Documentation Interne Veremes Documentation

Release latest

Jun 13, 2019

Contents

CHAPTER 1

GitLab

1.1 Notions GitLab

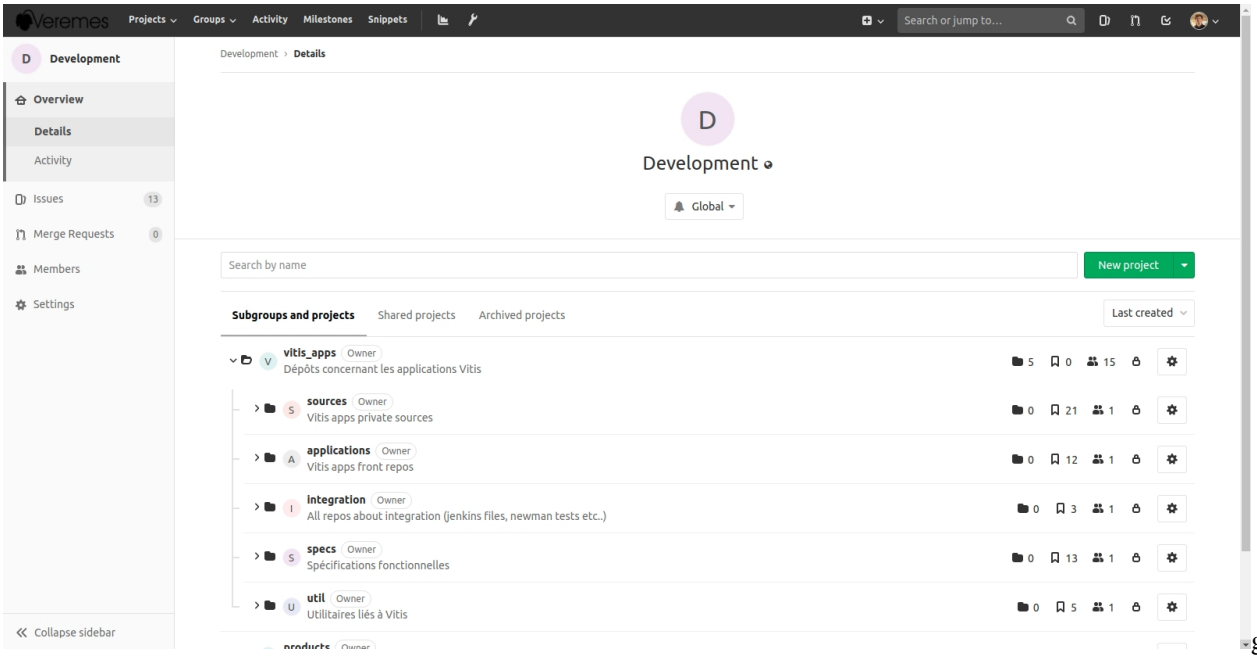
1.1.1 Objets GitLab

1. Groupes

Sur GitLab les projets peuvent êtres répartis par groupes, ces derniers vont agir comme des dossiers conteneurs de dépôts, on peut également placer des sous-groupes qui contiendront eux aussi des dépôts.

1.1. Interface

On peut accéder à la liste des groupes en cliquant sur le bouton **Groups** présent dans la barre suppérieure. On y distingue **Your groups** pour retourner les groupes sur lesquels l'utilisateur contribue et **Explore groups** pour visualiser les groupes de l'utilisateur plus les groupes publics.



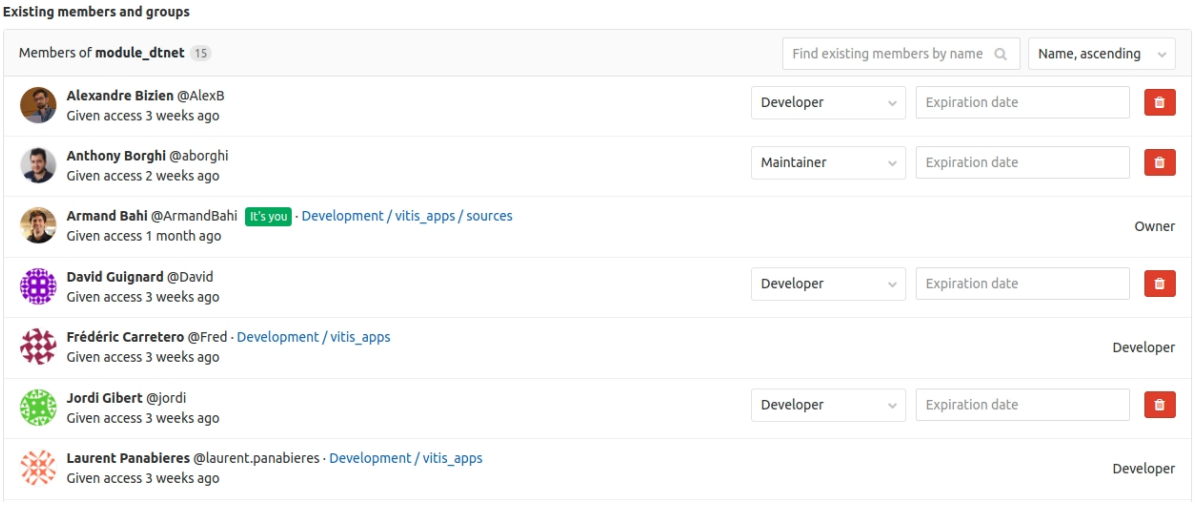
groups

1.2. Droits

On peut dans un groupe gérer les droits des utilisateurs, alors ces droits seront appliqués sur tous les projets contenus.

Une fois dans le dépôt on distinguera les droits donnés au travers du groupe symbolisés par un lien de ceux donnés par le dépôt où on retrouvera une liste déroulante.

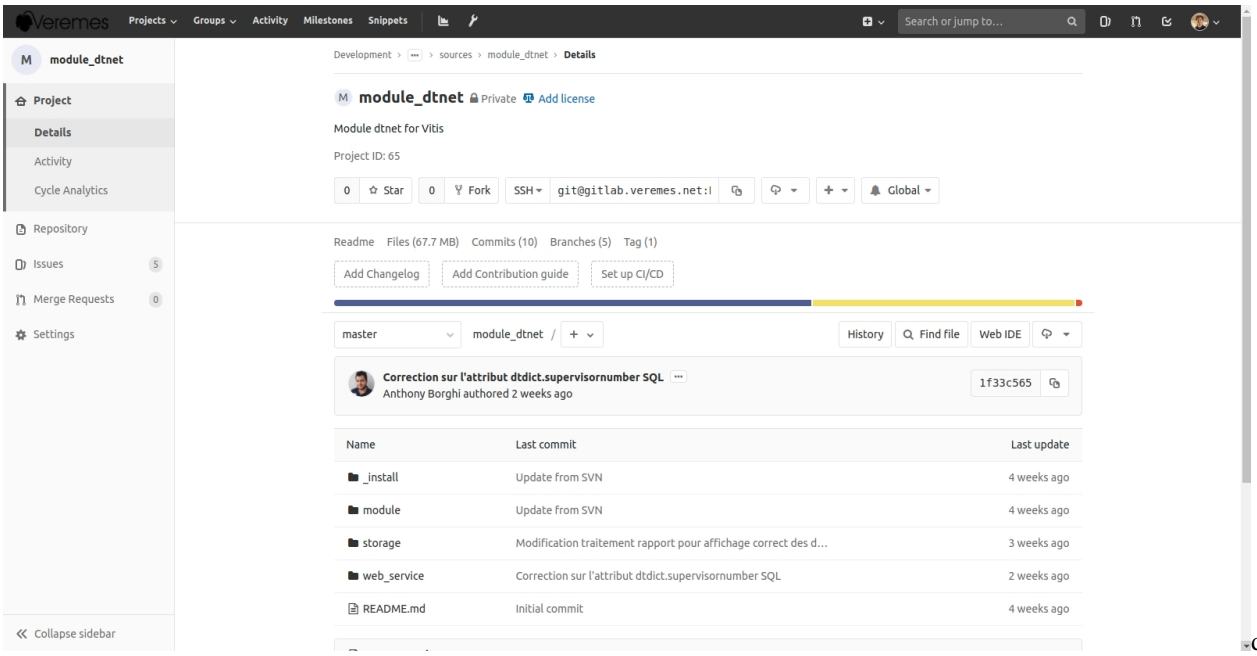
Vous trouverez plus de détails sur les différents droits dans la section correspondante de ce document.



droits

2. Dépôts

Les dépôts agissent comme des conteneurs de fichiers et permettent le versionnement, chaque dépôt est représenté par un lien unique proposé en HTTPS et SSH qui permettra d'effectuer un clone local du projet.



depot

2.1. Branches

Chaque dépôt est contitué au moins d'une branche **Master**, cette dernière est régulièrement protégée pour éviter les erreurs de rétro-compatibilité.

Pour effectuer les modifications il vaut mieux créer une branche pour y effectuer les modifications, puis créer une **merge request**.

2.2. Merge request

Une merge request est une demande d'intégration de modifications d'une branche enfant vers une branche parent, après avoir effectué des modifications sur une branche, il faudra créer une **merge request** et l'affecter à la personne en charge du projet.

De cette manière la personne en charge du projet pourra vérifier et accepter (ou pas) la demande.

2.3. Issue

Une issue est une tâche liée à un problème ou une évolution, l'auteur de l'issue pourra dialoguer avec le responsable de produit et suivre l'avancement de la tâche.

Quand le responsable de projet décide d'effectuer une tache il pourra lier l'issue à une merge request elle même liée à une branche, de cette manière on pourra une fois la tâche terminée vérifier les modifications effectuées.

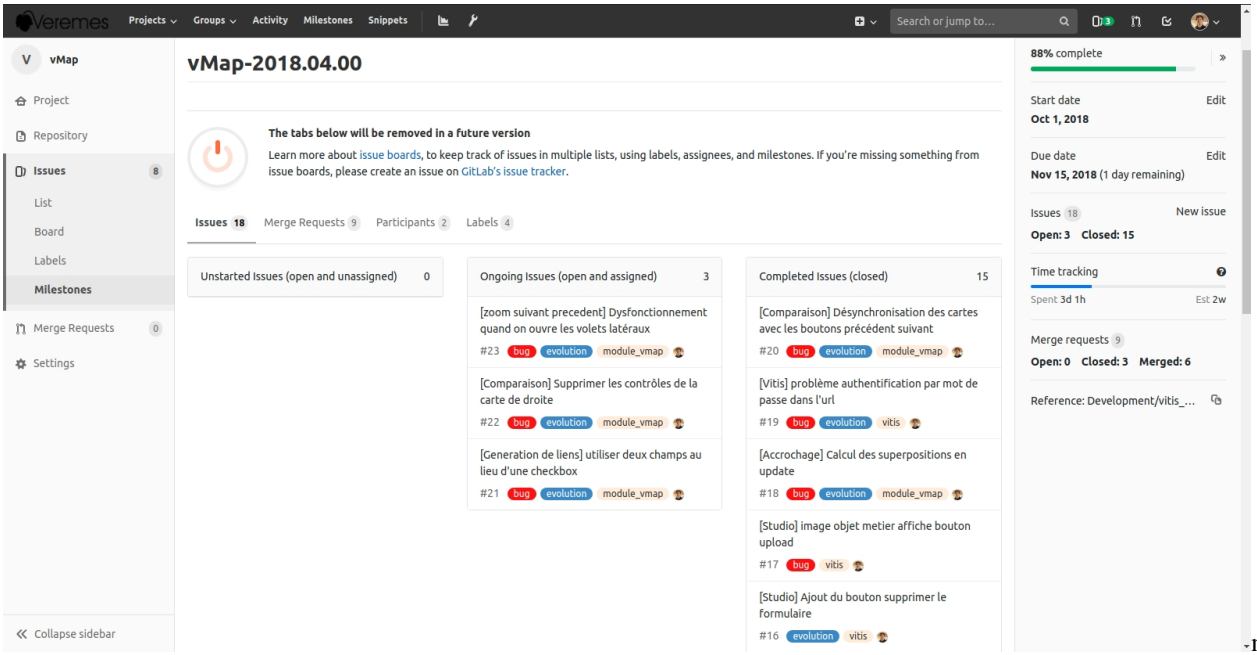
2.4. Labels

Les labels agiront comme des tags pour classifier les issues (bug, évolution etc..) on peut définir les labels à utiliser dans le dépôt ainsi que dans les groupes.

labels

2.5. Milestones

Un milestone est un ensemble d'issues à effectuer pour une version/projet, il permet entre autres de suivre l'avancement du projet.



milestone

3. Droits

Sur GitLab il y a 5 types de droits qui peuvent entre autres effectuer les opérations ci-dessous, une documentation complète est disponible sur https://docs.gitlab.com/ee/user/permissions.html

3.1. Guest

- Visualiser le projet
- Créer des issues
- Voir les issues y compris confidentielles
- Laisser des commentaires

3.2. Reporter

- Pull le projet
- Assigner des issues
- Donner les labels aux issues
- Administrer les labels
- Voir les merge requests
- Manager les issues

3.3. Developer

- Créer des branches
- Push sur les branches non protégées
- Créer/éditer les milestones
- Accepter les merge requests

3.4. Maintainer

- Ajouter des nouveaux membres
- Accéder aux paramètres du dépôt
- Rendre les branches protégées ou non

3.5. Owner

- Changer le niveau de visibilité
- Supprimer/renommer le projet
- Supprimer les issues

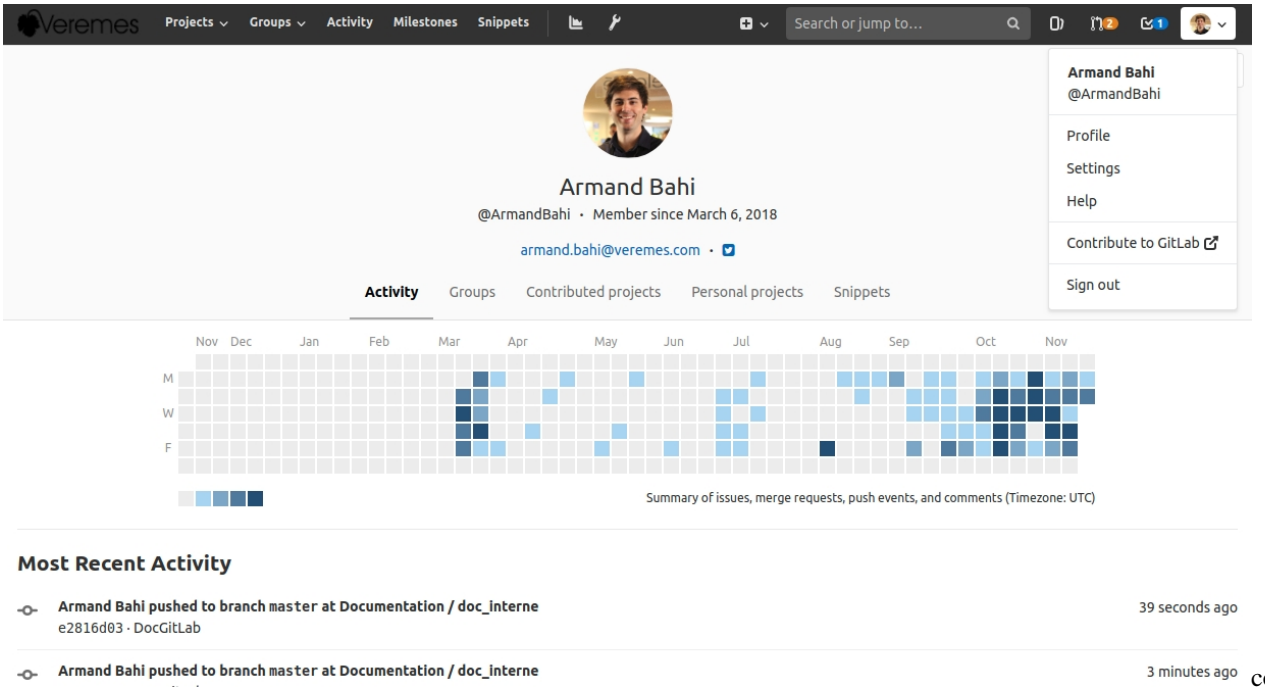
1.1.2 Compte GitLab

Pour utiliser GitLab chaque personne pourra se créer un compte qui aura différents droits et différents paramètres.

Chaque compte est unique et authendifié par l'identifiant du compte correspondant à une chaîne de caractères sans espaces (pour moi c'est *ArmandBahi*) que l'on voit affiché sous notre nom, après le @; tout compte possède également une adresse mail unique pour s'identifier.

Il est très important que les données que vous écrirez sur ces deux champs correspondent à celles que vous allez utiliser localement avec GitKraken etc..

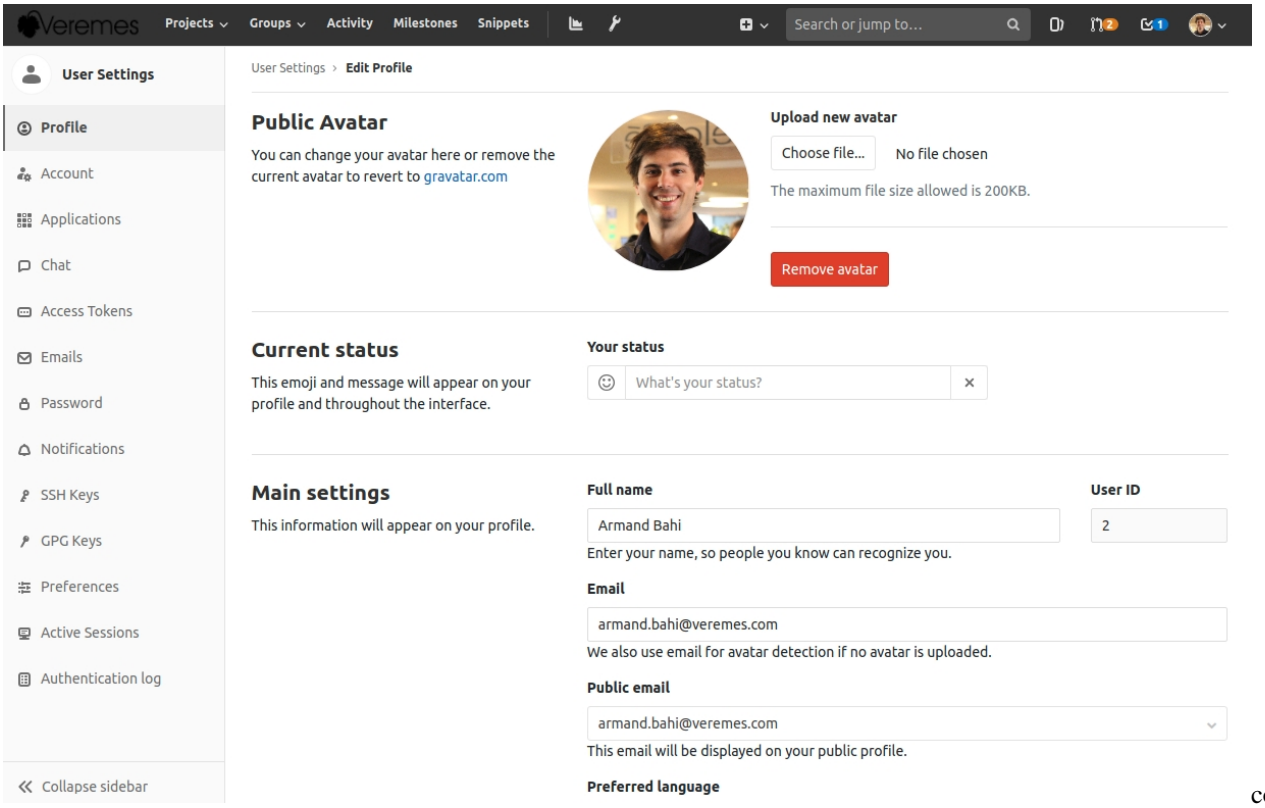
Je vais ci-après vous détailler les sections les plus importantes, pour éditer ces paramètres, il suffit de cliquer sur son profil (bouton en haut à droite) puis aller sur **Settings**



compte_1

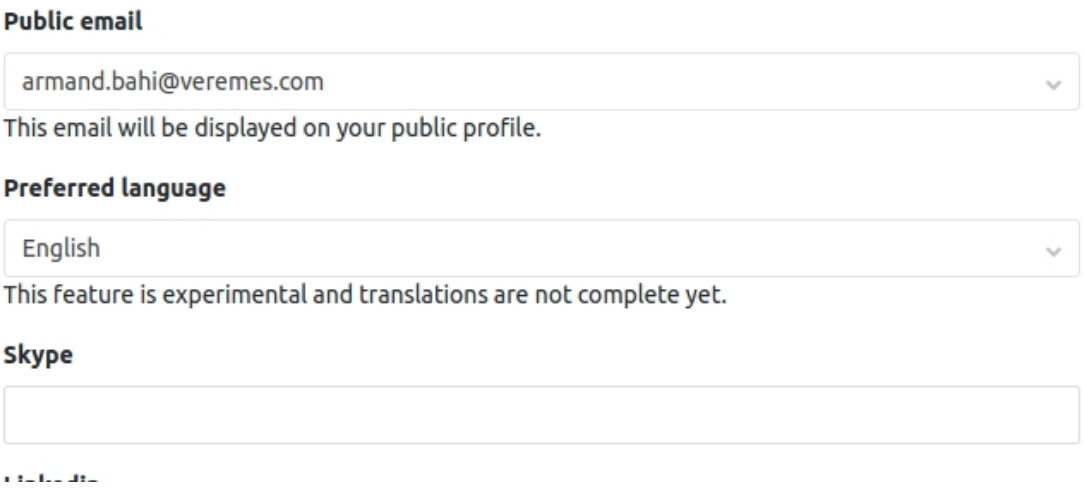
1. Profile

Dans la partie profile nous pouvons éditer notre adresse mail, notre photo etc. . .



compte_2

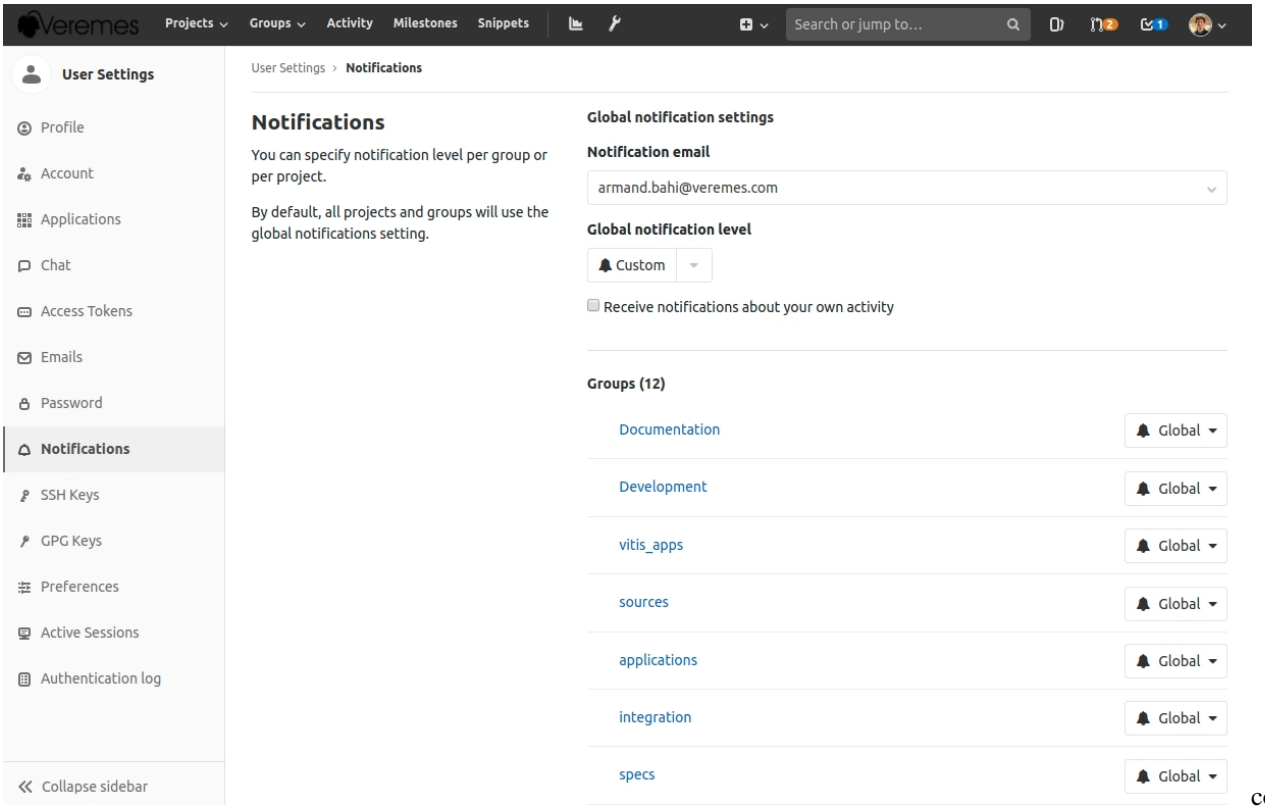
GitLab n'est pas entièrement traduit en Français et parfois on peut se perdre avec les termes techniques, mon conseil si vous arrivez à lire l'anglais technique est de passer l'interface en anglais.



compte_3

2. Notifications

GitLab va vous envoyer des notifications par email en fonction de votre implication dans les différents projets, vous pouvez paramétrer le niveau de notifications que nous allez recevoir directement dans cette section.



compte_4

3. SSH keys

Les clés SSH font partie intégrante de l'utilisation de Git en général car contrairement à SVN, git ne permet pas d'enregistrer ses mots de passe pour des raisons de sécurité. Si vous l'utilisez tel quel git vous demandera votre mot de passe à chaque fois que vous voudrez envoyer un fichier sur le serveur.

