface d'un ticket. On peut créer plusieurs tâches associées à un ticket, puis gérer chaque tâche comme si elle était un ticket à part entière. La liste des tâches associées apparaissant dans le ticket, ça donne une meilleure visibilité de l'avancée du travail pour les personnes travaillant à la résolution du ticket.

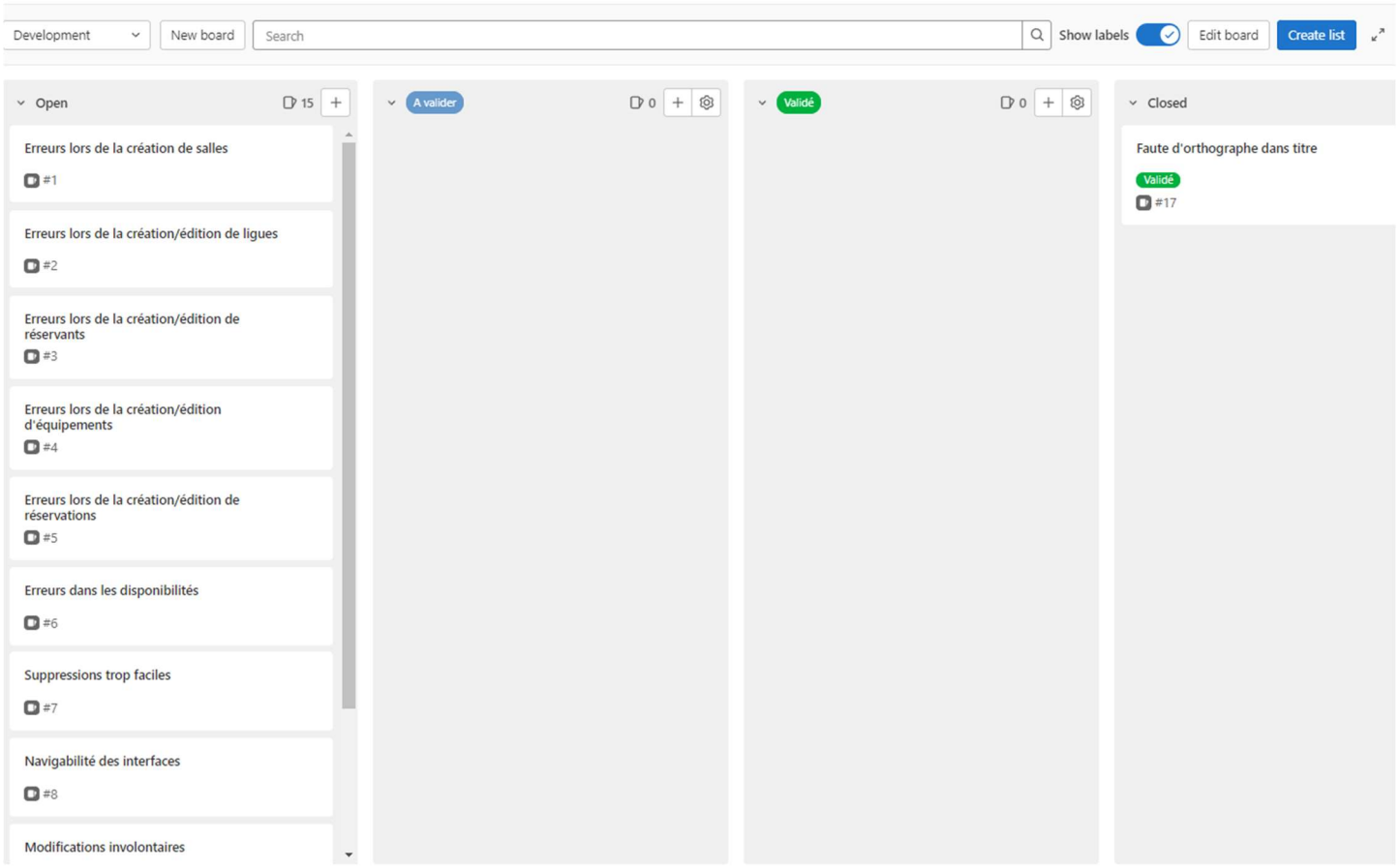
4. Gérer son travail avec Gitlab



Gitlab propose une interface de gestion des tâches assez similaire à Trello. Pour y accéder, il faut survoler l'option *Issue* dans le menu de gauche, puis sélectionner *Boards*

Les tickets et les tâches du projet apparaissent automatiquement dans les Boards sous forme de « post-it ».