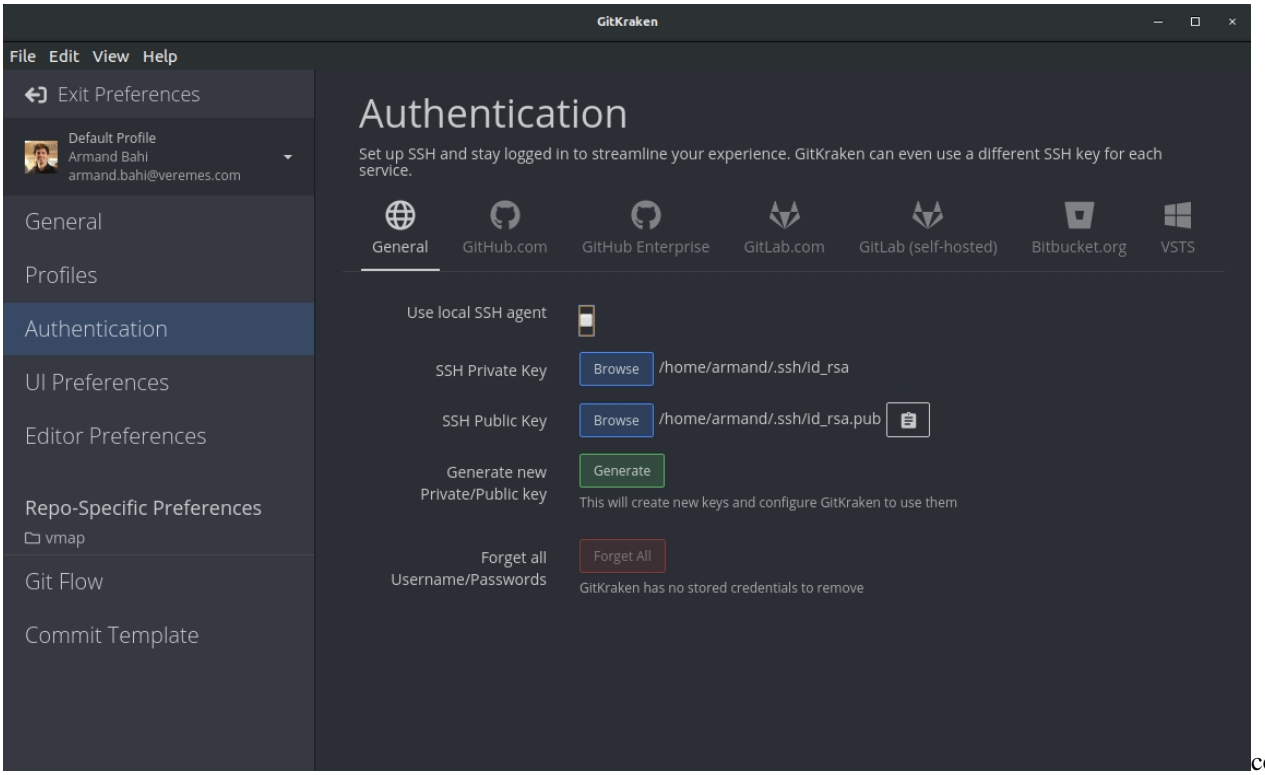
Pour ne pas devoir taper votre mot de passe à chaque fois, vous devrez créer une clé SSH et la lier à votre compte GitLab.

3.1. Générer une clé SSH

Pour générer une clé SSH il y a différentes solutions, l'une d'entre elles et la plus simple celons moi est d'utiliser GitKraken.

Pour cela il faut cliquer sur le bouton en situé à droite de votre profil, puis *Preferences>Authentification*, alors GitKraken permet de générer une clé SSH qu'il utilisera.

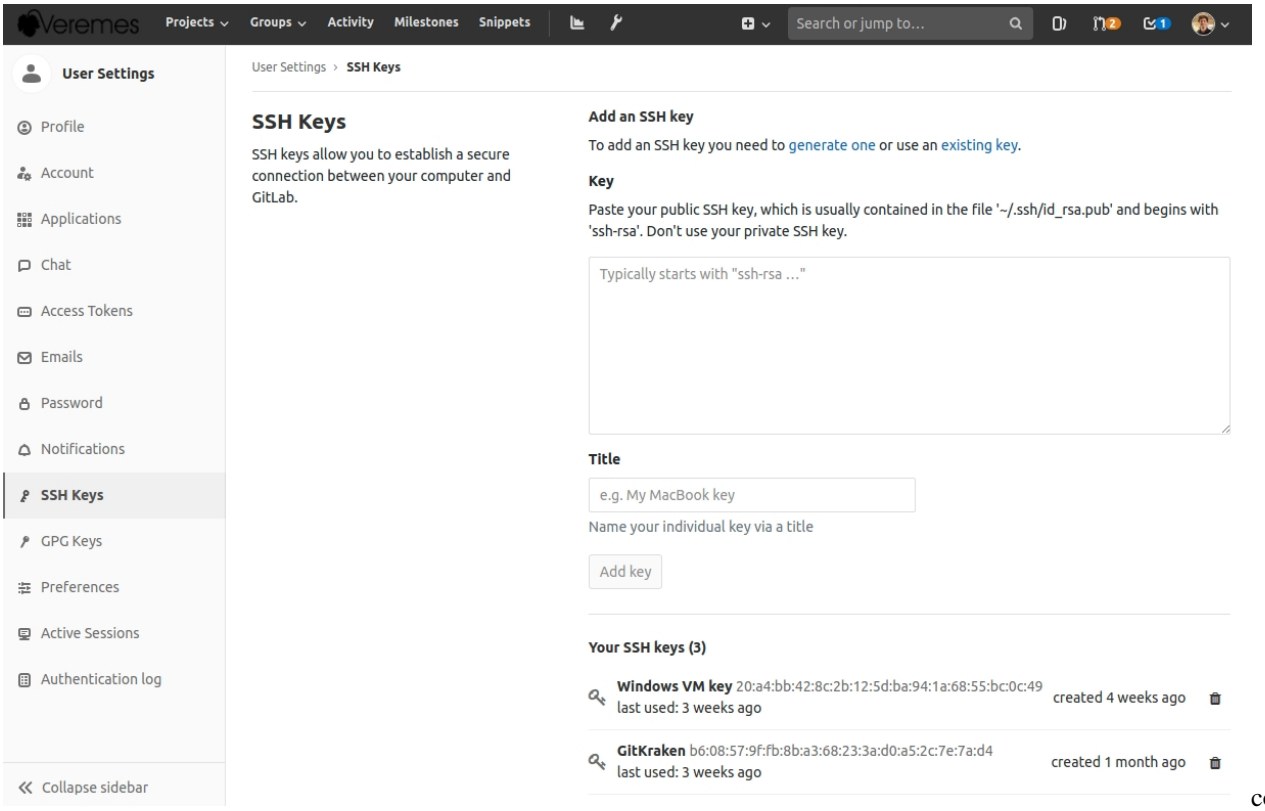
Une fois ceci fait deux fichiers auront étés générés id_rsa qui correspond à la clé privée **que vous ne devez communiquer sous aucun prétexte** et id_rsa.pub qui est la clé publique que vous fournirez aux différents services.



compte_ssh_gitkra

3.2. Enregistrer la clé

Une fois que vous avez généré votre paire de clés, copiez le contenu de la clé publique (id_rsa.pub) et collez-le dans le champ Key, enfin donnez un nom à votre clé (exemple : Ordinateur Armand) puis ajoutez-la.

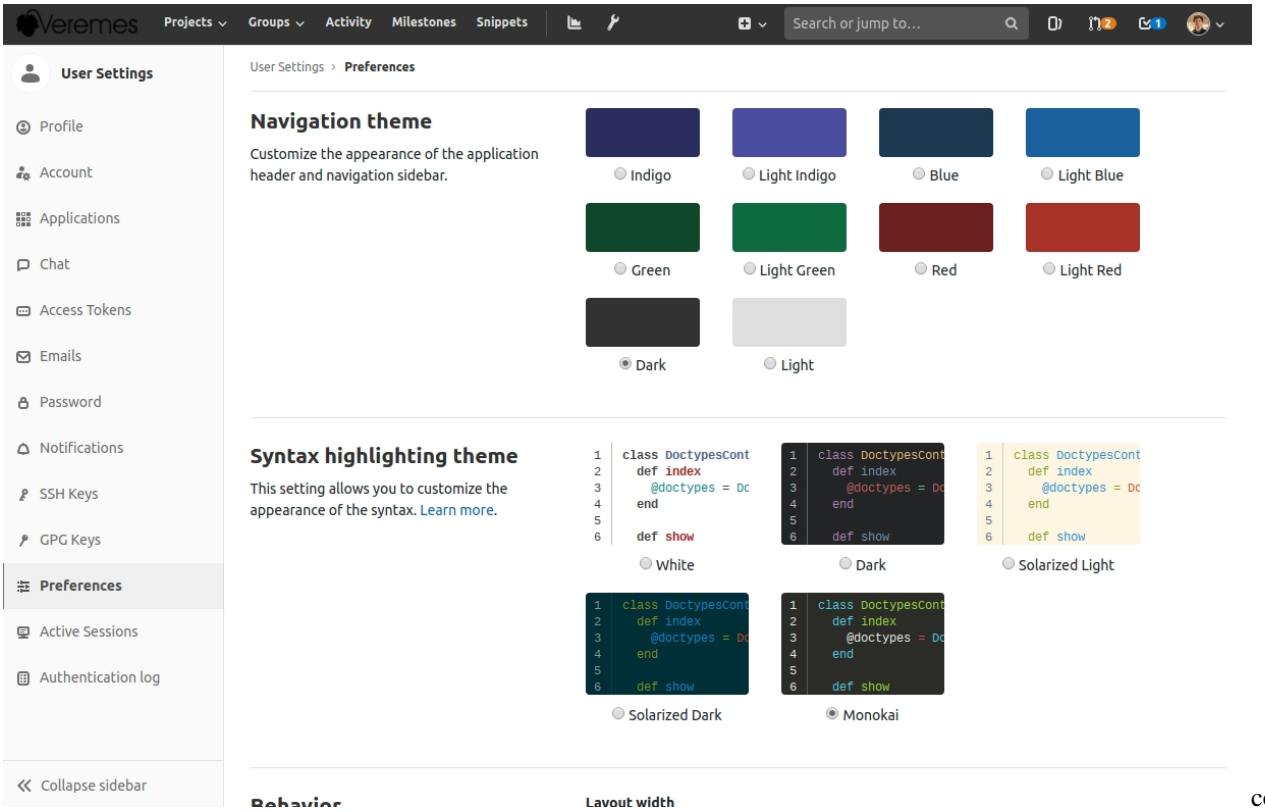


compte_5

Vous pouvez maintenant utiliser GitKraken sans taper votre mot de passe à nouveau.

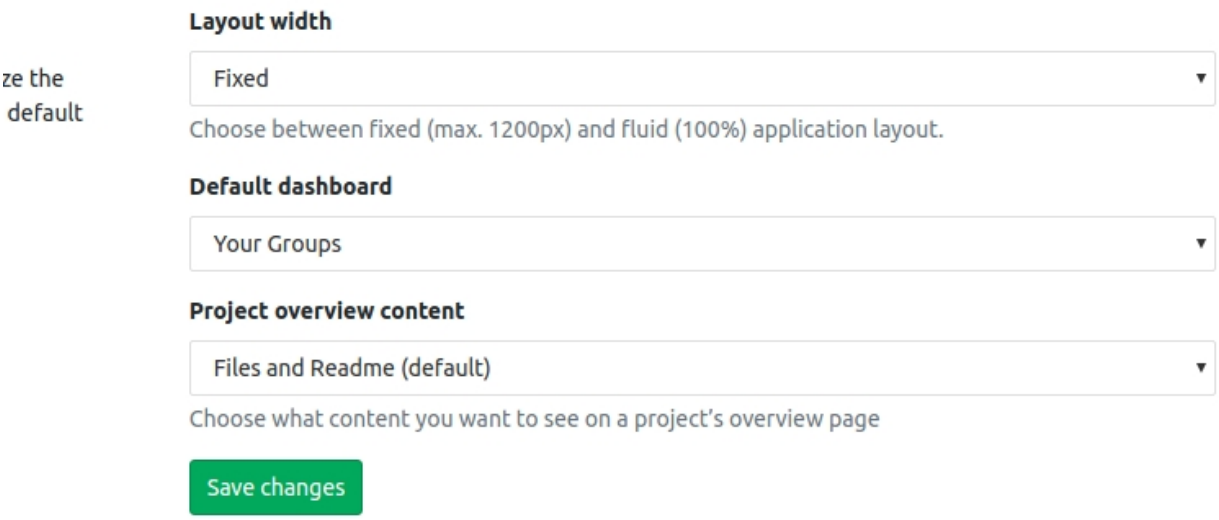
4. Preferences

Ici vous pourrez modifier la couleur de votre interface et surtout décider de quelle page s'affiche quand vous arrivez sur GitLab.



compte_6

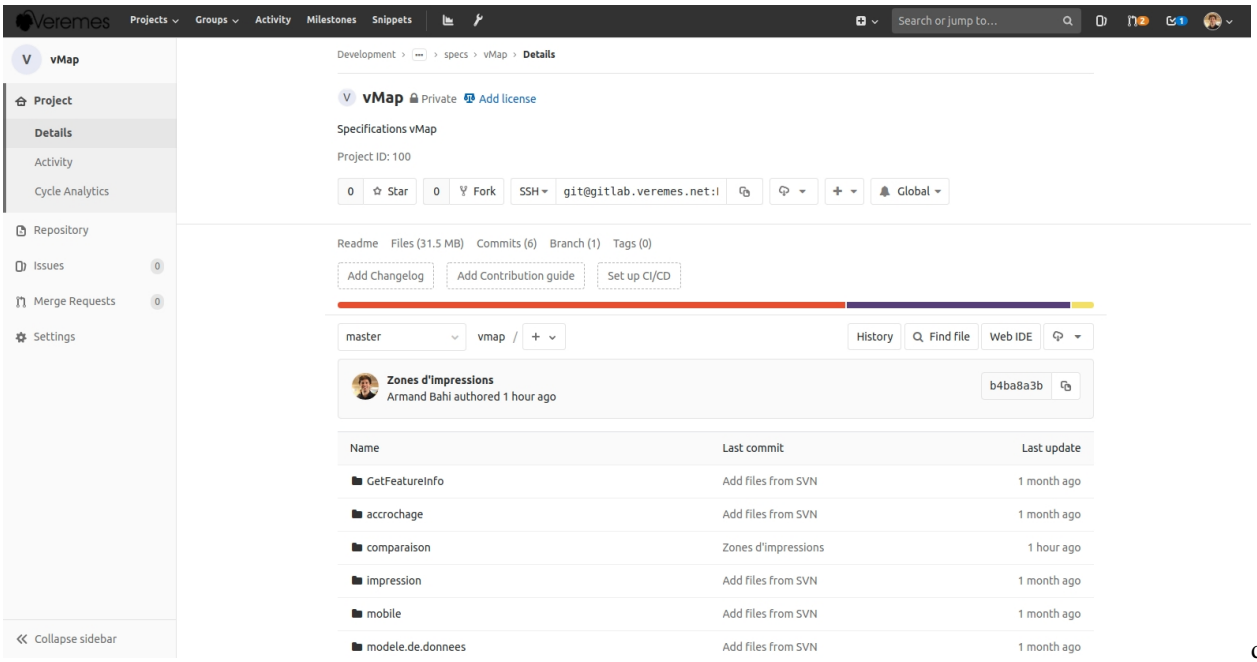
J'ai moi même décidé d'afficher **Your Groups** en page d'accueil de manière à arriver sur la liste des groupes dans lesquels je suis impliqué.



compte_7

1.1.3 Dépôt GitLab

Les dépôts agissent comme des conteneurs de fichiers et permettent le versionnement, chaque dépôt est représenté par un lien unique proposé en HTTPS et SSH qui permettra d'effectuer un clone local du projet.



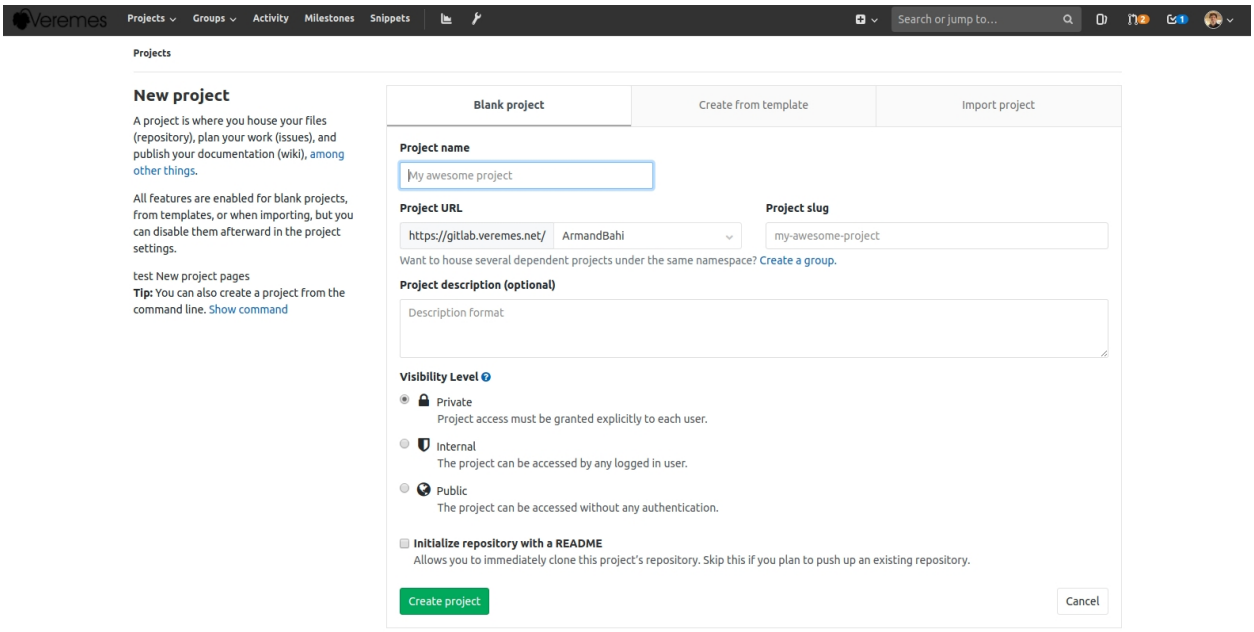
depot

1. Créer un dépôt

Pour créer un dépôt il suffit de cliquer sur le bouton **New project**, lors de la création d'un dépôt il est important de comprendre où doit-il se situer, par défaut il indiquera votre compte au quel cas vous serez hébergeur et il sera accessible à l'URL `https://gitlab.veremes.net/[votre nom]/[nom du dépôt]`. Pour un projet en production il est préférable le créer dans un groupe de manière à ce qu'il hérite des droits des utilisateurs et que son emplacement soit clair, pour cela il suffira de choisir le groupe voulu dans la liste déroulante située à droite de l'URL du projet.

La deuxième chose importante quand on crée un dépôt c'est sa visibilité en lecture et téléchargement, si un dépôt est **Private** alors seuls les personnes membres du projet pourront le visualiser, si il est **Internal** tous les utilisateurs authentifiés sur GitLab pourront y avoir accès ce qui dans notre cas n'a aucun sens car l'inscription est libre, la dernière option est l'option **Public** au quel cas tout le monde pourra avoir accès au dépôt.

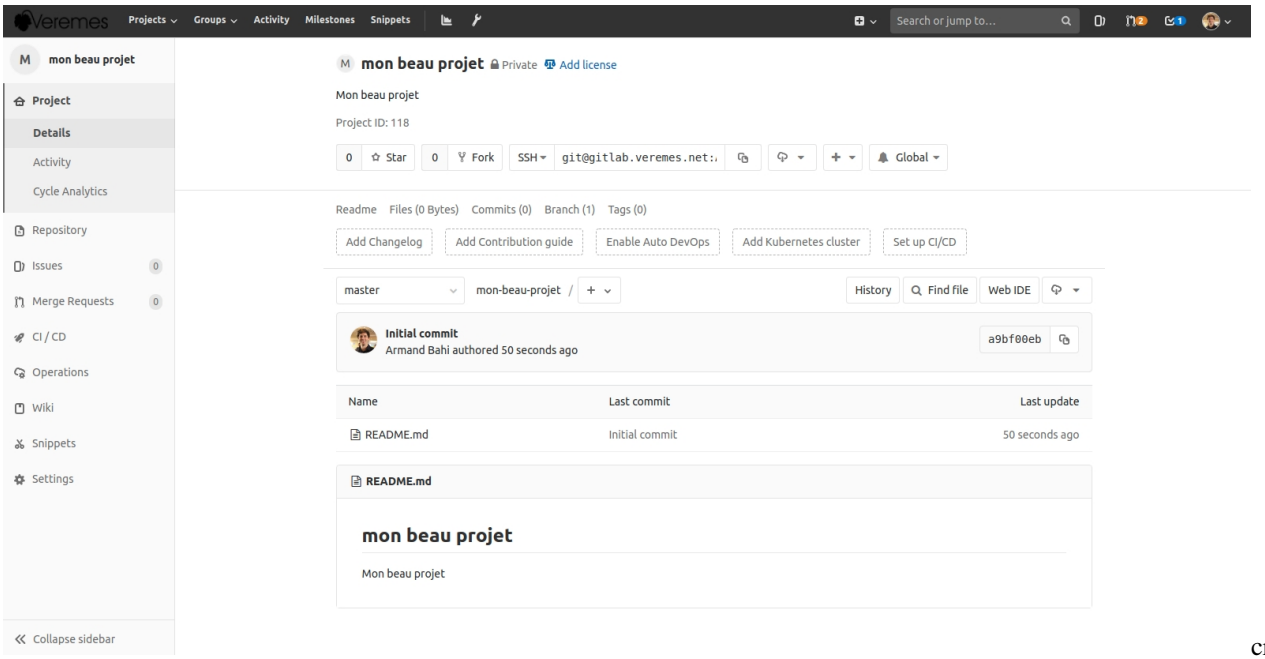
Chaque dépôt doit contenir un fichier README.md contenant une description du projet car ce sera la page d'accueil des visiteurs.



créer un dépôt

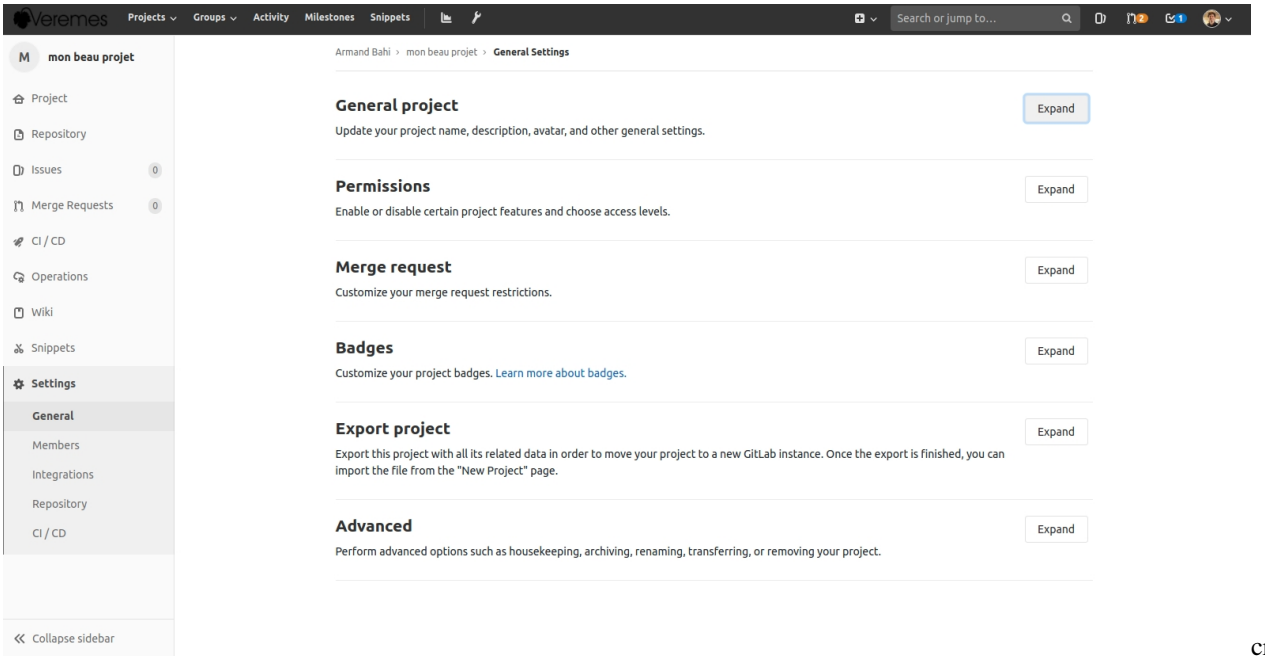
2. Paramétrer le dépôt

Une fois le dépôt crée on accède à la page suivante, on peut déjà le cloner pour alimenter ses fichiers mais nous allons tout d'abord le paramétrer. Pour cela il suffit de se rendre dans la section **Settings**



créer un dépôt 2

Cette section est divisée en plusieurs parties que nous détaillerons pour la plupart ci-après.



créer un dépôt 3

2.1. General

2.1.1 General project

Dans cette partie nous pouvons modifier le nom, la description, la photo etc.. affichés dans GitLab, ceci ne changera pas l'URL du projet et ce dernier pourra continuer à fonctionner sur les postes l'ayant clonné.

2.1.2 Permissions

Ici on peut activer/désactiver les différents modules GitLab disponibles, à Veremes nous utiliserons le paramétrage suivant :

- **Issues** permet la gestion des demandes, doit être acivée selon les besoins
- **Repository** doit être activé
- **Merge requests** doit être activé
- **Pipelines** doit être desactivé
- **Git Large File Storage** doit être desactivé
- **Wiki** doit être desactivé
- **Snippets** doit être desactivé

2.1.3 Advanced

Permet de supprimer le projet ou de changer son URL

2.2. Members

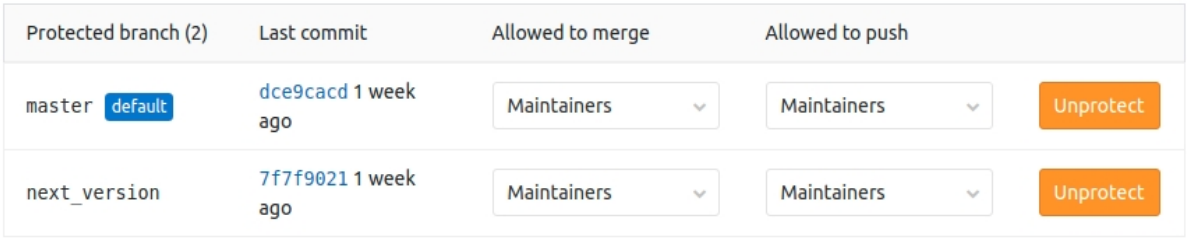
Ici nous pourrons définir quels seront les droits des différents contributeurs, pour plus d'informations vous pouvez lire la section droits GitLab.

2.3. Integrations

Cette partie permettra de mettre en place des Web-hooks pour lier le projet à un logiciel tiers (ReadTheDocs etc..)

2.4. Repository

Dans cette section il est important de ce pencher sur la partie **Protected branches**, c'est ici que nous déciderons des branches à protéger.



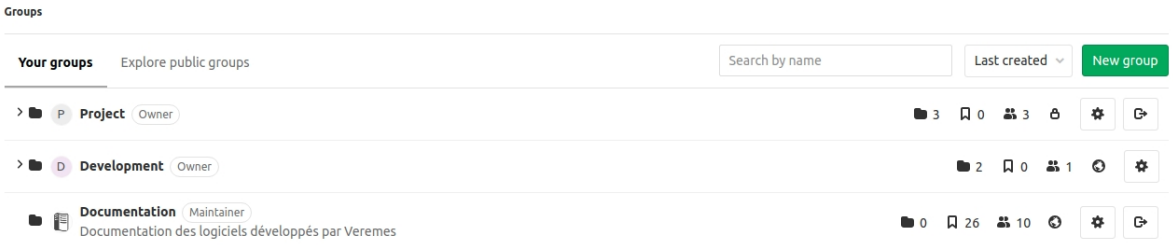
créer un dépôt 4

Il est **fortement conseillé** de protéger les branches **master** et **next_version** de telle sorte que les développeurs ne puissent pas commiter directement dessus, pour apporter leurs modifications les développeurs devront créer des branches puis des merge requests pour que la personne en charge du projet valide les modifications.

Vous trouverez plus d'informations à propos des branches dans le section Corrections et évolutions

1.2 Architecture des projets Veremes

Il y a trois grand groupes utilisés à Veremes, qui permettent de versionner et donner les droits sur l'ensemble des projets.

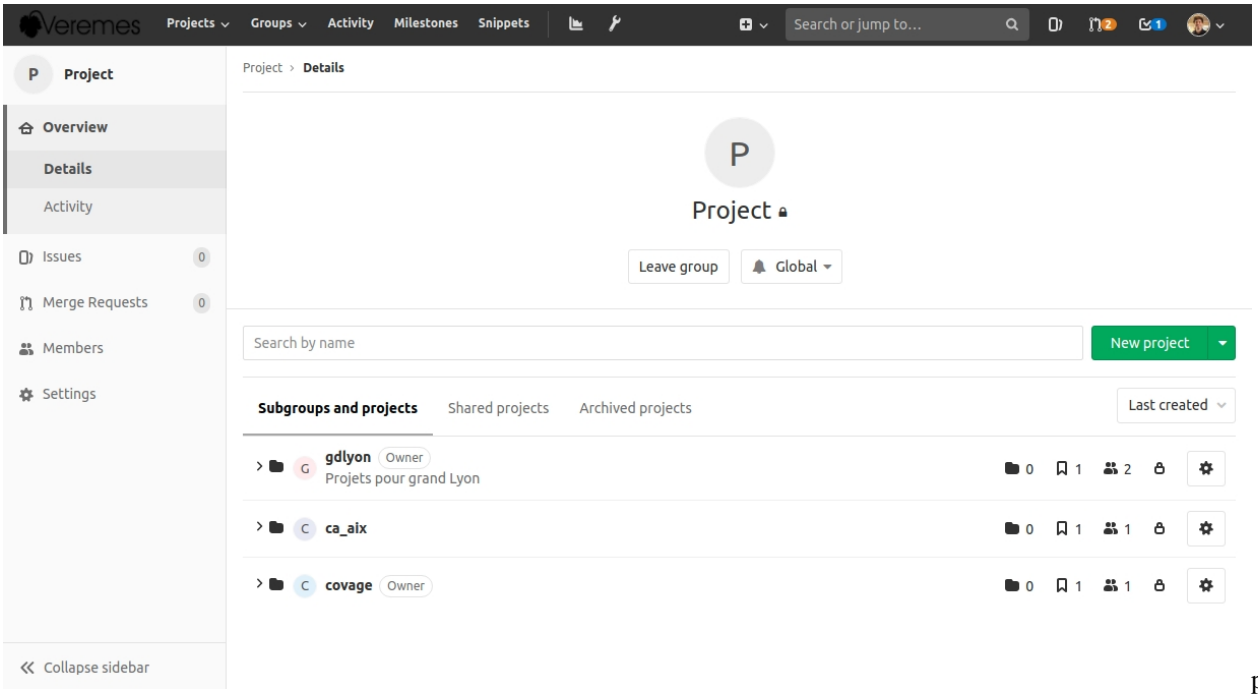


veremes_groups

1.2.1 1. Projet

Ce groupe encadré par Jordi, Yoann et Armand permet de versionner les différents projets faits pour des clients et où il n'y a pas de développement (ex : projets FME, SQL).

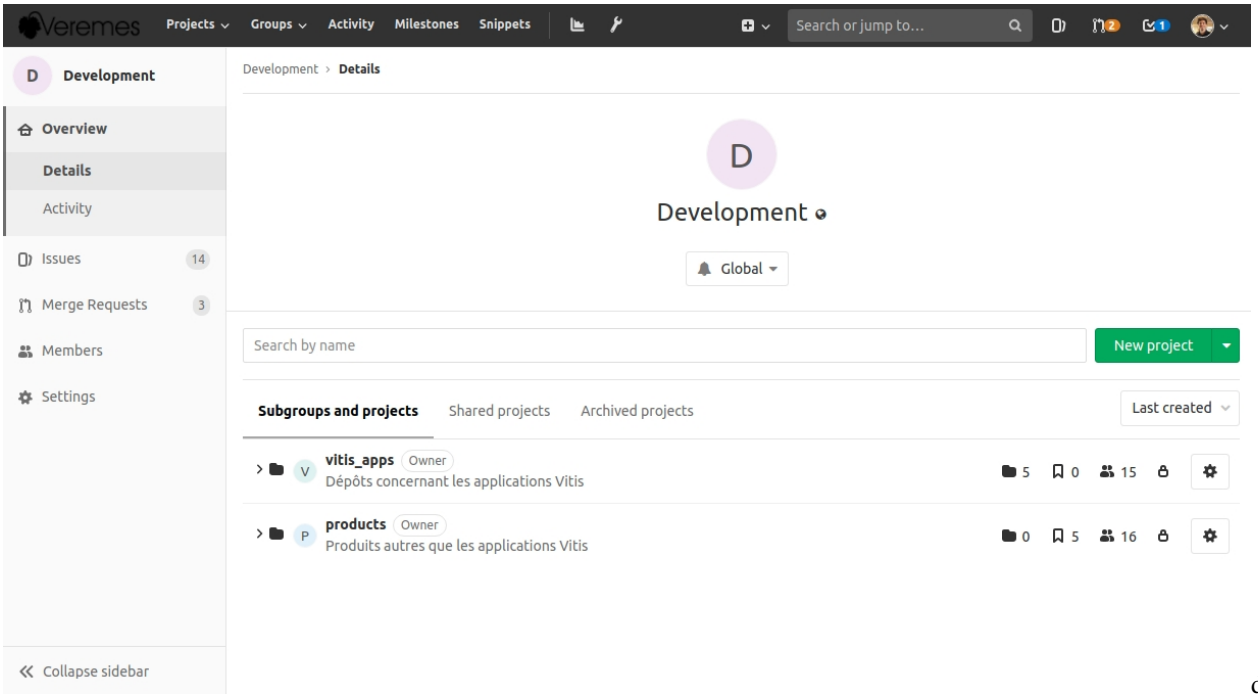
Chaque client est représenté par un sous-groupe qui contiendra les différents dépôts, lorsque vous aurez besoin d'un espace pour stoker des fichiers pour un projet en particulier il faudra demander à Jordi, Yoann ou Armand de créer le dépôt correspondant.



project_group

1.2.2 2. Development

Le groupe dévelopment est scindé en deux sous-groupes **vitis_apps** et **products**.



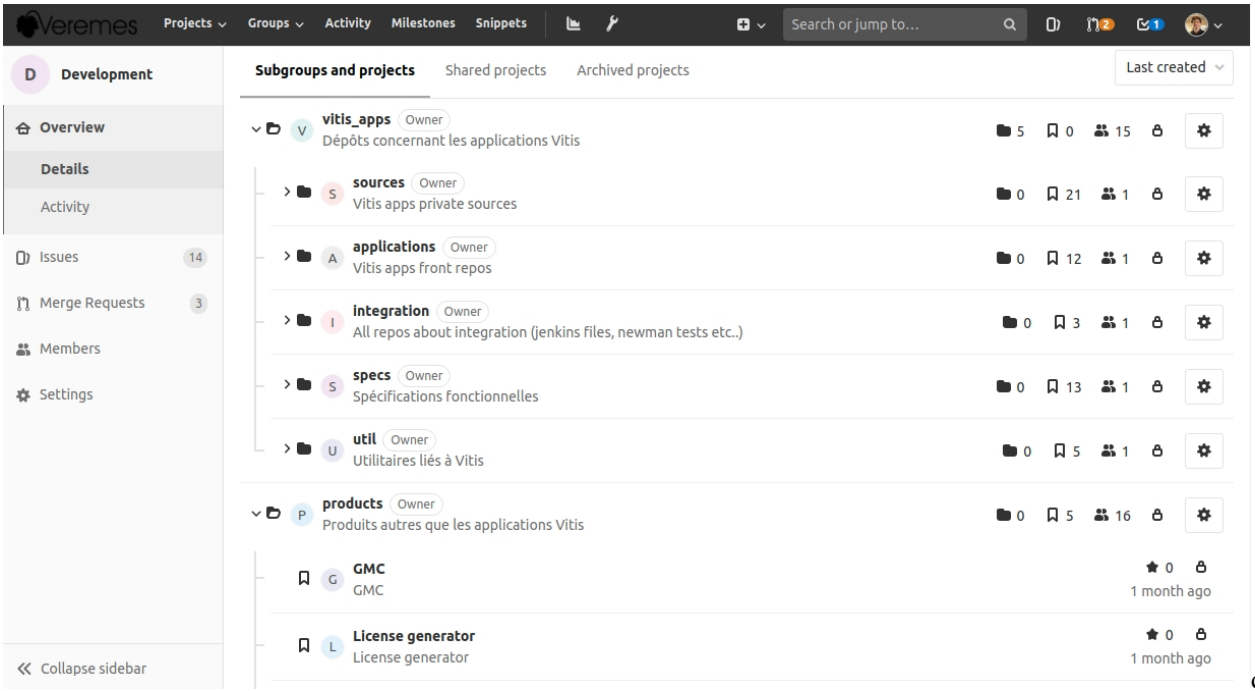
development_grou

2.1. products

Ce sous-groupe contient les différents développements faits pour des logiciels autres que Vitis.

2.2. vitis_apps

Vitis_apps est un ensemble de sous-groupes permettant de versionner tout ce qui touche aux applications Vitis allant des modules aux spécifications.



development_grou

2.2.1. Sources

Dans ce sous-groupe on retrouve l'ensemble des modules ainsi que Vitis, ces sources sont directement contenues dans les applications sous formes de subtrees, pour les modifier il faudra passer par les applications et **il ne faudra pas travailler directement sur les modules** mis à part pour créer ou merger des branches.

2.2.2. Applications

Ici on retrouve l'ensemble des applications Vitis **sur lesquelles il faudra travailler**, chaque application est dirigée par un chef de produit qui validera les différentes issues et merge-requests.

Chaque application contient une copie de chacun de ses modules sous la forme de subtrees, une documentation détaillée sur le fonctionnement de ces applications est disponible ici.

2.2.3. Integration

Contient tout ce qui touche à l'intération continue : tests, génération de setups etc…

2.2.4. Specs

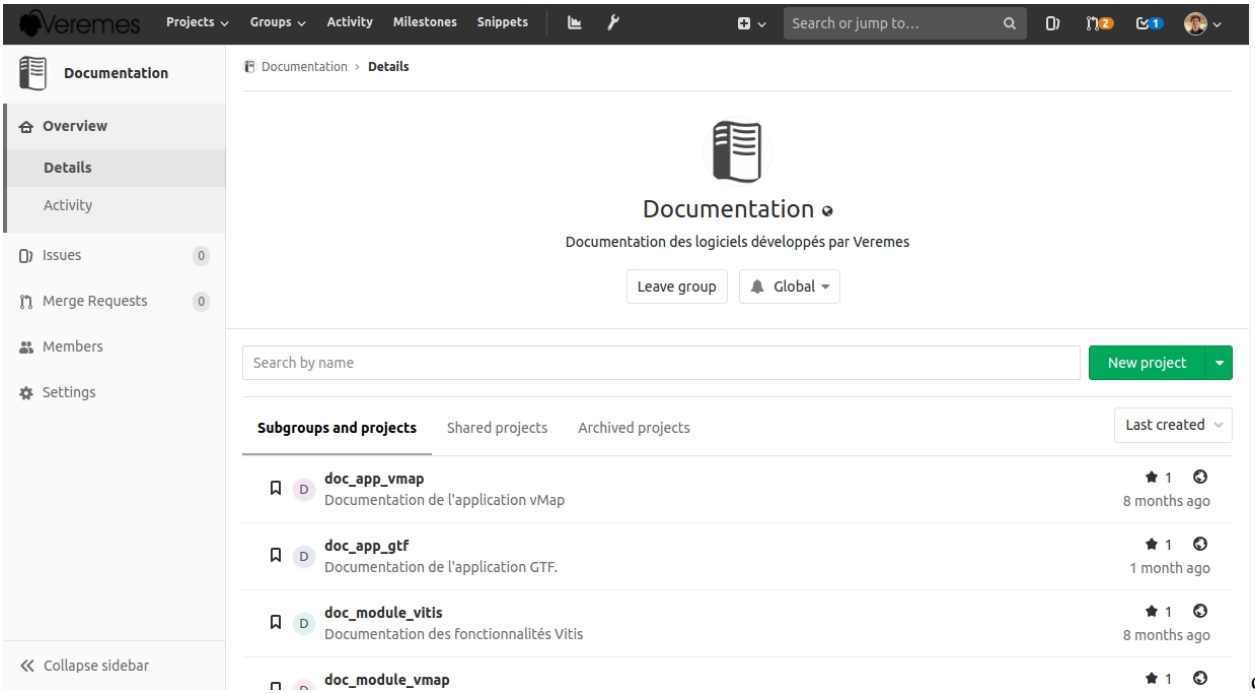
On y retrouvera les spécifications des différentes applications.

2.2.5. Util

Nous retrouverons dans les sous-groupe les utilitaires liés à Vitis pour compiler PHP, lancer les websockets etc…

1.2.3 3. Documentation

Ce groupe est encadré par Armand et Marot contient l'ensemble des documentations des applications.



documentation_gr

On y retrouve deux types de depôts :

3.1. Modules

Les dépôts de documentation des modules utiliés par les applications et se nomment doc_module_[nom du module]

3.2. Applications

Se nomment doc_module_[nom du module] ces dépôts contiennent à l'aide de *submodules* des copies à l'instant T des modules qu'ils contiennent.

Un même module peut être utilisé par plusieurs applications comme c'est le cas pour doc_module_vitis

3.3. Synchronisation

Des web hooks ont étés mis en place pour qu'à chaque fois qu'un module est modifié, alors toutes les applications qui le contiennent mettent à jour leurs *submodules*. Une documentation à été rédigée à ce sujet ici

1.3 [Procédure] Branches Correctifs/évolutions

1.3.1 1. Fonctionnement

1. Branches fixes

Pour les projets FME

Quand les projet abritent des fichiers non textuels (audio, fmw, zip etc..) des conflits non résolvables vont apparaître régulièrement lors des étapes de merge, dans ce cas il est préférable de créer une branche de travail ouverte à tous les contributeurs, de protéger la branche master pour y faire des merge quand les développements ont étés validés.

Pour les projets de développement

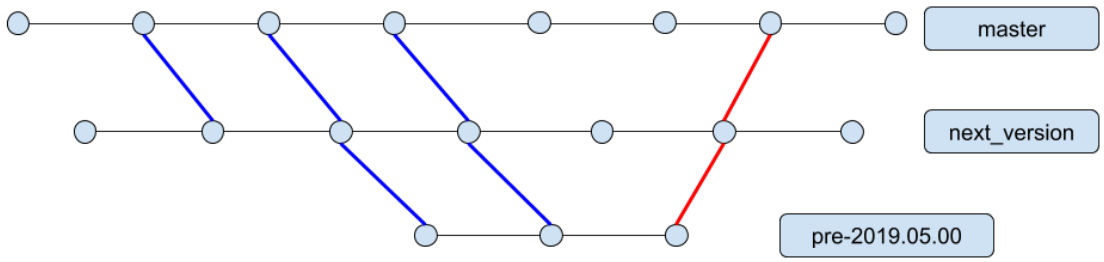
Dans ce cas il est possible d'utiliser 100% des capacités de Git et c'est ce que nous allons faire en appliquant la procédure suivante.

Sur la branche **master** nous appliquerons uniquement les correctifs, cette branche **doit absolument être stable** à tout moment car elle permettra de générer un correctif si on le souhaite.

Une deuxième branche **next_version** permettra quand à elle à appliquer les évolutions, quand des correctifs sont appliqués à master, il faudra les merger sur cette branche de manière à avoir les dernières évolutions ainsi que les derniers correctifs.

Une fois les développements d'une version terminés il est important de tester dans un environnement de pré-production notre application, à ce moment nous créerons à partir de next_version une branche **pre-20xx.xx.xx** contenant le numéro de build à générer (on utilise parfois **pre_prod** si il n'y a pas de numéro associé). Les correctifs à effectuer se feront sur master puis seront mergés sur next_version et sur la branche de pré-production.

Quand l'environnement de pré-production est validé, il est alors mergé sur next_version puis sur master à partir duquel on créera le tag correspondant.



branches

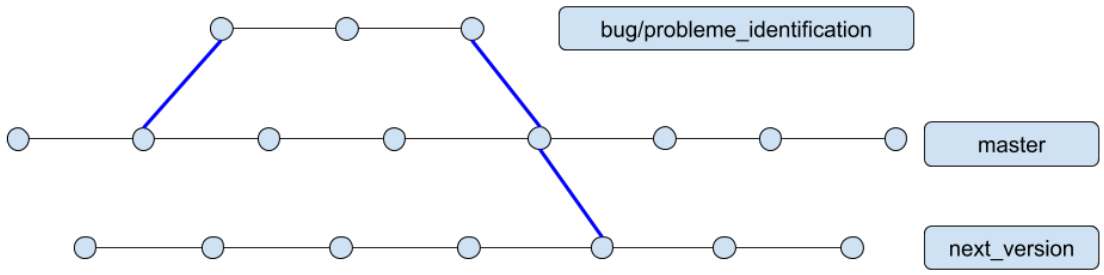
depots

2. Faire un correctif

Cette section est réservée aux projets de développement

Un correctif est un bug remonté sur l'application en cours de production qui doit être corrigé pour publier une nouvelle version mineure, pour ce faire on créera une branche **à partir de master** qui prendra comme nom l'intitulé du bug (nom automatiquement pre-saisi par GitLab).

Une fois le bug corrigé nous le rapatrierons sur master à l'aide d'une merge request puis nous effectuerons une deuxième merge request de master vers next_version



branches

depots 2

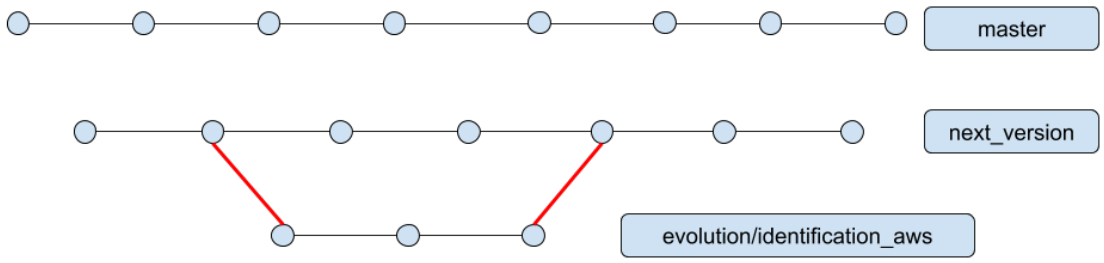
Chaque correctif doit être fait à travers une issue (documenté plus bas)

3. Faire une évolution

Cette section est réservée aux projets de développement

Les évolutions ne sont pas intégrées sur master qu'après la phase de pré-production, c'est pour cela qu'elles seront faites **à partir de next_version** et qui prendra comme nom l'intitulé du bug (nom automatiquement pre-saisi par GitLab).

Une fois terminée, la branche sera rapatriée sur next_version.



branches

depots 3

Chaque évolution doit être faite à travers une issue (documenté plus bas)

1.3.2 2. Créer un correctif/évolution dans GitLab

Dans GitLab la notion de correctif ou d'évolution est lié à une issue, une issue est une demande aux développeurs de résoudre un problème sur l'application actuellement en production ou alors de développer une nouvelle fonctionnalité.

Gitlab permettra en 4 étapes effectuer tout le processus de création de demande, création de branche, demande de validation et intégration.

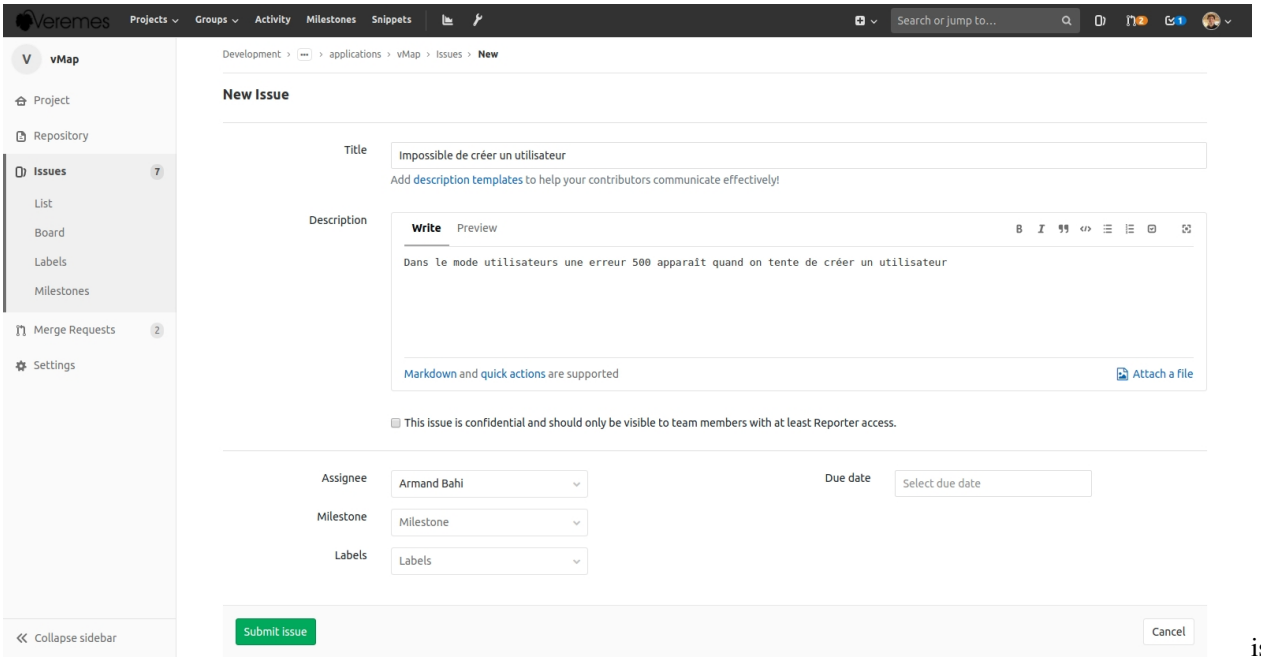
2.1 Céation d'une issue

Le titre de l'issue doit contenir sur quelques mots le but de la demande, puis dans la description il est possible de détailler.