L'intérêt de cet outil est son automatisation : on peut associer un label de ticket à une des colonnes (*appelée **List** dans Gitlab*), et le changement de label d'un ticket le fait automatiquement passer d'une colonne à une autre. L'ouverture d'un ticket le place automatiquement dans la colonne *Open*, sa fermeture le passe automatiquement dans la colonne *Closed*.

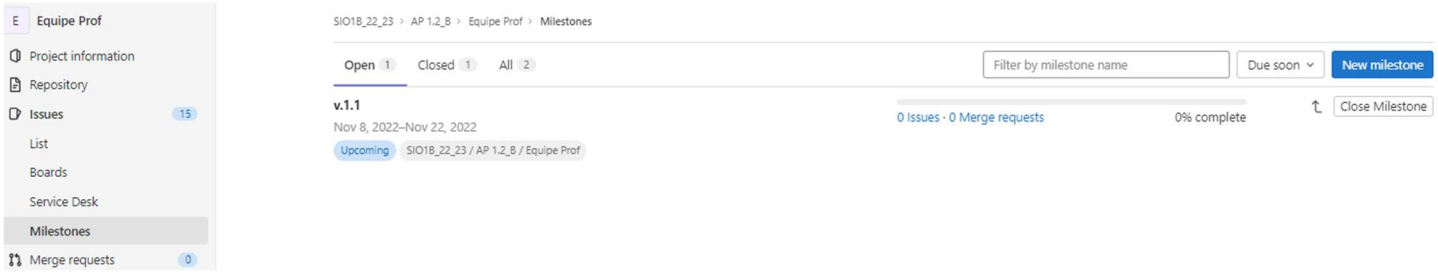


Si on utilise les tâches associées au ticket, elles apparaissent aussi dans les Boards !

Cet outil permet d'avoir une bonne visibilité du travail en cours. Le ou les développeurs affectés à un ticket ou une tâche apparaissent aussi dans sa version « post-it » sur le Board, ce qui simplifie grandement la gestion de projet.

5. Gérer des milestones avec Gitlab

Pour gérer des milestones avec Gitlab, il suffit d'aller dans le menu de gauche, *Issues > Milestones*



Cette interface permet de voir les milestones/étapes existantes/en cours (*Open*), et celles déjà accomplies (*Closed*)

Quand on sélectionne une milestone, on peut visualiser tous les tickets affectés à cette milestone. On peut aussi définir la date à laquelle la milestone doit commencer, et la date à laquelle elle doit se finir/être livrée.

6. Pourquoi utiliser ces outils ?

Parce que ces outils vont nous permettre d'avoir une gestion plus professionnelle de notre code :

- Avec le ticketing, nous pouvons communiquer avec le client et les utilisateurs. Les développeurs sont prévenus au plus vite des problèmes sur l'application, le client peut plus facilement exprimer ses besoins et les problèmes rencontrés, l'application devient donc de plus en plus fonctionnelle et adaptée aux besoins du client

- Avec l'outil de Boards, un chef de projet ou un client (*selon les paramètres de visibilité affectés*) peut voir l'avancée du travail d'un simple coup d'œil. La gestion est très automatisée et permet d'éviter de perdre du temps à recopier des tâches sur Trello à la main, puis de mettre à jour leur statut à la main (en les déplaçant de colonne en colonne)

- Dans la version payante de Gitlab, on peut suivre le temps passé sur chaque tâche ou ticket, on peut affecter un poids (*complexité, temps estimé, …*) à chaque tâche ou ticket et suivre le temps vraiment passé. Ces notions sont essentielles à la gestion de projet !

- Avec l'outil de Milestone, le client et les utilisateurs peuvent voir ce qui est prévu pour les versions suivantes de l'application. Ils peuvent prévoir en avance les modifications qu'il faudra faire sur leur SI pour accueillir la nouvelle version de l'application, ou savoir quand leur problème sera résolu.

- Gitlab propose beaucoup d'autres outils permettant d'augmenter la qualité du code (*tests automatisés, tests de sécurité, etc.*) ou de simplifier la vie des développeur ou du client : par exemple, Gitlab peut déployer automatiquement la nouvelle version d'un projet vers l'environnement du client ou vers un environnement de recette (*tests utilisateurs côté client*), ou alors proposer un lien de téléchargement de l'application au client, etc.

  Ces fonctionnalités ne sont pas exclusives à Gitlab : tous les outils moderne de développement proposent maintenant ces fonctionnalités, souvent sous le nom de fonctionnalités **DevOps**. Elles sont parfois entièrement intégrées (*Gitlab, Github, outils Amazon Web Service ou Google Cloud, …*), ou parfois partiellement intégrées (*Jira, etc.*) Dans d'autres cas, l'organisation qui développe le projet déploiera ses propres outils : Jenkins, Redmine, Mantis, …