En cas de bug, il vaut mieux associer la demande au responsable produit de telle sorte à ce qu'il reçoive par mail et dans l'interface une notification.

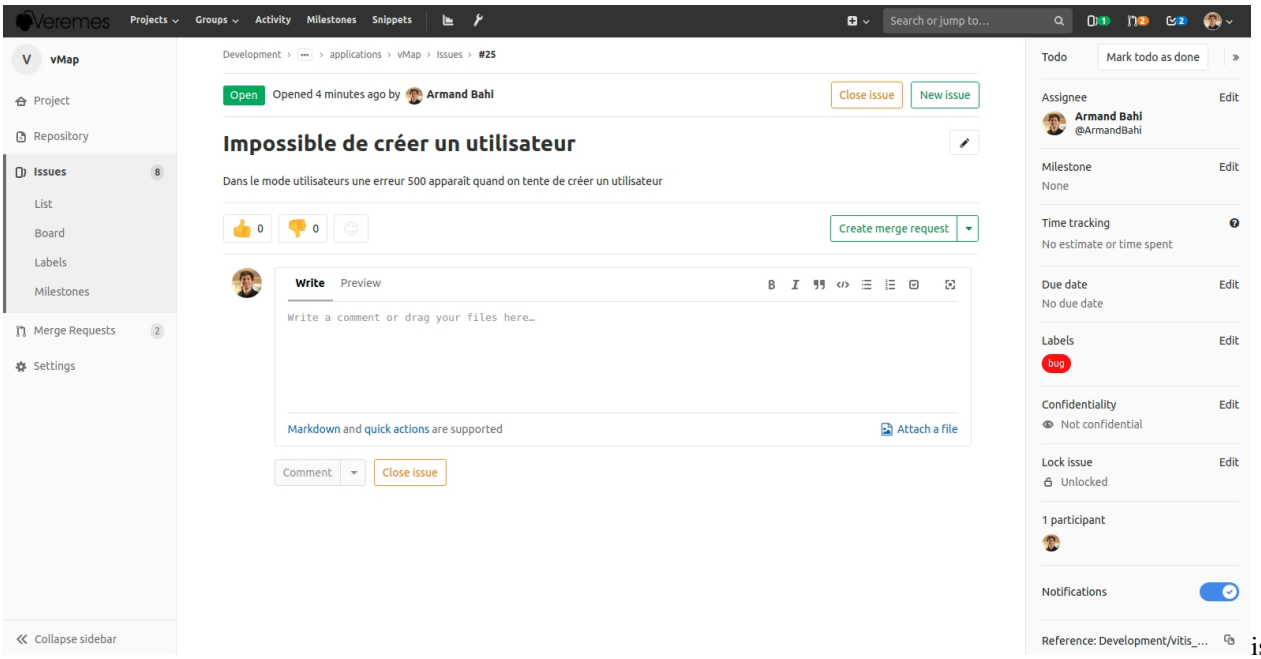
Pas besoin de fournir de milestone car ce dernier sera décidé par le responsable produit.

Parmi les labels disponibles il faudra spécifier si il s'agit d'un bug ou d'une évolution.



issues 1

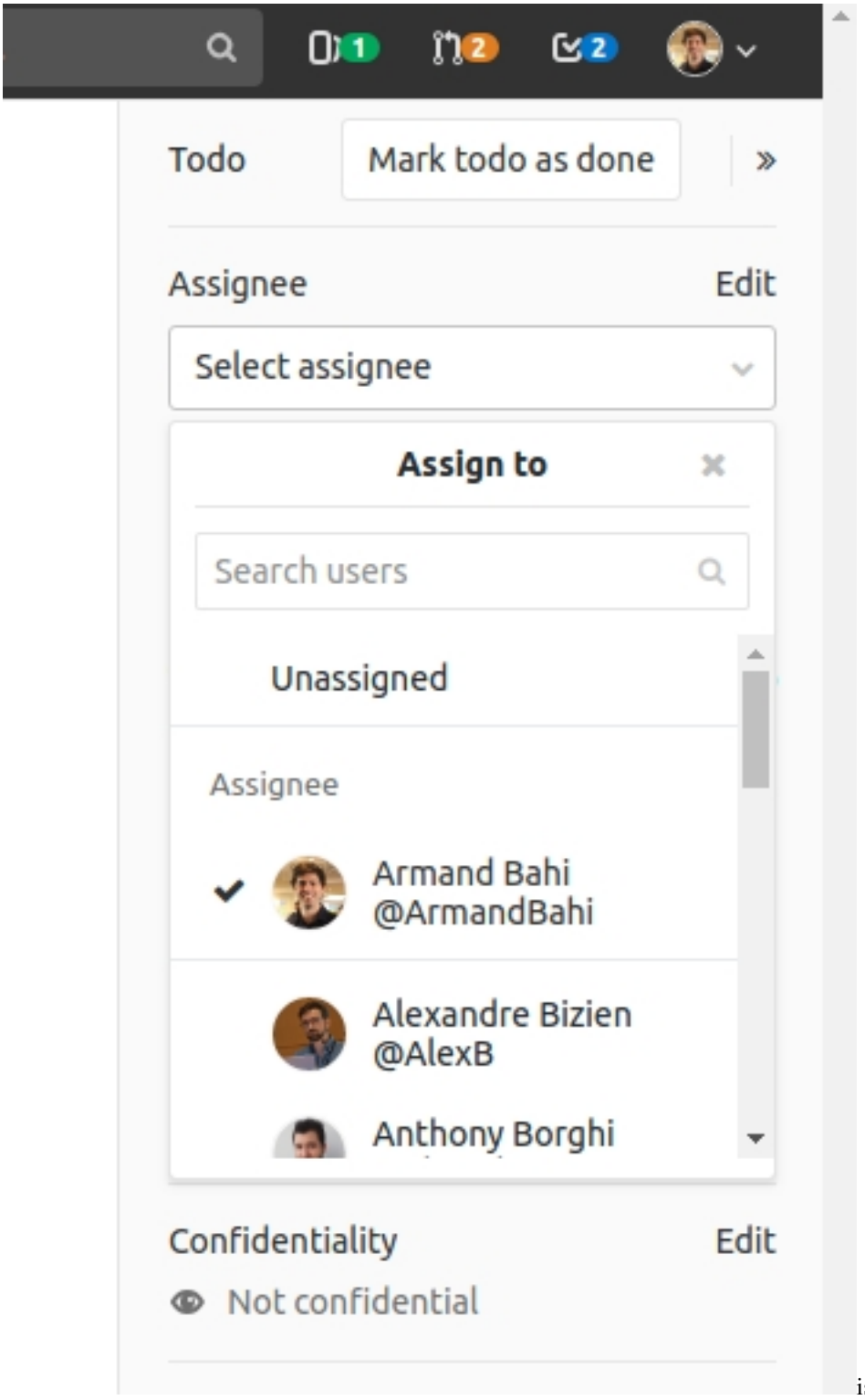
Lorsqu'un responsable produit va accepter une issue il voudra si c'est un bug créer une branche ainsi qu'une merge request, pour cela plusieurs étapes sont nécessaires.



issues 2

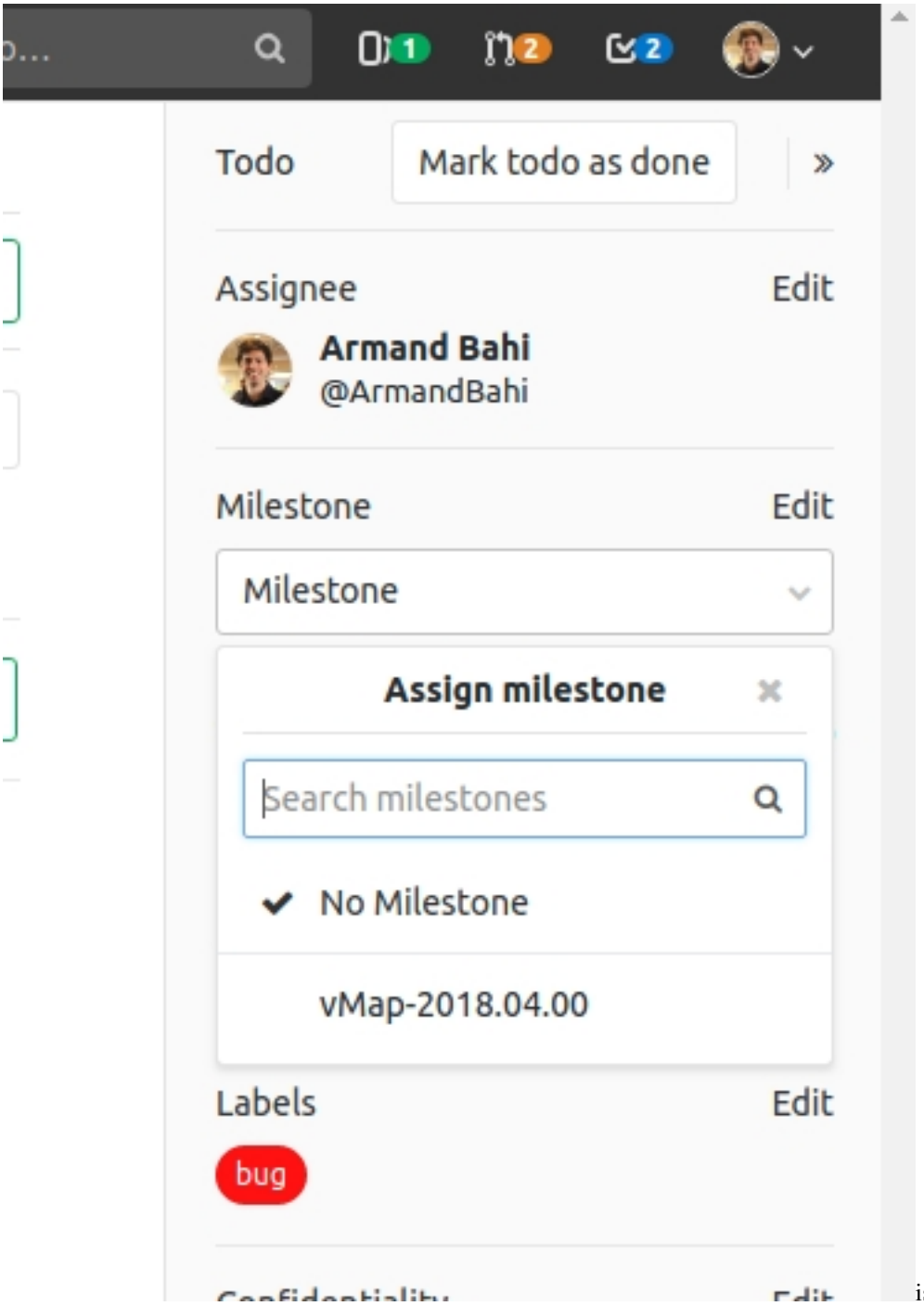
2.2 Assigner la demande

En assignant la branche à quelqu'un il recevra une notification par mail ainsi que sur l'interface.



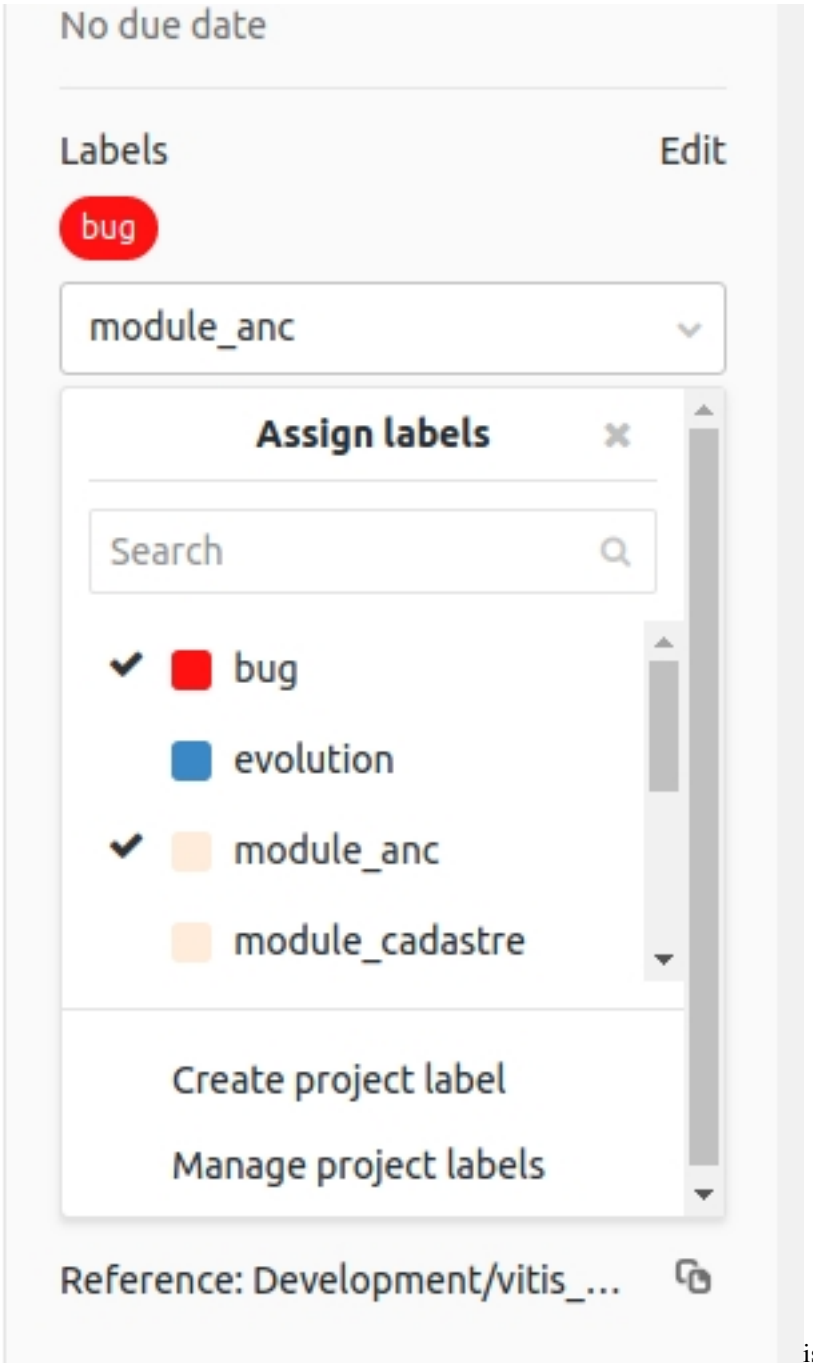
issues 3

Le milestone permettra de connaître l'avancement du projet dans sa globalité, il est important de le spécifier pour savoir à quelle version de l'application correspond cette issue.



issues 4

Les labels permettront de filtrer les issues, les deux labels bug et evolution sont toujours présents, sur les applications vitis on retrouve autant de labels que de modules pour savoir rapidement à quel module correspond chaque issue.



issues 5

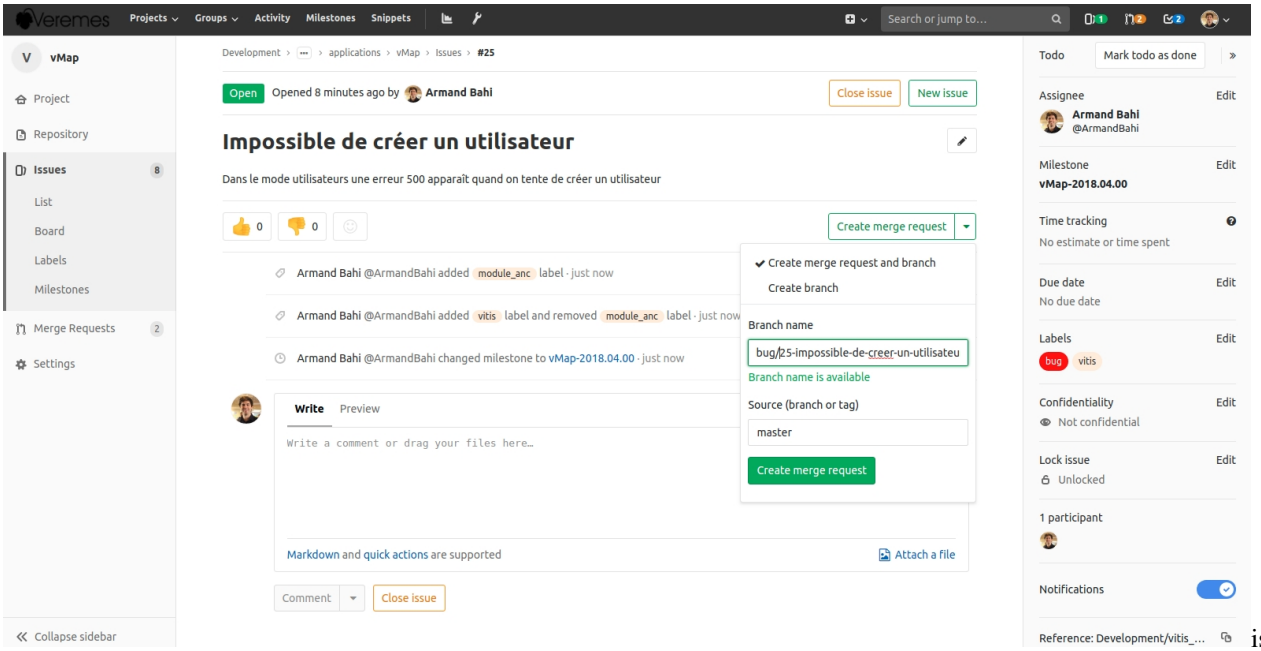
2.3. Corriger la demande

Quand l'issue vous est attribuée et que vous commencez à la résoudre il faudra créer de manière automatique une branche ainsi qu'une merge request.

Ne cliquez pas directement sur le bouton Create merge request, en revanche cliquez sur le bouton déroulant situé à sa droite pour définir le nom de la branche à créer ainsi que la source.

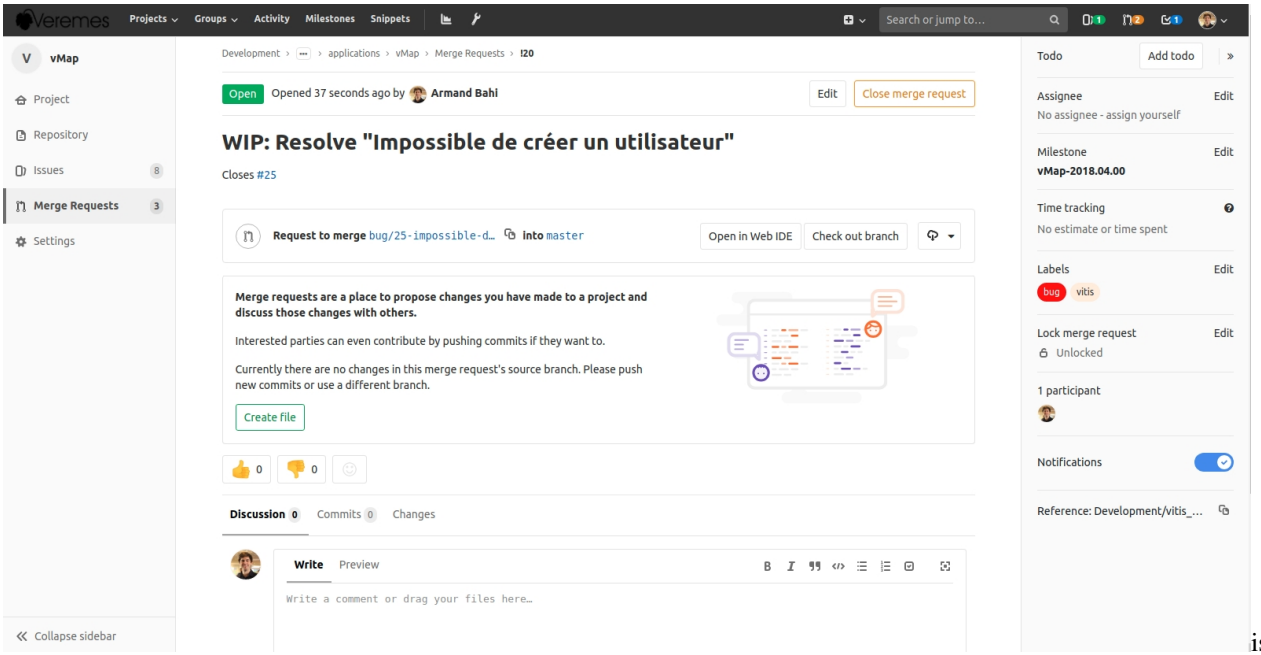
Si il s'agit d'un bug on choisira comme source master.

Si il s'agit d'une évolution on choisira comme source next_version



issues 6

Désormais une branche portant le nom de l'issue a été créée. Vous remarquerez que le nom de la merge request commence par **WIP** (Work In Process) ce qui nous permet de savoir si la travail du développeur est terminé ou non.



issues 7

Une fois les développements terminés sur la branche il est temps de terminer la merge request pour demander au responsable projet d'intégrer les développements sur la banche source. Pour cela il faudra lui assigner la merge request puis enlever le status WIP en cliquant sur **Resolve WIP status**.

2.4. Valider la demande

Une fois que le développeur a terminé et vous a assigné la demande de merge vous pouvez visualiser les modifications effectuées et valider la demande.

Si la demande ne correspond pas à vos attentes il faudra l'éditer pour ajouter à nouveau WIP: au début du titre, puis décrire dans la fentre de discussion pourquoi la demande n'est pas valide.

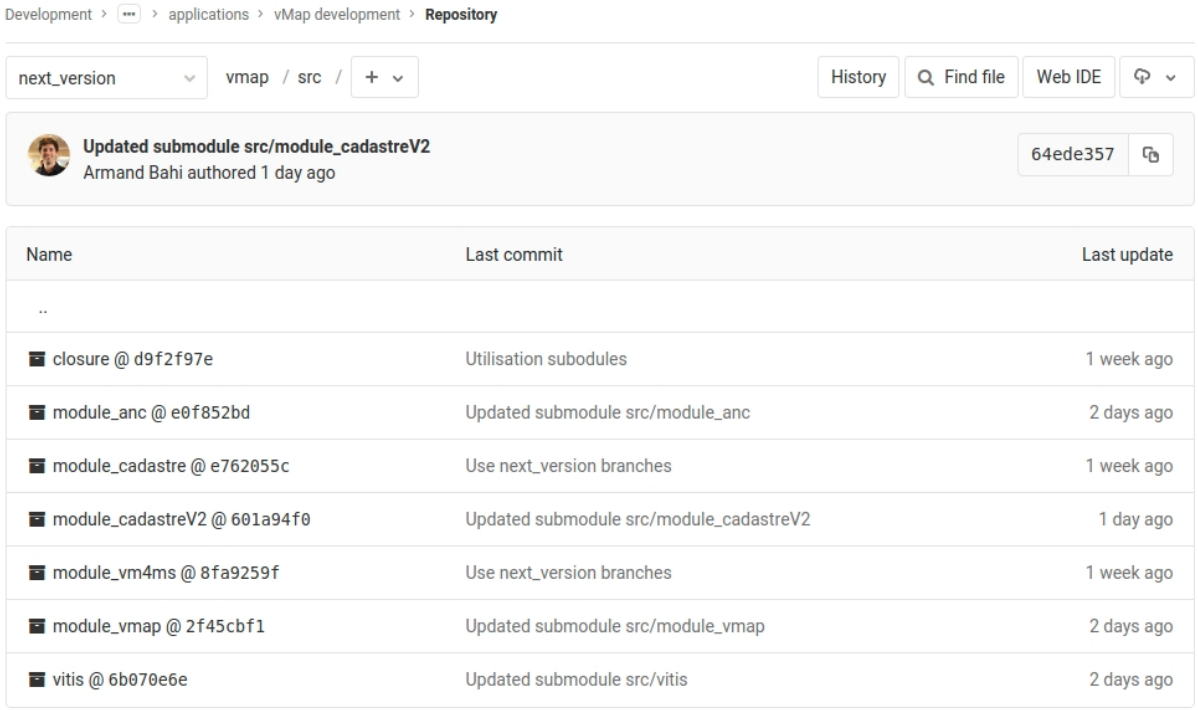
Une fois la demande de merge effectuée l'issue se met à jour automatiquement en état terminé.

1.4 [Procédure] Applications Vitis - Submodules

1.4.1 1. Definition

Les applications Vitis sont des ensemble de modules composés potentiellement eux mêmes de SQL, Backend et Frontend, chaque application ainsi que chaque module doivent être versionnés, tagués et mis à jour en permanence, comme vous pouvez l'imaginer tout ceci rend l'opération particulièrement difficile à versionner.

Pour faire ceci le plus simplement possible, nous utiliserons des submodules, un submodule est clairement un dépôt contenu dans un dépôt mais en gardant un certain lien.

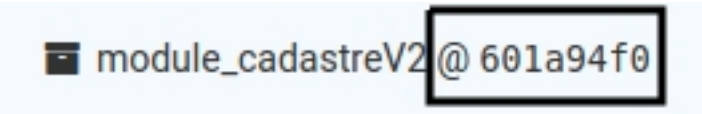


L'image ci-dessus est l'exemple du dossier src/ de vMap, on retrouve dans ce dossier l'ensemble des modules et autres dépendances utilisées par l'application :

- closure : utilisé pour la compilation
- module_anc : module anc
- module_cadastre : module cadastre
- module_cadastreV2 : module cadastreV2
- module_vm4ms : module vm4ms

- module_vmap : module vMap
- vitis : framework Vitis

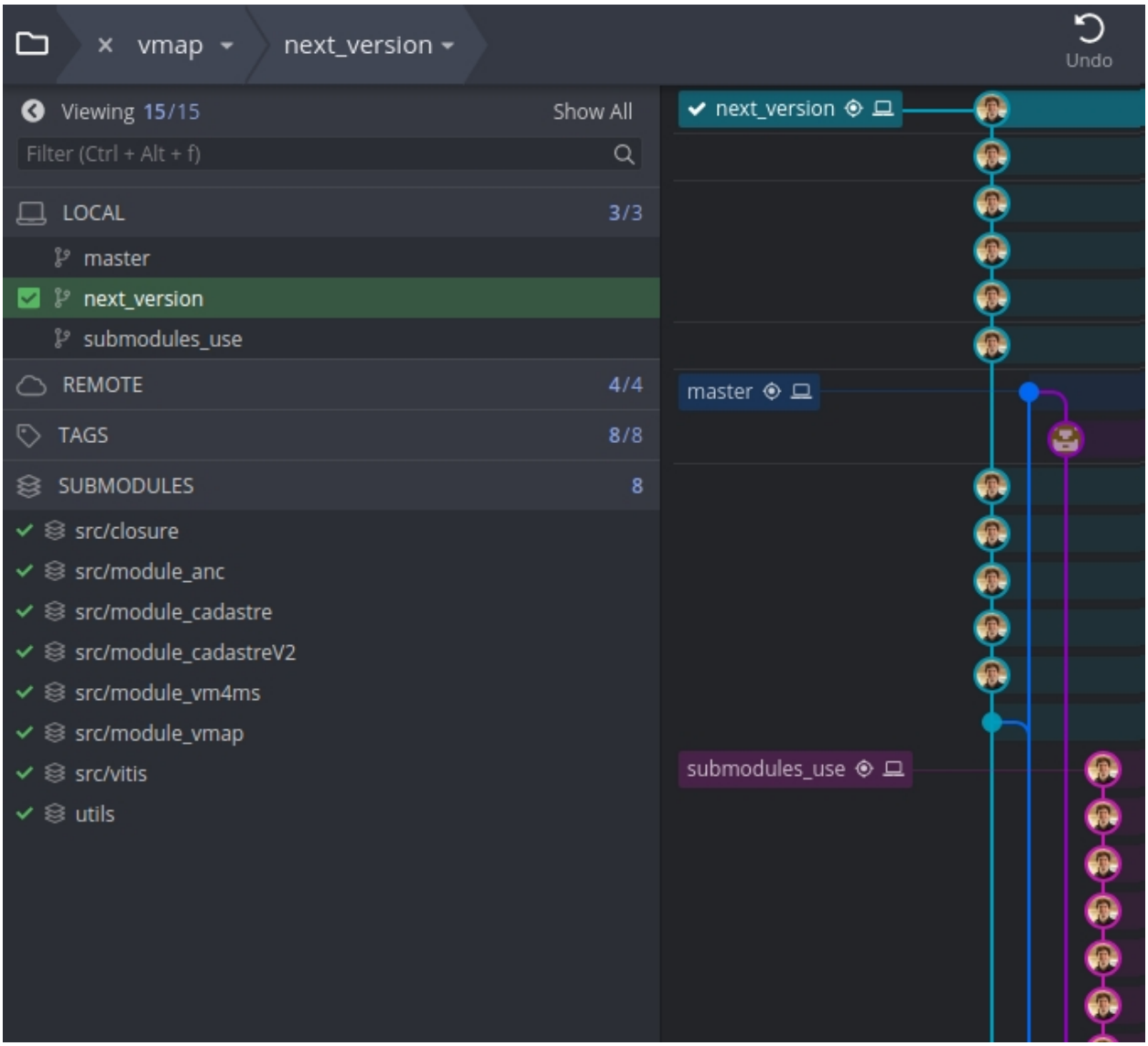
Vous remarquerez sur cette image que chaque submodule inscrit après son nom une référence :



Cette référence correspont à un commit, il est très important de comprendre que **un submodule contient un depôt non sur sa version en cours mais sur un commit précis**.

Gitkraken gère parfaitement les submodules, en ouvrant le projet il va directement détecter l'ensemble des submodules utiliés, puis il permettra de les administrer facilement.

Sur l'image ci-dessous on voit le projet vMap contenant les différents submodules.

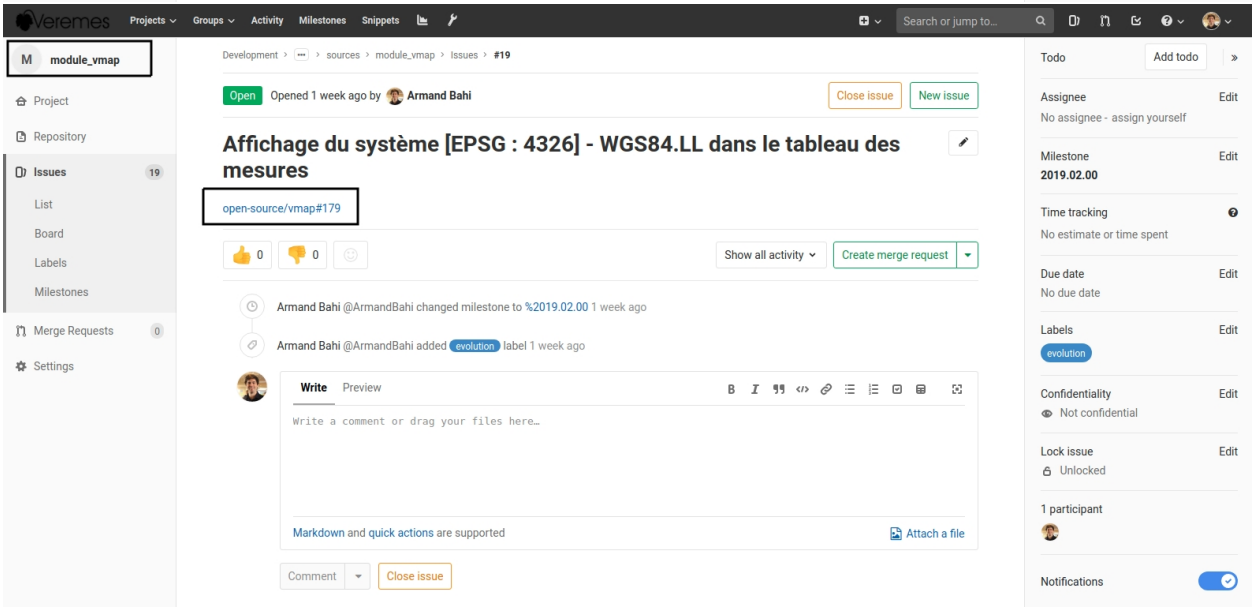


1.4.2 2. Effectuer une correction / évolution

2.1. GitLab

Chaque issue est répertoriée dans un premier lieu au niveau de l'application, lorsqu'elle sera identifiée l'administrateur la copiera dans le(s) module(s) impacté.

Lors de la copie il mentionnera l'URL de l'issue de l'application, ainsi un lien sera crée entre les deux.



| |:————————————-:|:————————————-:| | Application | Module |

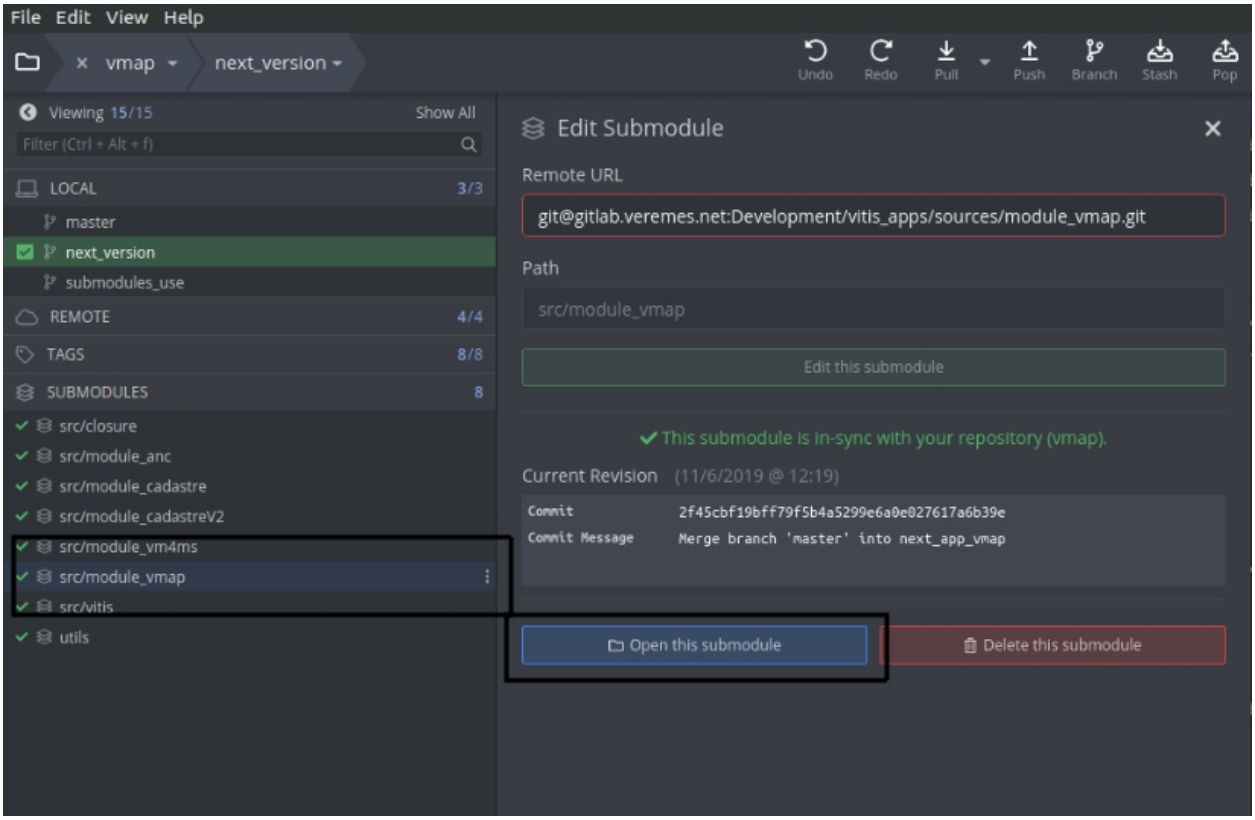
Pour commencer le correctif, le développeur créera **depuis l'issue du module** une merge request ce qui aura pour effet de créer également la branche associée.

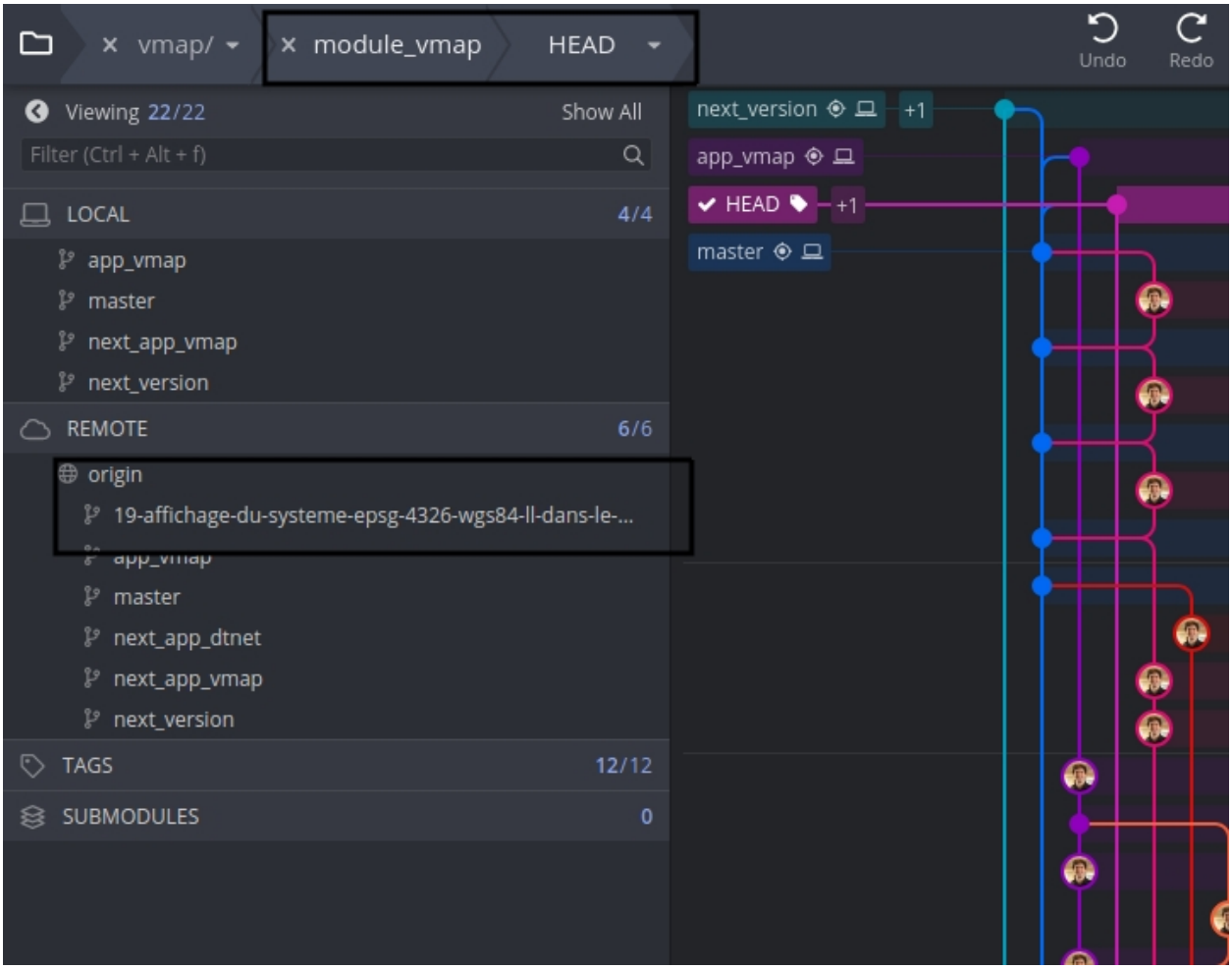
2.2. GitKraken

2.2.1. Avant développement

Pour faire une correction ou une évolution il va falloir modifier le contenu d'un des modules, dans notre exemple il s'agit d'une évolution au niveau du module_vmap.

En doublecliquant sur le submodule GitKraken me propose de l'ouvrir.





| |:————————————-:|:————————————-:| | Application | Module |

Une fois dedans deux détails sont importants à remarquer.

- Sur le bandeau du haut on voit bien qu'on se situe sur le submodule, plus précisément sur une branche **HEAD**.
- Parmis les branches disponibles, on me propose la branche correspondant à l'issue que je veux corriger.

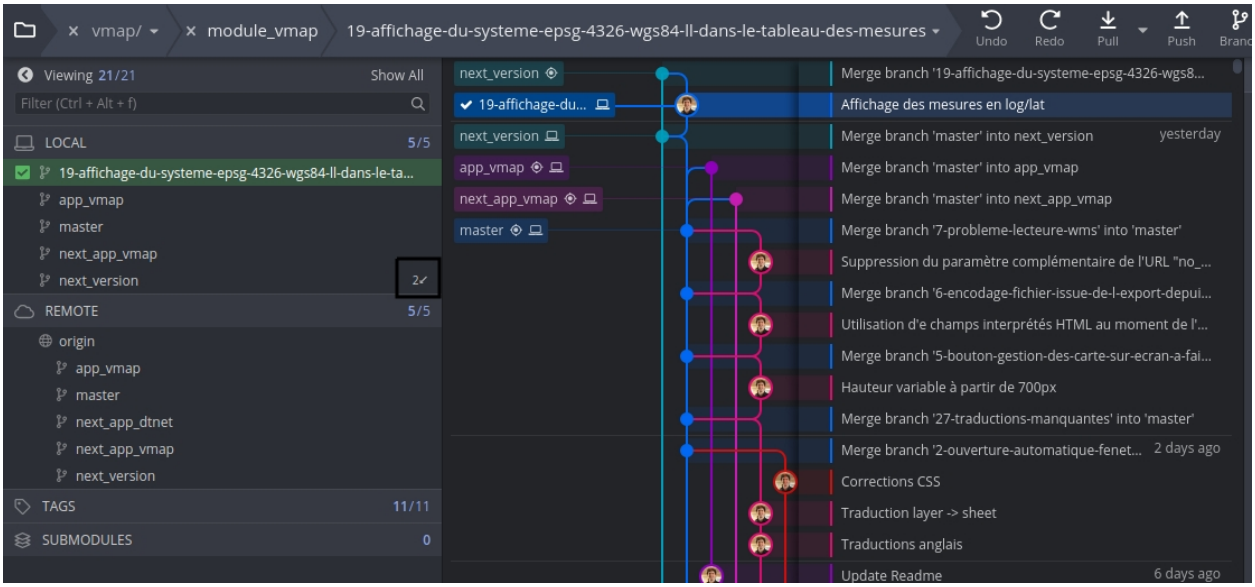
Pour commencer le developpement, double-cliquer sur la branche en question pour se placer dessus.

Vous pouvez désormais effectuer vos correctifs directement dans le répertoire src/ de l'application.

2.2.2. Après développement

Une fois les développements effectués vous pourrez commiter sur la branche impliquée, faire le push, puis assigner le responsable de projet.

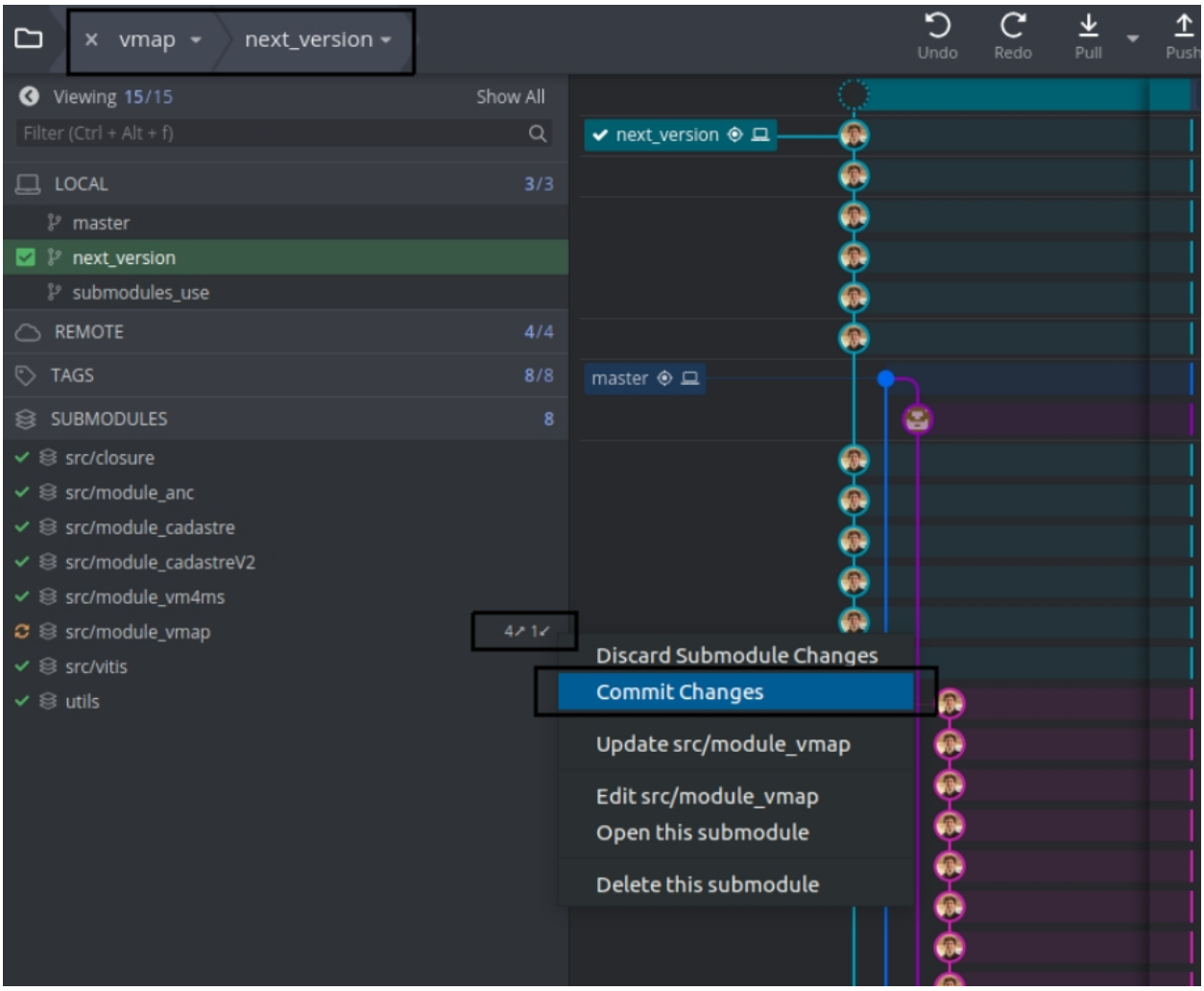
Une fois la merge request acceptée, GitKraken nous indique que nous devons faire un pull sur next_version (car sur l'exemple il s'agit d'une évolution).



Il ne restara qu'à re-passer sur master/next_version puis faire un pull pour rapatrier les modifications précédament effectuées pour que le module soit à jour.

L'application quand à elle pointe toujours sur le même commit qui lui sert de référence.

En remontant au niveau de l'application sur GitKraken on remarque donc qu'il y a eut des modifications sur module_vmap.



En faisant un clic droit et en cliquant sur **Commit changes** on modifie le commit de référence par le nouveau, désormais lorsque quelqu'un fera un clone il disposera de la nouvelle feature.

1.5 [Procédure] Applications Vitis - Publier version

On peut facilement publier une version en 4 étapes à l'aide d'outils automatisés, on distinguera deux types de versions, les **versions mineures** correspondant à des corrections de bugs qui seront taguées en changeant uniquement le dernier nombre (ex: 2018.01.**xx**), les **versions majeures** correspondent à des évolutions apportées au produit, elles seront taguées en changeant le nombre du milieu (ex: 2018.**xx**.00).

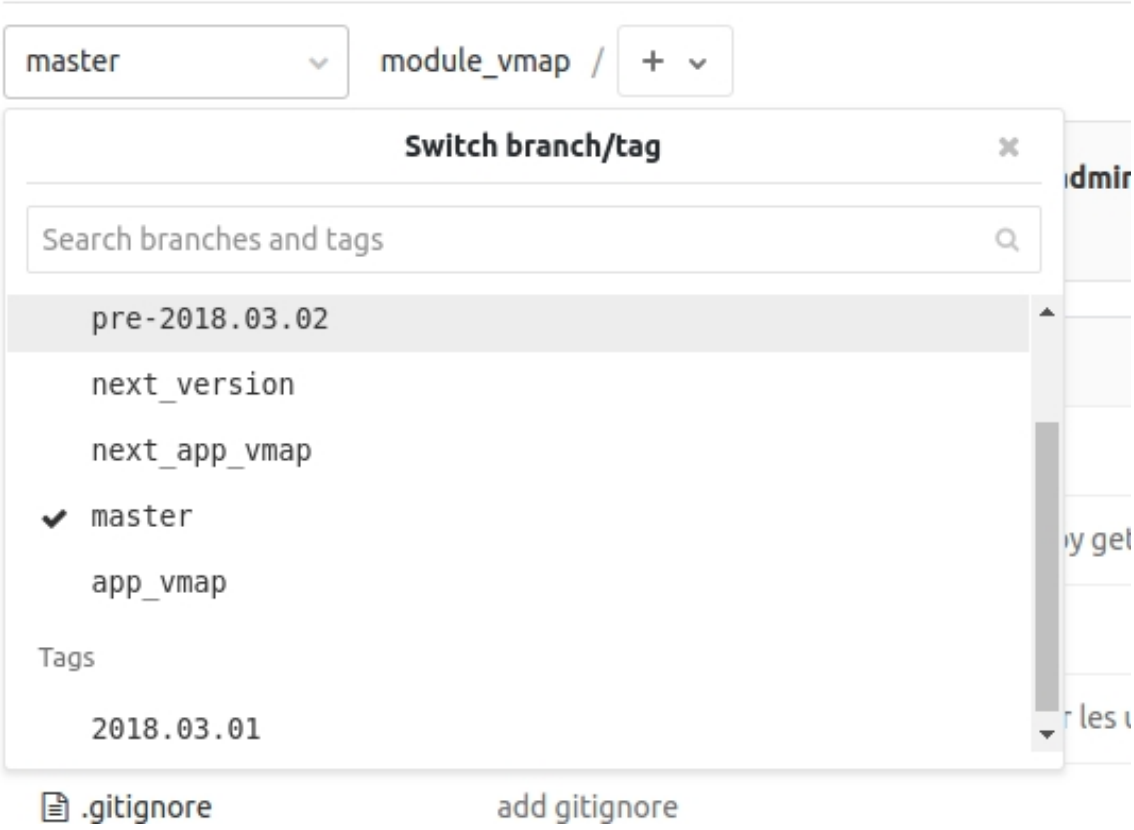
1.5.1 1. Versionner les modules impactés

La première chose à faire c'est de versionner les modules modifiés, pour cela on va aller sur la branche impactée pour créer une branche qui portera comme nom le nom **pre-2018.xx.xx**.

Dans notre exemple on veut créer une version mineure, la version en cours est la 2018.03.01 on créera donc une branche pre-2018.03.02

La deuxième chose à faire c'est de versionner l'application, pour cela on créera une branche portant le nom de la version à générer, puis on modifiera le fichier **conf/_install/dependency.xml** pour utiliser les branches de pre-production

et enfin on utilisera GitKraken pour rapatrier les nouvelles branches sur l'application.

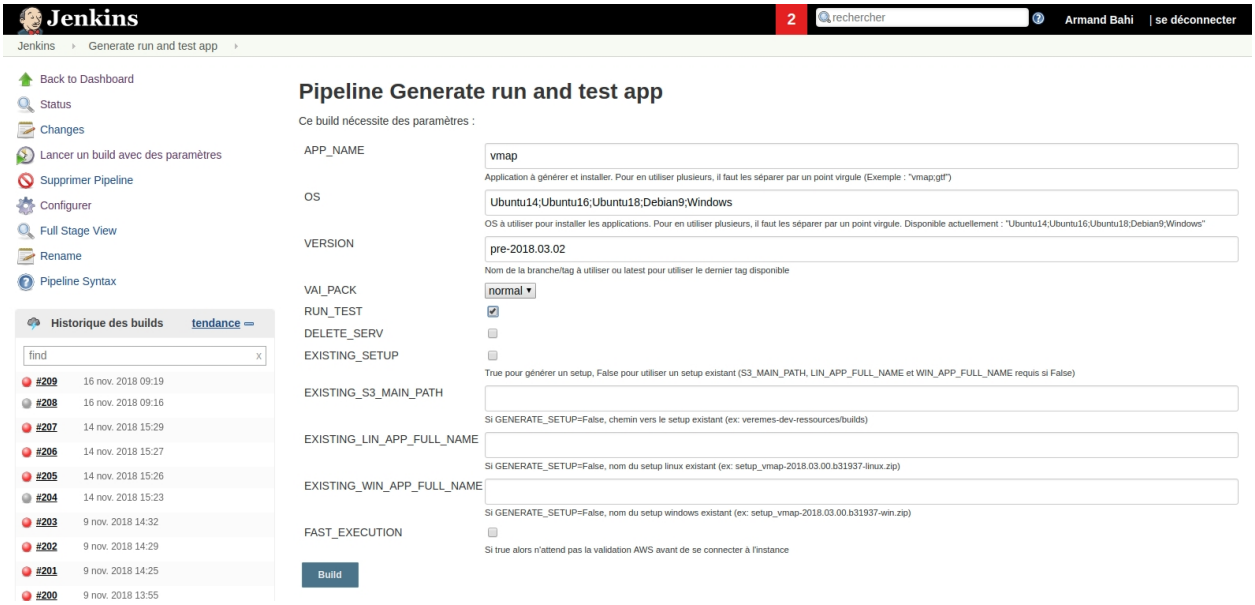


1.5.2 2. Générer une version pre-prod et effectuer les tests

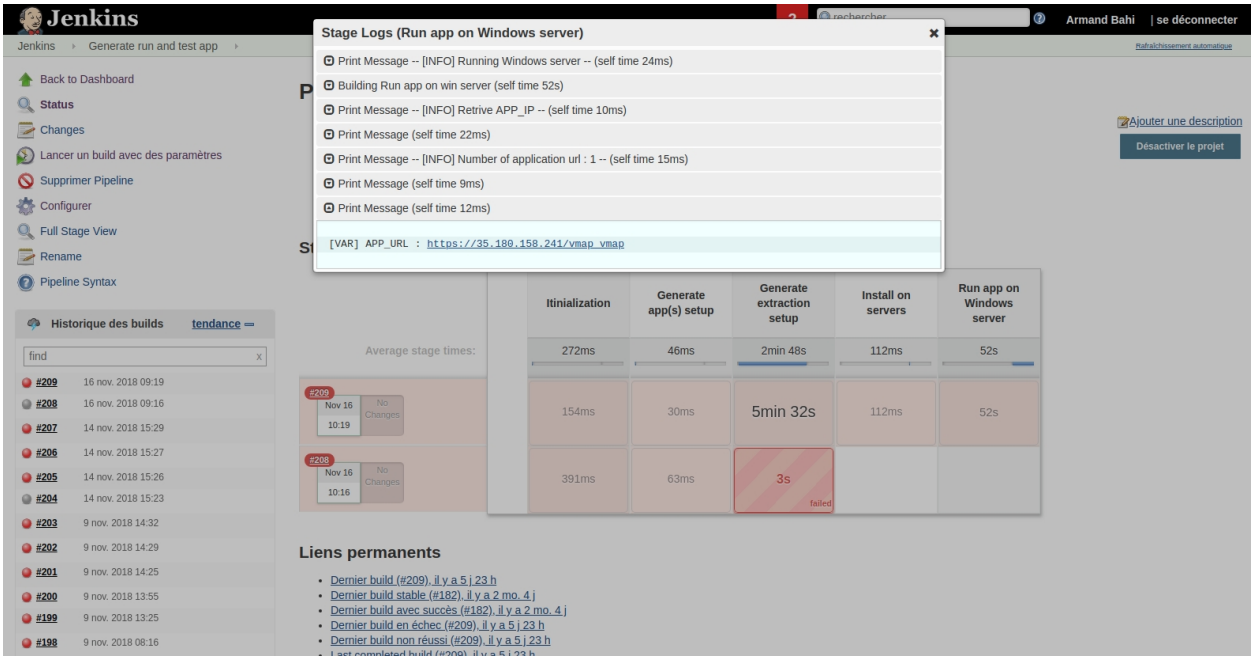
Maintenant que la structure est versionnée il faudra générer un setup et effectuer les tests, pour cela il suffit de ce rendre sur http://jenkins.veremes.net/ et lancer un job de type **Generate un and test app**. Ce dernier va générer un setup, le poser sur S3, démarrer une machine AWS, installer la version dessus et lancer les tests API-REST.

Pour lancer le job il faudra saisir les paramètres suivants:

- **APP_NAME** : nom de l'application à générer
- **OS** : le ou les OS sur lesquels installer
- **VERSION** : numéro de version à générer
- **RUN TEST** : cocher la case pour lancer les tests API-REST (en cours de développement)



Une fois le job terminé vous pouvez récupérer l'url de l'application générée en cliquant sur l'étape "*Run app on . . .*" puis aller tester manuellement l'application.



Il est important d'effectuer également des tests en mise à jour, aucun job n'a été développé pour l'instant donc il faudra effectuer le test manuellement.

1.5.3 3. Créer les tags associés

Une fois que les tests sont concluants on pourra créer les tags associés pour ceci il faudra sur chacune des branches modifiées appliquer la procédure suivante :

1. Faire un merge vers master
2. Créer un tag à partir de la branche pre-prod

Pour l'application la procédure est légèrement différente :

1. Modifier dependency.xml pour utiliser les tags créés précédemment
2. Utiliser GitKraken pour tester localement la configuration
3. Créer un tag à partir de la branche pre-prod

1.5.4 4. Générer une version de production

La dernière étape consiste à rejouer l'étape 2.2 puis récupérer le setup généré sur https://s3.console.aws.amazon.com/s3/buckets/veremes-dev-ressources/builds/

Une fois que tout est ok, on peut supprimer les branches de pré-production

1.6 [Procédure] Applications Vitis - Créer application

La première étape est de créer un dépôt dans Development/vitis_apps/application/[nom de votre application] il sera utile de d'initialiser avec ce dépôt les fichiers traditionnels README.md et .gitignore

1.6.1 2. Dossier conf

Une fois le dépôt en place nous allons mettre en place les seuls fichiers de code qui seront relatifs à l'application (les autres seront liés à des modules), pour cela créer un dossier conf contenant la structure ci-dessous :

- install/
  - apache.conf
  - dependency.xml
  - fileToDelete.txt
  - folderToDelete.txt
- less/
  - variables.less
- requires/
  - config.js
  - requiresApp.js
- Gruntfile.js
- credits.json
- package.json
- properties.json

install/apache.conf

Ce fichier sera incorporé en tant qu'alias apache et définira le point d'entrée vers l'application, ci-dessous l'exemple à suivre de l'application FTTH, il suffira de remplacer le nom de l'application sur la première ligne `Alias /ftth[ENV]`

```
Alias /ftth[ENV] "[APPDIRECTORY]"
        <Directory "[APPDIRECTORY]">
                AllowOverride FileInfo
                Options FollowSymLinks
                Require all granted
        </Directory>
```

install/dependency.xml

Ce fichier est l'**un des plus importants** de l'application car il définit les modules à utiliser ainsi que leurs versions.

Il doit au minimum contenir le framework vitis, pour cela j'intègre le framework **en utilisant une branche spécifique à mon application app_ftth**.

Il est très important d'utiliser une branche spécifique pour ne pas avoir de conflits avec les autres applications : quand je push sur vitis je ne veux pas impacter l'application GTF du coup j'utilise la branche app_ftth que je merge avec master en temps voulu, de cette manière je contrôle les versions des modules utilisés et je n'empiète pas sur le travail des autres.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Application -->
<installer>
        <dependenciesCollection>
                <dependency>
                        <nature>framework</nature>
                        <name>vitis</name>
                        <version>app_ftth</version>
                </dependency>
        </dependenciesCollection>
</installer>
```

Une fois vitis déclaré il faudra faire de même pour les modules à utiliser dans mon application, on remarque que pour vitis j'ai utilisé *framework* en tant que nature, la seule différence sera d'utiliser *modules* pour les modules.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Application -->
<installer>
        <dependenciesCollection>
                <dependency>
                        <nature>framework</nature>
                        <name>vitis</name>
                        <version>app_ftth</version>
                </dependency>
                <dependency>
                        <nature>modules</nature>
                        <name>module_ftth</name>
                        <version>app_ftth</version>
                </dependency>
        </dependenciesCollection>
</installer>
```

Dans mon exemple FTTH je n'ai qu'in seul module ftth que j'incorpore à mon XML

install/fileToDelete.txt

Correspond aux fichiers à supprimer une fois l'application compilée, rien à modifier ici, j'utilise les lignes ci-dessous

```
package.json
Gruntfile.js
```

install/folderToDelete.txt

Correspond aux dossiers à supprimer une fois l'application compilée, rien à modifier ici, j'utilise les lignes ci-dessous

```
requires
```

less/variables.less

Ce fichier est le fichier déterminant le style de l'application : on y définit les couleurs à utiliser et les administrateurs pourront y rajouter leur CSS personnalisé.

Attention : ce fichier n'est pas écrasé lors des mises à jour avec VAI.

```
// user colors
@user-color-theme-1: #23282D;
@user-color-theme-2: #6F8456;


// Veremes colors
@veremes-gtf-color: #39468A;
@veremes-majic-color: #CF461F;
@veremes-qualigeo-color: #832C7A;
@veremes-wab-color: #198D9F;
/*@veremes-vmap-color: #CB0650;*/
@veremes-vmap-color: #198D9F;

@veremes-map-backgroud-color: #020237;
@veremes-gray-backgroud-color: #3A3A3A;
@veremes-light-gray-backgroud-color: #9A9A9A;


// Application colors
@application-color-theme: #39468A; // <- Change this color

@application-lighen-color-theme: lighten(@application-color-theme, 20%);
@application-darken-color-theme: darken(@application-color-theme, 10%);

@application-action-1-color: #337ab7; // blue bootstrap
@application-action-2-color: #5cb85c; // green bootstrap

// Other variables
@ui-grid-bg-image: "../images/ui-grid/wbg.gif";
@font-color-purple: #020237;

// Dimension des vignettes dans les listes.
@ui-grid-thumbnail-width: 100px;
@ui-grid-thumbnail-heigth: 60px;
```

requires/config.js

Permet de lister les fichiers des librairies externes à charger, rien à changer de ce côté, j'utilise les lignes ci-dessous

```
'use strict';

goog.provide('vitis.application.config');

// Fichiers js et css à charger pour l'application ftth.
var oApplicationFiles = {
    'css': [
        // Vitis
        'css/lib/bootstrap/css/bootstrap.min.css',
        'css/lib/jquery/plugins/bootstrap-datepicker/bootstrap-datepicker3.min.css',
        'css/lib/jquery/plugins/bootstrap-fileinput/css/fileinput.min.css',
        'css/lib/ui-grid/ui-grid.min.css',
        'css/lib/jquery/plugins/bootstrap-colorpicker/css/bootstrap-colorpicker.min.
↪css',
        'css/lib/bootstrap-checkbox/build.css',
        'css/lib/bootstrap-checkbox/font-awesome.css',
        'css/lib/jquery/plugins/bootstrap-slider/bootstrap-slider.min.css',
        'css/lib/jquery/plugins/malihu-custom-scrollbar/jquery.mCustomScrollbar.min.
↪css',
        'css/lib/ui-grid/plugins/draggable-rows.less',
        'less/main.less',
        // App
        'modules/vitis/less/main.less',
        'modules/ftth/less/main.less',
        'css/lib/codemirror/codemirror.css',
        'css/lib/codemirror/codemirror_foldgutter.css',
        'css/lib/codemirror/map.css',
        'css/lib/codemirror/show-hint.css',
        'css/lib/jquery/plugins/bootstrap-treeview/bootstrap-treeview.min.css',
        'css/lib/jquery/plugins/bootstrap-tagsinput/bootstrap-tagsinput.css',
        'css/lib/jquery/plugins/bootstrap-colorpicker/css/bootstrap-colorpicker.min.
↪css',
        'css/lib/viewer/viewer.min.css'
    ],
    'js': {
        'externs': [
            // Studio
            'javascript/externs/bootbox/bootbox.min.js',
            'javascript/externs/codemirror/codemirror.min.js',
            'javascript/externs/codemirror/htmlmixed.js',
            'javascript/externs/codemirror/css.js',
            'javascript/externs/codemirror/javascript.js',
            'javascript/externs/codemirror/clike.js',
            'javascript/externs/codemirror/php.js',
            'javascript/externs/codemirror/xml.js',
            'javascript/externs/codemirror/sql.js',
            'javascript/externs/codemirror/map.js',
            'javascript/externs/codemirror/show-hint.js',
            'javascript/externs/codemirror/addon/fold/foldcode.js',
            'javascript/externs/codemirror/addon/fold/foldgutter.js',
            'javascript/externs/codemirror/addon/fold/brace-fold.js',
            'javascript/externs/codemirror/addon/fold/xml-fold.js',
            'javascript/externs/angular/modules/ui-codemirror/ui-codemirror.min.js',
            'javascript/externs/tinymce/tinymce.min.js',
```

(continues on next page)

(continued from previous page)

```
            'javascript/externs/angular/modules/ui-tinymce/tinymce.js',
            'javascript/externs/jquery/plugins/bootstrap-treeview/bootstrap-treeview.
↪js',
            'javascript/externs/jquery/plugins/bootstrap-tagsinput/bootstrap-
↪tagsinput.min.js',
            'javascript/externs/scripts_cryptage.js',
            'javascript/externs/moment/moment.min.js',
            'javascript/externs/moment/min/moment-with-locales.min.js',
            'javascript/externs/jquery/plugins/notify/notify.js',
            'javascript/externs/jquery/plugins/bootstrap-colorpicker/bootstrap-
↪colorpicker.min.js',
            'javascript/externs/jquery/plugins/bootstrap-confirmation/bootstrap-
↪confirmation.min.js',
            'javascript/externs/jquery/plugins/bootstrap-datetimepicker/bootstrap-
↪datetimepicker.min.js',
            'javascript/externs/viewer/viewer.min.js'
        ]
    },
    'vitisModuleDependencies': ['ui.codemirror', 'ui.tinymce', 'ui.grid.draggable-rows
↪']
};
```

requires/requiresApp.js

Permet de lister les dépendances JS à charger avec closure, après `goog.provide('vitis.application.requires');` j'incorpore les scritps à inclure pour différents modules

```
/**
 * Ce fichier permet de charger les fichiers de l'applicaiton
 * de façon non compilée
 * @author: Armand Bahi
 */

goog.provide('vitis.application.requires');
goog.require('ftth.script_module');
```

Gruntfile.js

Ce fichier va permettre à closure de savoir quels fichiers utiliser pour la compilation, il faudra ajouter les dossiers contenant les différents modules (lignes 31 et 88) ainsi que modifier les noms des fichiers compilés par le nom de l'application (lignes 56, 62 et 94)

```
module.exports = function (grunt) {

    var compilerPackage = require('google-closure-compiler');
    compilerPackage.grunt(grunt);

    var devMode = grunt.option('dev');

    var sHome = '../';
    if (devMode) {
        sHome = '../client/';
    }
```

(continues on next page)

(continued from previous page)

```
    var sClosureDepsHome = '../../../../../';

    // Project configuration.
    grunt.initConfig({
        'closure-compiler': {
            ftth: {
                files: {
                    [sHome + '/javascript/ftth.min.js']: [
                        // Fichiers Vitis
                        sHome + 'javascript/require/*.js',
                        sHome + 'javascript/app/**/*.js',
                        sHome + 'javascript/externs/formReader/**/*.js',
                        sHome + 'javascript/externs/mapJSON/**/*.js',
                        sHome + 'modules/vitis/javascript/**/*.js',
                        // OpenLayers
                        sHome + 'javascript/externs/openLayers/**/*.js',
                        // Fichiers ftth
                        sHome + 'conf/requires/*.js',
                        sHome + 'modules/ftth/javascript/**/*.js'
                    ]
                },
                options: {
                    js: [
                        'node_modules/google-closure-library/closure/goog/**.js',
                        '!node_modules/google-closure-library/closure/goog/**_test.js'
                    ],
                    externs: [
                        'closure/externs/angular-1.3.js',
                        'closure/externs/bingmaps.js',
                        'closure/externs/jquery-1.9.js',
                        'closure/externs/bootstrap.js',
                        'closure/externs/geojson.js',
                        'closure/externs/jspdf.js',
                        'closure/externs/html2canvas.js'
                    ],
                    compilation_level: 'ADVANCED',
                    manage_closure_dependencies: true,
                    generate_exports: true,
                    angular_pass: true,
                    debug: false,
                    language_in: 'ECMASCRIPT5',
                    language_out: 'ECMASCRIPT5',
                    closure_entry_point: ['vitis'],
                    create_source_map: sHome + '/javascript/ftth.min.js.map',
                    output_wrapper: '(function(){\n%output%\n}).call(this)\n//#␣
↪sourceMappingURL=../javascript/ftth.min.js.map'
                }
            },
            formReader: {
                files: {
                    [sHome + '/javascript/externs/formReader/formReader.min.js']: [
                        // Fichiers FormReader
                        sHome + 'javascript/externs/formReader/**/*.js'
                    ]
                },
                options: {
```

(continues on next page)

(continued from previous page)

```
                    compilation_level: 'WHITESPACE_ONLY',
                    angular_pass: true,
                    language_in: 'ECMASCRIPT5',
                    language_out: 'ECMASCRIPT5'
                }
            }
        },
        'closureDepsWriter': {
            options: {
                depswriter: 'closure/depswriter/depswriter.py',
                root_with_prefix: [
                    // Fichiers Vitis
                    '"' + sHome + 'javascript/app ' + sClosureDepsHome + 'javascript/
↪app"',
                    '"' + sHome + 'modules/vitis/javascript ' + sClosureDepsHome +
↪'modules/vitis/javascript"',
                    '"' + sHome + 'javascript/externs/formReader ' + sClosureDepsHome␣
↪+ 'javascript/externs/formReader"',
                    '"' + sHome + 'javascript/externs/mapJSON ' + sClosureDepsHome +
↪'javascript/externs/mapJSON"',
                    '"' + sHome + 'conf/requires ' + sClosureDepsHome + 'conf/requires
↪"',
                    // OpenLayers
                    '"' + sHome + 'javascript/externs/openLayers ' + sClosureDepsHome␣
↪+ 'javascript/externs/openLayers"',
                    // Fichiers ftth
                    '"' + sHome + 'modules/ftth/javascript ' + sClosureDepsHome +
↪'modules/ftth/javascript"',
                    // Closure library
                    '"' + sHome + 'conf/node_modules/google-closure-library/closure/
↪goog ' + sClosureDepsHome + 'conf/node_modules/google-closure-library/closure/goog"'
                ]
            },
            targetName: {
                dest: sHome + '/javascript/ftth.deps.js'
            }
        }
    });

    grunt.loadNpmTasks('grunt-closure-tools');

    // Tache par défaut
    // cmd: grunt
    grunt.registerTask('default', ['closureDepsWriter', 'closure-compiler:ftth']);
    // cmd: grunt generate-deps
    grunt.registerTask('generate-deps', ['closureDepsWriter']);
    // cmd: grunt formReader minify
    grunt.registerTask('minify-formReader', ['closure-compiler:formReader']);
    // cmd: grunt formReader and studio minify
    grunt.registerTask('minify-libs', ['closure-compiler:formReader']);
    // cmd: grunt compile
    grunt.registerTask('compile', ['closure-compiler:ftth']);
};
```

credits.json

Crédits à intégrer dans l'application

```
[
    {
        "label": "CREDITS_SOFTWARE",
        "rows": [
            {
                "type": "text",
                "value": "FTTH"
            }
        ]
    },
    {
        "label": "CREDITS_VERSION",
        "rows": [
            {
                "type": "text",
                "value": "CREDITS_VERSION_CONTENT"
            }
        ]
    },
    {
        "label": "CREDITS_DATE",
        "rows": [
            {
                "type": "text",
                "value": "CREDITS_DATE_CONTENT"
            }
        ]
    },
    {
        "label": "CREDITS_EDITOR",
        "rows": [
            {
                "type": "link",
                "value": "Veremes",
                "url": "http://www.veremes.com"
            }
        ]
    },
    {
        "label": "CREDITS_AUTHOR",
        "rows": [
            {
                "type": "text",
                "value": "Armand Bahi"
            },
            {
                "type": "text",
                "value": "Alexandre Bizien"
            },
            {
                "type": "text",
                "value": "Anthony Borghi"
            },
            {
```

(continues on next page)

(continued from previous page)


```
                "type": "text",
                "value": "Frédéric Carretero"
            },
            {
                "type": "text",
                "value": "Olivier Gayte"
            },
            {
                "type": "text",
                "value": "Sébastien Legrand"
            },
            {
                "type": "text",
                "value": "Laurent Panabieres"
            },
            {
                "type": "text",
                "value": "Yoann Perollet"
            }
        ]
    },
    {
        "label": "CREDITS_GRAPHICS",
        "rows": [
            {
                "type": "link",
                "value": "Nematis",
                "url": "http://www.nematis.com"
            }
        ]
    },
    {
        "label": "CREDITS_LIBRARIES",
        "rows": [
            {
                "type": "link",
                "value": "Bootstrap",
                "url": "http://getbootstrap.com"
            },
            {
                "type": "link",
                "value": "AngularJS",
                "url": "https://angularjs.org"
            },
            {
                "type": "link",
                "value": "Google Closure",
                "url": "https://developers.google.com/closure"
            },
            {
                "type": "link",
                "value": "jQuery",
                "url": "http://jquery.com"
            },
            {
                "type": "link",
                "value": "Rgraph",
```


(continues on next page)

(continued from previous page)

```
                "url": "http://www.rgraph.net"
            },
            {
                "type": "link",
                "value": "CodeMirror",
                "url": "http://codemirror.net"
            },
            {
                "type": "link",
                "value": "OpenLayers",
                "url": "http://openlayers.org"
            },
            {
                "type": "link",
                "value": "TinyMCE",
                "url": "https://www.tinymce.com"
            }
        ]
    },
    {
        "label": "CREDITS_CONTACT",
        "rows": [
            {
                "type": "link",
                "value": "www.veremes.com",
                "url": "http://www.veremes.com"
            },
            {
                "type": "link",
                "value": "www.veremes.com/nous-contacter#1",
                "url": "http://www.veremes.com/nous-contacter#1"
            }
        ]
    },
    {
        "label": "CREDITS_LICENSE",
        "rows": [
            {
                "type": "link",
                "value": "Cecill-B",
                "url": "http://www.cecill.info/licences/Licence_CeCILL-B_V1-fr.html"
            }
        ]
    }
]
```

package.json

Fichier utilisé par npm lors de la phase de compilation, à utiliser comme ci-dessous

```
{
  "name": "Vitis",
  "version": "0.1.0"
}
```

properties.json

Properties client à intégrer

```
{
    "services_alias": "rest[ENV]",
    "web_server_name": "https://[hostname]:[PORT]/",
    "version": "[VERSION]",
        "build": "[BUILD]",
    "status": "unstable",
        "month_year": "[MONTH_YEAR]",
    "application": "extraction",
    "environment": "[ENV]"
}
```

Désormais notre application est viable et VAI saura la générer.

CHAPTER 2

AWS

2.1 Règles d'utilisation AWS

2.1.1 Type d'instance

On choisira toujours l'instance la plus petite puis on l'augmentera si besoin, on préfèrera augmenter le nombre plutôt que d'utiliser des grosses instances.

2.1.2 Tags

Chaque service AWS peut et doit avoir certains tags permettant en amont de répertorier, facturer et automatiser ces services.

Les tags suivants sont **obligatoires** :

- Tags administratifs :
  - **Name** : nom descriptif du service
  - **Stage** : environnement du service [dev, integration, test, recette, pre, prod]
  - **Author** : nom et prénom de la personne qui crée la machine
  - **Project** : nom du projet associé à l'élément
- Tags logiciels :
  - **isStarted** : démarre la machine tous les matins [true, false]
  - **isStopped** : arrête la machine tous les soirs [true, false]
  - **isTerminated** : supprime définitivement l'instance et les données qui s'y trouvent tous les soirs [true, false]

Le Tag suivant est **facultatif** :

- Tag logiciel :

- **isBackuped** : Réalise un instantané de l'EBS, 3 jours de rétention [true, false]

CHAPTER 3

Développement

3.1 Règles de développement Veremes

Ce document décrit les différentes règles qui doivent être respectées lors des développements sur les produits Veremes.

3.1.1 Sécurité

Les règles les plus importantes sont celles liées à la sécurité ; tout écart à ces directives pourrait créer des failles de sécurité et mettre en danger les exploitants des applications Veremes.

JavaScript

- **Utiliser ajaxRequest :** il est interdit d'utiliser des fonctions du type $.ajax, $.get, $.post (Jquery) ou $http (AngularJS). **Toutes les requêtes de type ajax doivent passer par la fonction ajaxRequest** qui va vérifier la bonne utilisation du token, écrire des logs, etc.
- **Ne jamais passer le token dans l'URL :** une personne mal intentionnée ayant accès au réseau d'un utilisateur d'application Veremes pourrait récupérer ce dernier en "sniffant" le réseau (communément appelé "man in the middle"). Pour éviter ceci, il faudra utiliser la fonction ajaxRequest.
- **Ne pas permettre les injections XSS :** une injection XSS, c'est quand un utilisateur saisit sur l'application du code HTML ou JavaScript qui sera interprété lors de l'utilisation de cette dernière. Exemple :

```
<img src=".." onload="$.post("http://...", {data: sessionToken})">
```

Le code ci-dessus pourrait envoyer les tokens de tous les utilisateurs à un serveur distant, y compris ceux des administrateurs ! Pour pallier à ceci, il faut utiliser AngularJS ou utiliser le code suivant :

```
sMaChaine.replace(/</g, "&lt;").replace(/>/g, "&gt;");
```

PHP

- **Ne jamais concaténer une variable dans une requête SQL :** ce type de comportement peut ouvrir des failles de type SQLi (injection SQL) qui font partie des failles les plus critiques. Pour pallier à ça **il faudra utiliser la fonction executeWithParams()** de la classe BD et passer les variables en JSON. Tout est documenté ici.
- **Utiliser les droits PostgreSQL :** la gestion de l'accès à une ressource se fera toujours sur deux niveaux (PHP et PostgreSQL). Si on se limite à PHP, une personne mal intentionnée pourrait, au travers d'une autre ressource, accéder à la table ; si on se limite à PostgreSQL, les erreurs remontées risquent de ne pas être bonnes.

3.1.2 Style PHP/JavaScript

Afin que le code soit cohérent, compréhensible et robuste, les développeurs doivent suivre certaines règles lors de leurs développements.

- **Indentation :** un code compréhensible est un code indenté ; pour cela, nous utilisons la fonction d'indentation automatique de Netbeans avec 4 espaces par indentation.
- **Ne pas dupliquer de code :** il ne faut jamais dupliquer du code car ce genre de comportement est très difficile à maintenir.
- **Utiliser les fonctions génériques :** il faut éviter de créer des fonctions chacun dans son coin et utiliser des fonctions génériques le plus souvent possible afin d'éviter les duplications de code.
- **Commentaires :** il faudra écrire des commentaires au moins **pour chaque fonction** sous le modèle de commentaires **JSDoc/PHPDoc**, et au moins un commentaire toutes les 8 lignes de code pour expliquer ce que l'on fait.
- **Nommage :**
  - Il faut éviter l'utilisation d'abréviations.
  - Chaque variable devra être précédée par son type (ex: aTableau, sChaine, iEntier, oObjet).
  - Les noms des classes commencent toujours par une majuscule.
  - Les fonctions et les variables (constantes non incluses) commencent toujours par une minuscule.
  - Les noms des constantes sont en majuscules.
  - Les fonctions et variables privées se terminent par un underscore.
  - Il faut utiliser d'abord la simple quote, puis la double quote.
- **Spécificités JavaScript :**
  - Utiliser **this_**: pour pointer sur l'objet en cours dans une fonction anonyme, il est conseillé de créer la variable this_ = this pour l'utiliser dans la fonction.
  - Utiliser **prototype** pour créer les fonctions.
  - Écrire **@export** où **@private** dans le commentaire d'une fonction pour spécifier si elle est publique ou privée lors de la compilation.

3.1.3 Modèle de données SQL

Procédure

Pour écrire ou modifier dans le modèle de données, il faut toujours respecter la procédure suivante :

1. Écrire dans Visual Paradigm

2. Faire valider par Olivier, Yoann ou Armand
3. Écrire le code SQL correspondant dans le fichier queries.xml en mentionnant son nom, la date et l'heure

Règles

- **Modifications :** il est interdit de faire des modifications sur les versions précédentes.
- **Suppressions :** il est interdit de supprimer une colonne, une table ou une vue ; il faudra créer une nouvelle si besoin et commenter l'ancienne en DEPRECATED.
- **Nommage:**
  - Il faut éviter l'utilisation d'abréviations.
  - Les noms des tables doivent être en minuscules.
  - Toute table contient un identifiant sous le modèle id_[nom de l'objet].
  - Les tables de relation s'écrivent [table1]_[table2].
  - Les tables constantes (non éditables depuis l'interface) s'écrivent rt_[table].
  - Les contraintes de type clés étrangères s'écrivent fk_[table 1]_[table 2].
  - Les contraintes de type clés primaires s'écrivent pk_[nom de la colonne].
  - Les colonnes géométriques doivent toujours être sous la contrainte du type de géométrie et de la projection.
  - Il ne faut pas utiliser d'underscores sauf dans les cas spécifiques cités ci-dessus.

3.2 Participer à la documentation Readthedocs

Au sein des équipes de Veremes, il arrive souvent que la documentation d'un produit ou d'un service se fasse par l'intermédiaire de Git et Readthedocs. Pour ce faire, il faut utiliser le format Markdown et écrire des fichiers qui seront publiés sur un dépôt Git ; ces derniers seront alors lus par Readthedocs, qui va publier la documentation au format sur le web HTML.

3.2.1 1. Les outils

Git

Pour commencer, il faudra au minimum installer Git et créer un compte sur le GitLab de Veremes. Ce qu'il faut comprendre de ces deux logiciels, c'est que **Git** s'installe sur un poste et permet de versionner des fichiers de manière à pouvoir voir quels ont été les derniers changements, revenir en arrière et faire plein d'autres choses très intéressantes ; quant à **GitLab**, c'est une plateforme web où on pourra visualiser et héberger tout ceci.

Atom

Il existe plusieurs logiciels permettant d'écrire au format Markdown ; en revanche, je conseille grandement Atom car il a été développé par les équipes de GitHub et qu'il permet une grande interaction avec Git ; de plus, en utilisant la commande ctrl + maj + M, on peut rapidement visionner ce que l'on écrit.

TortoiseGit

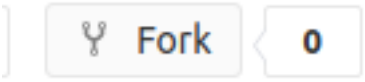
En option, si vous avez bien compris le fonctionnement de Git, il est intéressant d'utiliser TortoiseGit pour s'éviter de taper des lignes de commande quand on doit faire des choses compliquées.

3.2.2 2. Mise en place de l'environnement

Fork du projet

Si vous ne faites pas partie de l'équipe de développement de la documentation du produit, vous pouvez quand même effectuer des modifications puis demander à ce qu'elles soient appliquées. Si vous êtes membre du projet et que vous avez des droits en édition vous pouvez sauter cette étape.

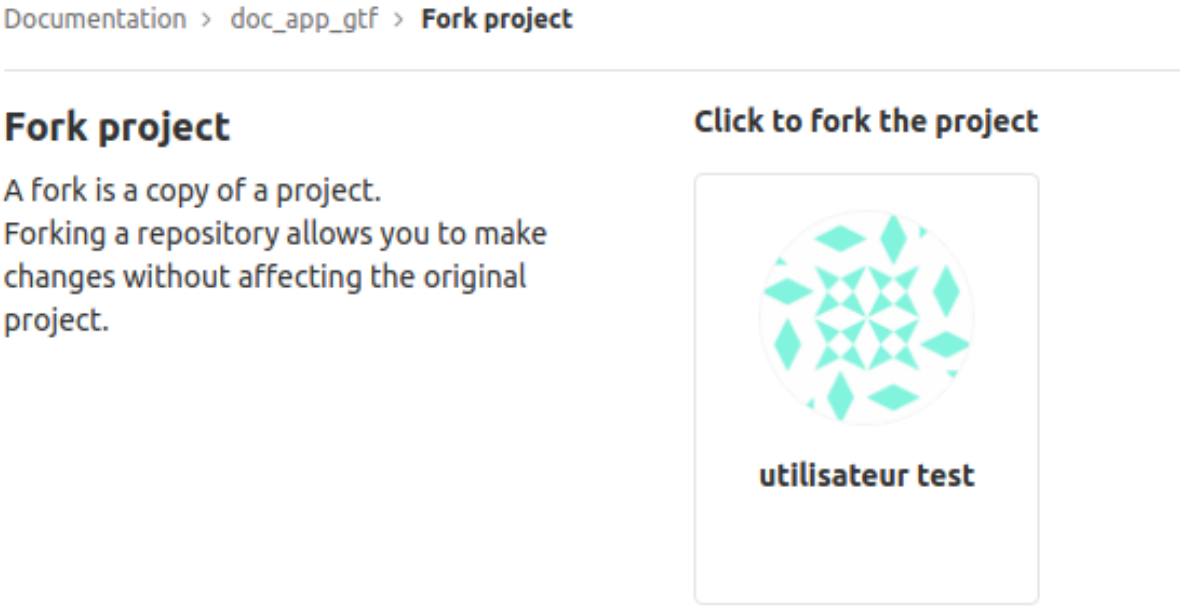
Pour cela il faudra faire un **Fork** (ou "fourche") du dépôt officiel vers votre compte en cliquant sur le bouton



Clone fork repo ou

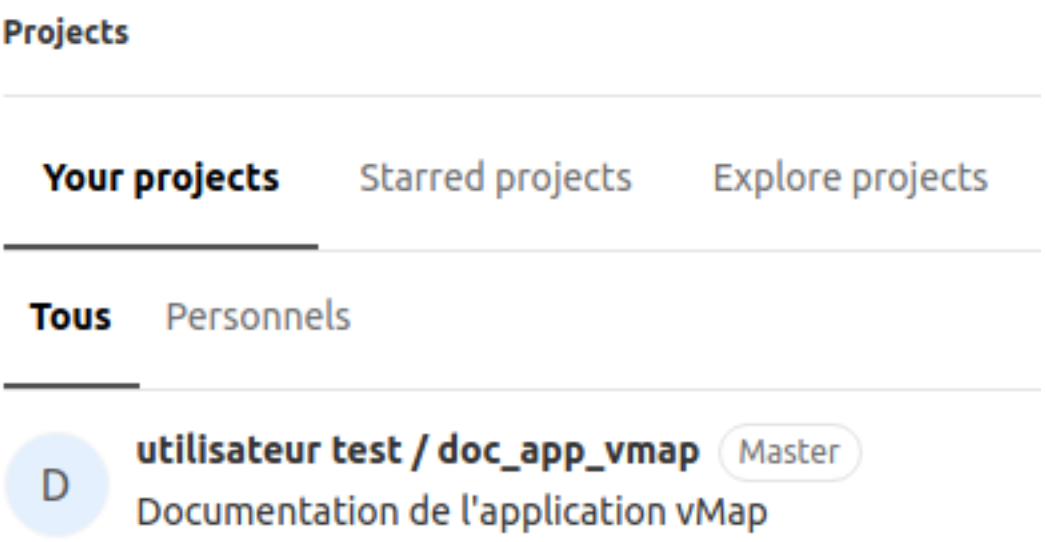
Clone fork repo fr

Maintenant GitLab vous demande où stocker le projet et c'est sur votre compte qu'il faudra le stocker.



Clone fork repo 2

Désormais le projet à été copié sur votre compte et vous avez tous les droits dessus car vous en êtes le propriétaire.



Clone

fork repo 3

Clone

Pour éditer et créer des fichiers de documentation, il faudra **clôner** le dépôt sur lequel vous voulez travailler localement, ceci va créer une copie de ce dépôt dans un dossier de votre ordinateur. Pour cela, rendez-vous sur la page GitLab de Veremes du dépôt sur lequel vous voulez travailler, puis copiez l'adresse qui apparaît.

Clone

gitlab repo

Créez ensuite un répertoire sur votre poste où vous souhaitez stocker les dépôts ; si vous êtes sous Windows, faites un clic droit puis “Open Git bash here” ; si vous êtes sous Linux, rendez-vous simplement dans ce dossier. Maintenant que vous êtes sur votre terminal git, lancez une à une les commandes suivantes en prenant soin de remplacer votre adresse mail, votre nom ainsi que l’url par celle que vous avez copié précédemment.

```
git config --global user.email "you@example.com"
```

```
git config --global user.name "Your Name"
```

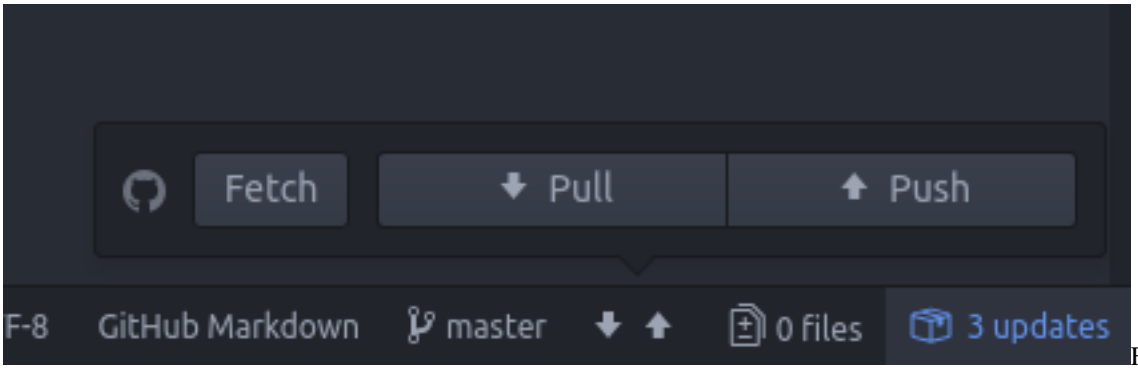
```
git clone http://vm09.veremes.net/Documentation/doc_app_vmap.git
```

Maintenant que vous avez rapatrié le dépôt chez vous, vous pouvez l’ouvrir avec l’éditeur Atom.

3.2.3 3. Travail sur le contenu des fichiers et envoi sur le serveur

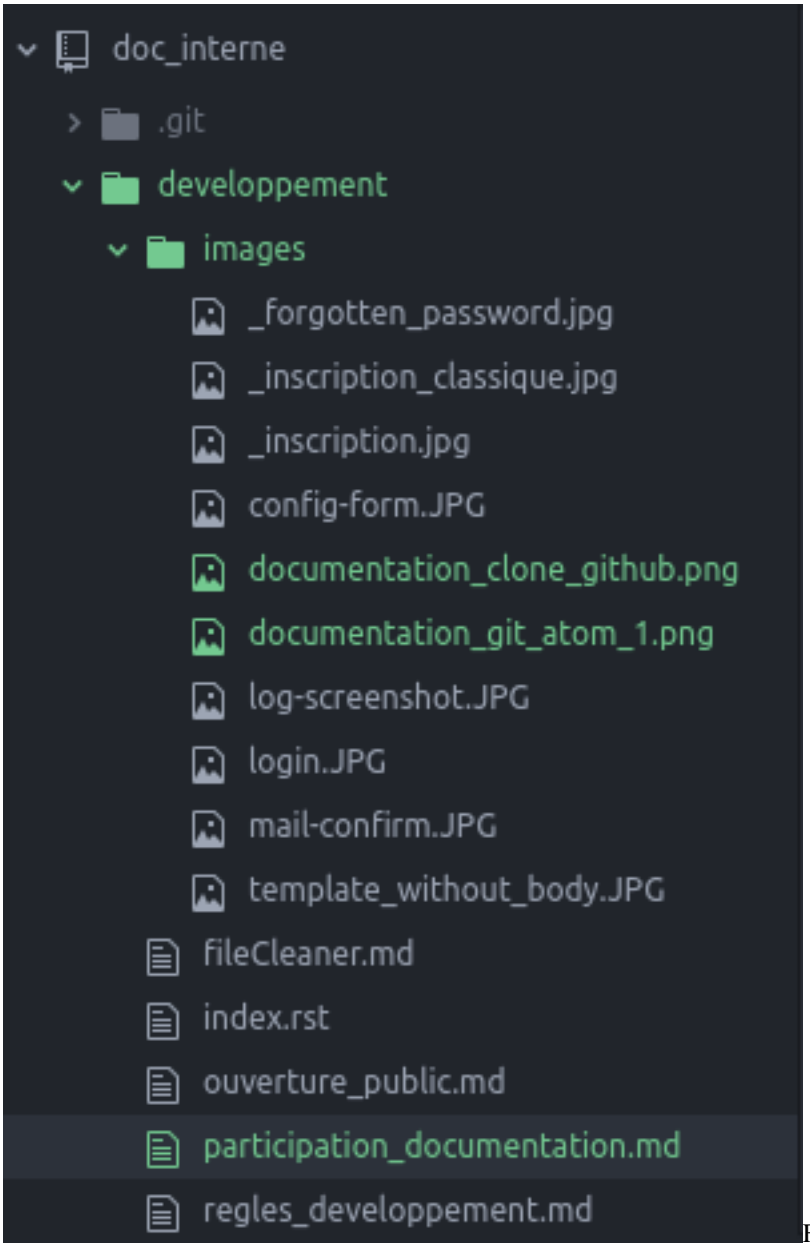
Désormais que votre environnement est mis en place et que vous avez ouvert Atom sur le dossier qui vous intéresse, vous pouvez faire vos modifications en respectant la norme Markdown ; une fois vos modifications terminées, il faudra les valider puis les envoyer sur le serveur.

Avant toute chose, il faut être sûr que la version du dépôt située sur votre poste est bien à jour avec celle du serveur : peut-être que quelqu’un a fait des modifications entre-temps. Pour cela, il y a un bouton dédié sur Atom nous permettant de faire un **Pull**, c’est-à-dire rapatrier les modifications en local.



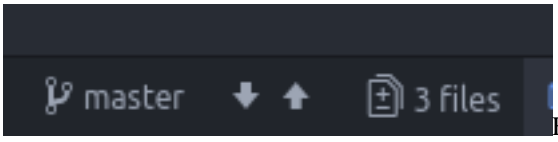
Fichiers modifiés git atom 0

Depuis Atom, vous avez peut-être remarqué que les fichiers ayant été modifiés ont changé de couleur depuis le bandeau de gauche :



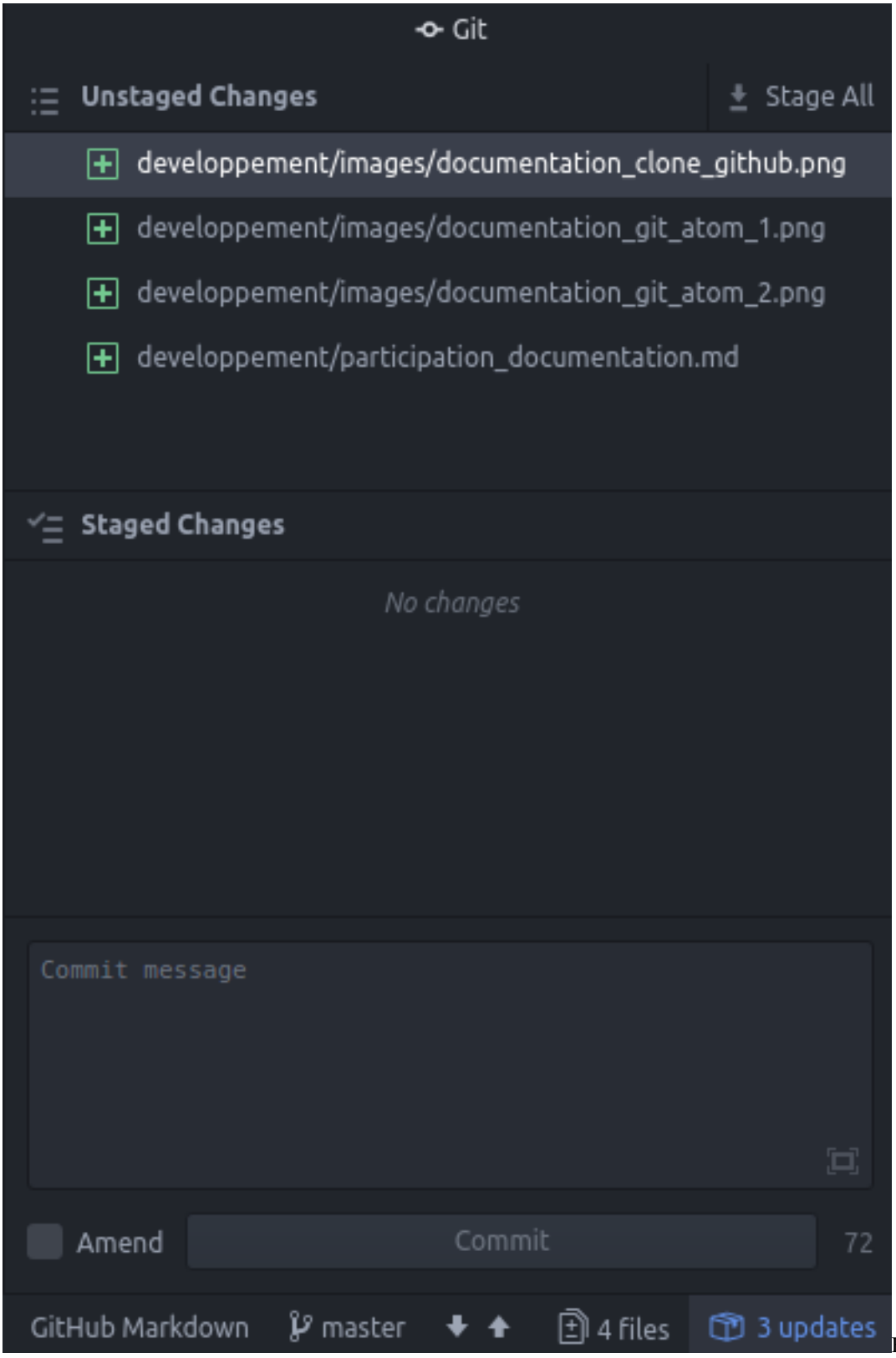
Fichiers modifiés git atom 1

Cela signifie que vous pouvez, quand vous le souhaitez, valider vos modifications en effectuant un **commit** ; pour cela, utilisez le bouton symbolisant les fichiers à commiter situé en bas à droite de l'écran.



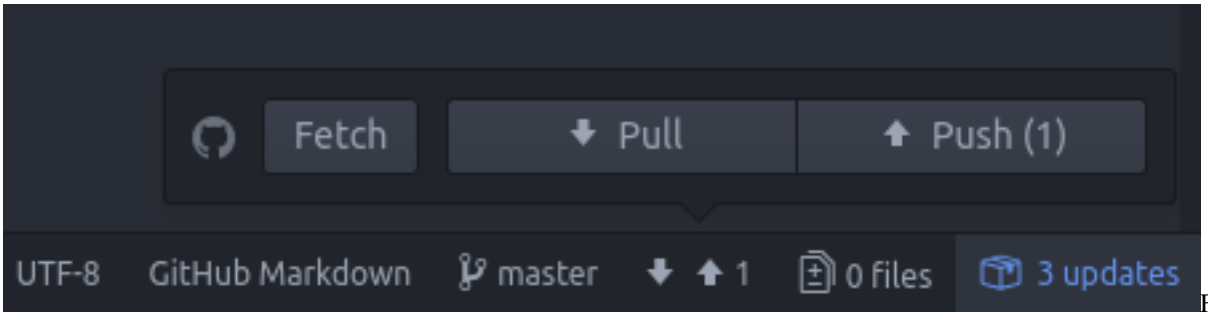
Fichiers modifiés git atom 2

Le menu suivant va apparaître sur l'écran ; depuis ce menu, vous pouvez visualiser les changements effectués en cliquant sur les fichiers et vous devrez placer les fichiers que vous voulez envoyer dans la partie **Staged changes** ; une fois que ceci est fait, il faudra écrire un commentaire décrivant les modifications que vous avez effectuées puis cliquer sur le bouton **Commit**.



Fichiers modifiés git atom 3

Maintenant que vous avez validé vos différentes modifications, il vous faudra les envoyer sur le serveur ; pour cela, utilisez le bouton **Push**.



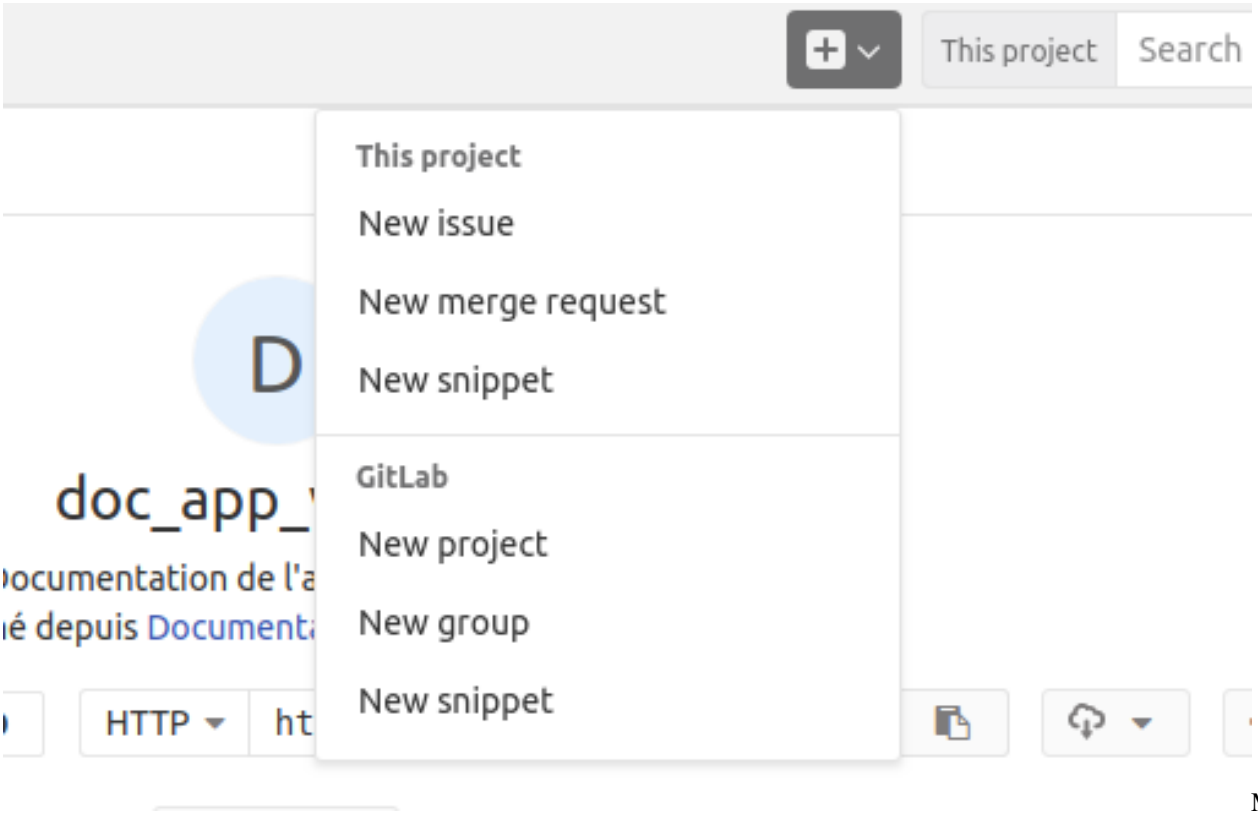
Fichiers modifiés git atom 4

Désormais, vos modifications sont directement visibles sur l'interface GitLab du projet.

Demande de Merge

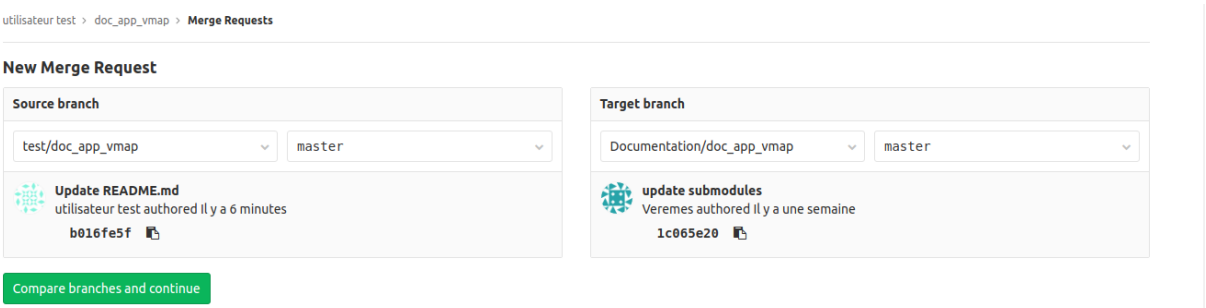
Si vous avez effectué un **Fork** du projet les modifications effectuées se situent sur le projet précédemment copié sous votre compte, pour que les modifications puissent être effectives sur le projet officiel, vous pouvez demander un **merge** aux administrateurs.

Pour cela cliquez sur le bouton *New merge request*:



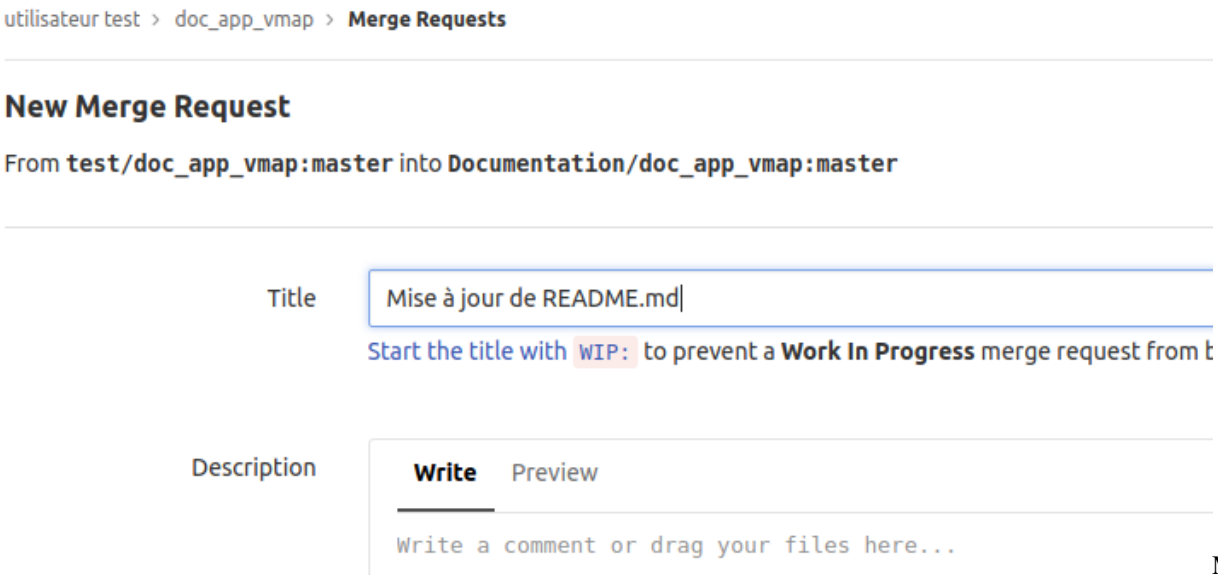
Merge request 1

Sélectionnez la source et la destination puis lancez la comparaison des branches. Sur cet exemple nous avions modifiés le fichier Readme.md il apparaît donc sur l'interface



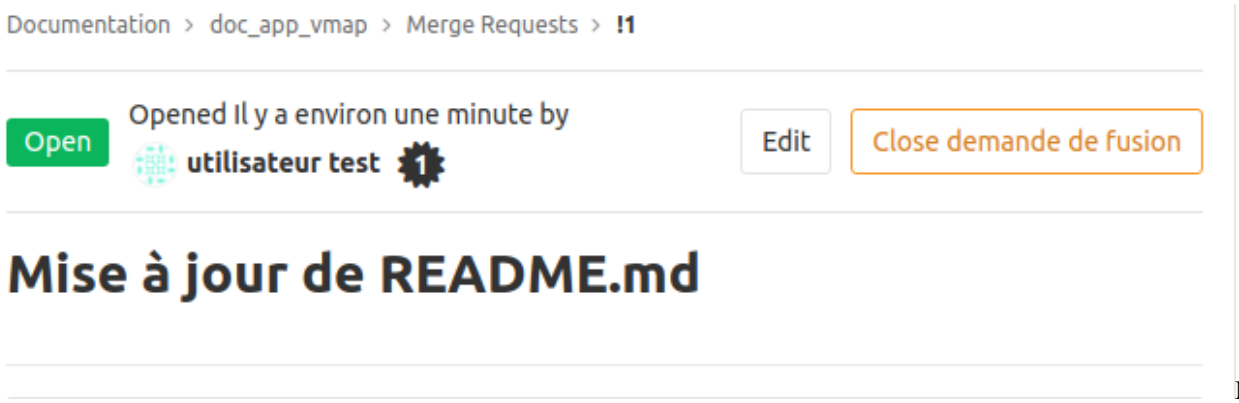
Merge request 2

Écrivez un titre à votre demande et cliquez sur le bouton *Submit demande de fusion*



Merge request 3

Votre merge a été demandé



Merge request 5

Fin

Désormais, vos modifications sont directement visibles sur l'interface GitLab du projet officiel, et si ce dernier est correctement lié à une page Readthedocs, il suffira de quelques minutes pour les voir apparaître sur la documentation

en ligne.

3.3 Utilisation du script de suppression

Ce document décrit le fonctionnement et la mise en place du script de suppression.

Ce script n'a pas vocation à faire le ménage à la place des développeurs, il doit être utilisé avec modération dans des cas comme le nettoyage des fichiers .map, ou du dossier "public".

Le script est disponible sur SVN et doit être déployé dans **VAS/util/vitis.fileCleaner**

3.3.1 Principe de fonctionnement

Le script est générique et n'a pas à être modifié pour nettoyer de nouveaux dossiers.

Le script va scanner un dossier de ressources contenant des fichiers JSON.

Chaque fichier JSON sera de cette forme :

```
{
    "remove":[
        {
            "path":"[VAS_DIR]/public/dtnet/dtdict",
            "expiration" : 4
        }
    ]
}
```

L'attribut **"path"** spécifie le chemin à nettoyer, il doit être relatif à un dossier de l'application en utilisant les clés définies plus loin dans ce document.

L'attribut **"expiration"** définit le nombre de jours nécessaires pour que le fichier soit considéré comme supprimable.

Il existe un fichier de ressource pour :

- **Vitis** :
  - **VAS/**upload (120 jours)
- **GTF** :
  - **VAS/**public/gtf (120 jours)
- **DtNet** :
  - **VAS/**public/dtnet/dtdict (4 jours)

Liste des clés de remplacement dans les chemins :

- **VAS_DIR** : répertoire VAS
- **PUBLIC_DIR** : répertoire Public (imp ASAP)
- **WS_DATA_DIR** : répertoire ws_data (imp ASAP)

En sortie d'exécution vous aurez un fichier de log disponible en consultation dans l'interface (**report_ANNEE-MOIS-JOUR**.log) de ce type :

Application/filecleaner/report_2017-12-10.log

```
10/12/2017 11:06:01    Lancement du script de nettoyage
10/12/2017 11:06:01    Récupération du fichier de ressource : dtnet.json
10/12/2017 11:06:01    Scan du dossier : /var/www/dtnet/vas/public/dtnet/dtdict (limite --> 4 jours)
10/12/2017 11:06:01    Récupération du fichier de ressource : gtf.json
10/12/2017 11:06:01    Scan du dossier : /var/www/dtnet/vas/public/gtf (limite --> 120 jours)
10/12/2017 11:06:01    suppression de l'élément /var/www/dtnet/vas/public/gtf/2017081108300252050 modifié pour la dernière fois il y a 121jours
10/12/2017 11:06:01    suppression de l'élément /var/www/dtnet/vas/public/gtf/2017081108310232792 modifié pour la dernière fois il y a 121jours
10/12/2017 11:06:01    suppression de l'élément /var/www/dtnet/vas/public/gtf/2017081108320215813 modifié pour la dernière fois il y a 121jours
10/12/2017 11:06:01    suppression de l'élément /var/www/dtnet/vas/public/gtf/2017081108330248554 modifié pour la dernière fois il y a 121jours
10/12/2017 11:06:01    suppression de l'élément /var/www/dtnet/vas/public/gtf/2017081108340271562 modifié pour la dernière fois il y a 121jours
10/12/2017 11:06:01    suppression de l'élément /var/www/dtnet/vas/public/gtf/2017081108350295991 modifié pour la dernière fois il y a 121jours
10/12/2017 11:06:01    suppression de l'élément /var/www/dtnet/vas/public/gtf/2017081108360282362 modifié pour la dernière fois il y a 121jours
10/12/2017 11:06:01    suppression de l'élément /var/www/dtnet/vas/public/gtf/2017081109240313431 modifié pour la dernière fois il y a 121jours
10/12/2017 11:06:01    suppression de l'élément /var/www/dtnet/vas/public/gtf/2017081109280222841 modifié pour la dernière fois il y a 121jours
10/12/2017 11:06:01    suppression de l'élément /var/www/dtnet/vas/public/gtf/2017081117210279 40 modifié pour la dernière fois il y a 121jours
10/12/2017 11:06:02    Récupération du fichier de ressource : vitis.json
10/12/2017 11:06:02    Scan du dossier : /var/www/dtnet/vas/upload (limite --> 120 jours)
10/12/2017 11:06:02    fin du script de nettoyage
```

log-screenschot

3.3.2 Mise en place du script

Le script devra être lancé via une tâche cron sous linux, ou une tâche planifiée sous windows (voir PyCron à définir).

Sous Linux

Vous devrez autoriser l'exécution du script par le système en utilisant les commandes suivantes dans le dossier **util/vitis.fileCleaner** :

```
chmod +x filecleaner.php
chown -R www-data.www-data ./
```

Crontab

Voici un exemple de crontab testé sous linux pour lancer le script tous les jours à 20H00 :

```
#<vitis.fileCleaner [VAS_DIR]/util/vitis.fileCleaner>
# Execution tous les jours à 20H00
0 20 * * * '[VAS_DIR]/server/php/bin/php' '[VAS_DIR]/util/vitis.fileCleaner/
↪fileCleaner.php'
#</vitis.fileCleaner>
```

3.4 Fonctionnalités d'inscription et de récupération de compte

Ce document décrit le fonctionnement et la mise en place des fonctionnalités d'inscription et de récupération de mots de passe.

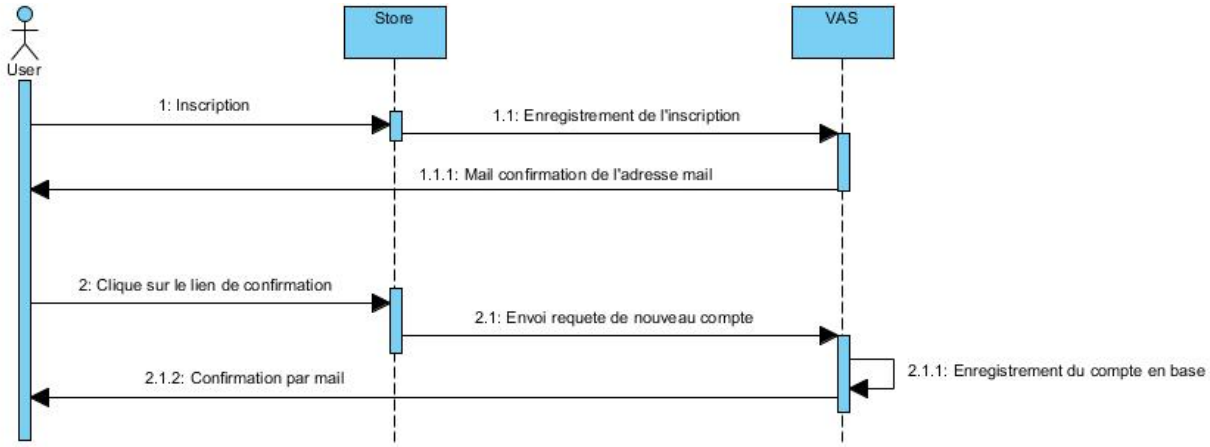
3.4.1 Fonctionnement

Nos applications peuvent désormais être ouvertes au public.

Il existe deux fonctionnements pour l'inscription :

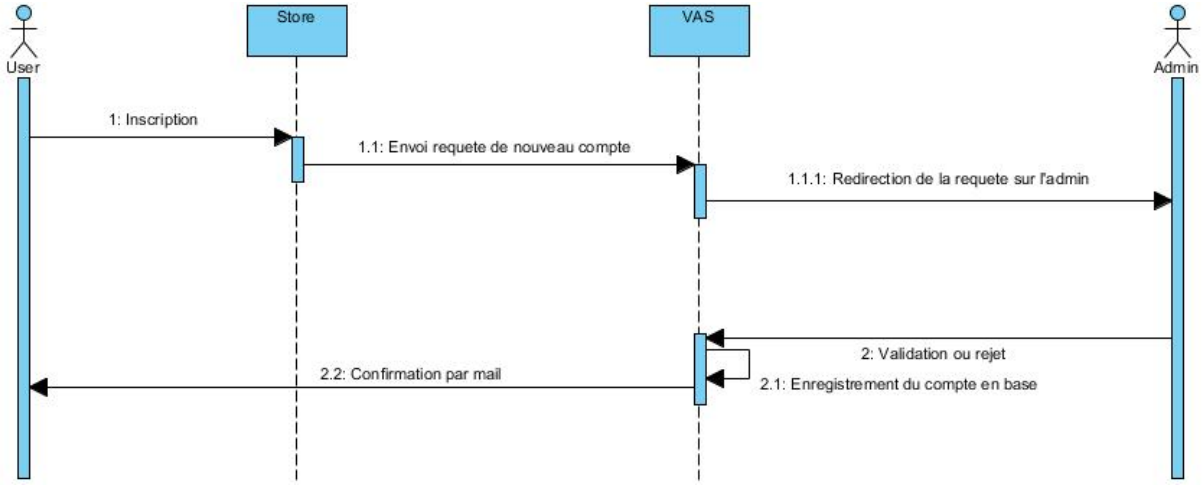
- **classique** : l'utilisateur s'inscrit et confirme lui-même son adresse mail.
- **contrôlé** : l'utilisateur s'inscrit et attend qu'un administrateur valide sa demande.

Fonctionnement classique



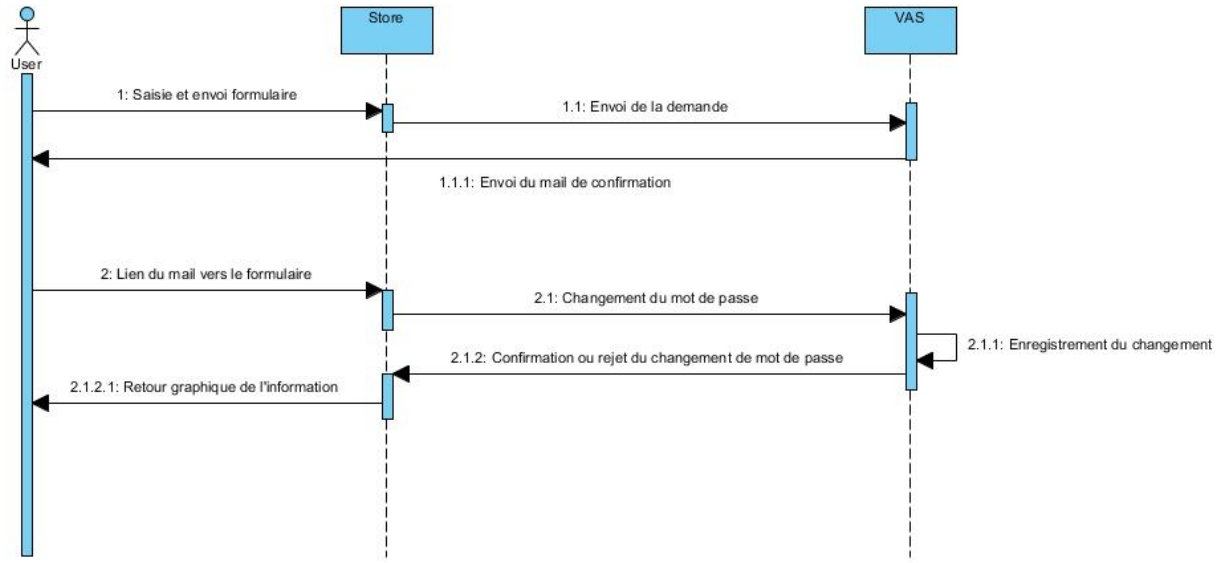
classique-sequence

Fonctionnement contrôlé



controle-sequence

Récupération de compte



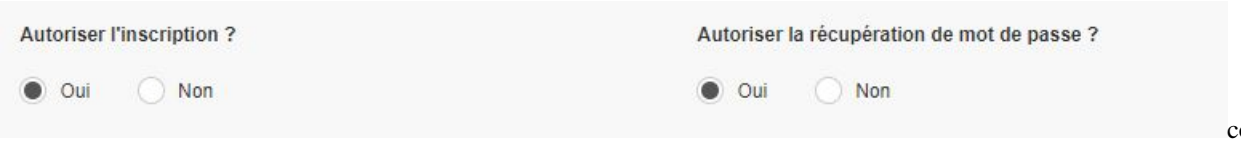
fpwd-confirm

3.4.2 Mise en place

L'activation de ces fonctions se fait via les **properties** de l'application.

Paramétrage dans properties.inc

Vous pouvez passer par l'interface et paramétrer cette partie via le formulaire (section **Configuration**)



config-form

Vous pouvez également intervenir sur le serveur en ajoutant/modifiant ces deux lignes dans le fichier **properties.inc** qui se trouve dans **VAS/rest/conf**

```
$properties["sign_up"] = "enabled";
$properties["password_forgotten"] = "enabled";
```

Paramétrage dans properties_server.inc

Pour paramétrer cette partie, vous serez obligé d'intervenir directement sur le serveur, en modifiant le fichier **properties_server.inc** qui se trouve dans **VAS/rest/conf**

Il vous faut maintenant configurer le compte qui créera les comptes dans votre base postgresSQL (assurez-vous que ce compte puisse créer des rôles de connexion) :

```
$properties['db_superuser_login'] = 'XXXX';
$properties['db_superuser_pass'] = 'XXXX';
```

Il vous faudra également configurer une connexion SMTP pour envoyer les mails (**ATTENTION** : ces properties existent aussi dans GTF ; pensez à les commenter pour que GTF utilise les properties de Vitis) :

```
$properties['mail_sender'] = 'XXXX@gmail.com';
$properties['nickname_sender'] = 'Veremes';
$properties['smtp_host'] = 'smtp.gmail.com';
$properties['smtp_port'] = 587;
$properties['smtp_authentification'] = true;
$properties['smtp_login'] = 'XXXX@gmail.com';
$properties['smtp_password'] = 'XXXX';
```

Il vous faut aussi définir le fonctionnement (classique ou contrôlé) :

```
$properties['automated_sign_up'] = true; // true pour 'classique' false pour 'controlé
↪'
```

Pour finir, vérifiez que vous avez bien ces properties :

```
//adresse mail de l'admin pour le fonctionnement controlé
$properties["admin_sign_up"] = "anthony.borghi@veremes.com";
//clé du captcha google
$properties["google_private_captcha"] = "6LdWLR8UAAAAAPE5wdl7hHJsmFxxrcmR5fZhApG2";
//liste des privilèges à donner aux nouveaux inscrits (vitis_user est donné par␣
↪défaut)
$properties['sign_up_automated_roles'] = array('vmap_user', 'gtf_user');
```

Modification de l'interface

Une fois la configuration terminée, l'interface de connexion de votre application doit afficher deux nouveaux boutons :

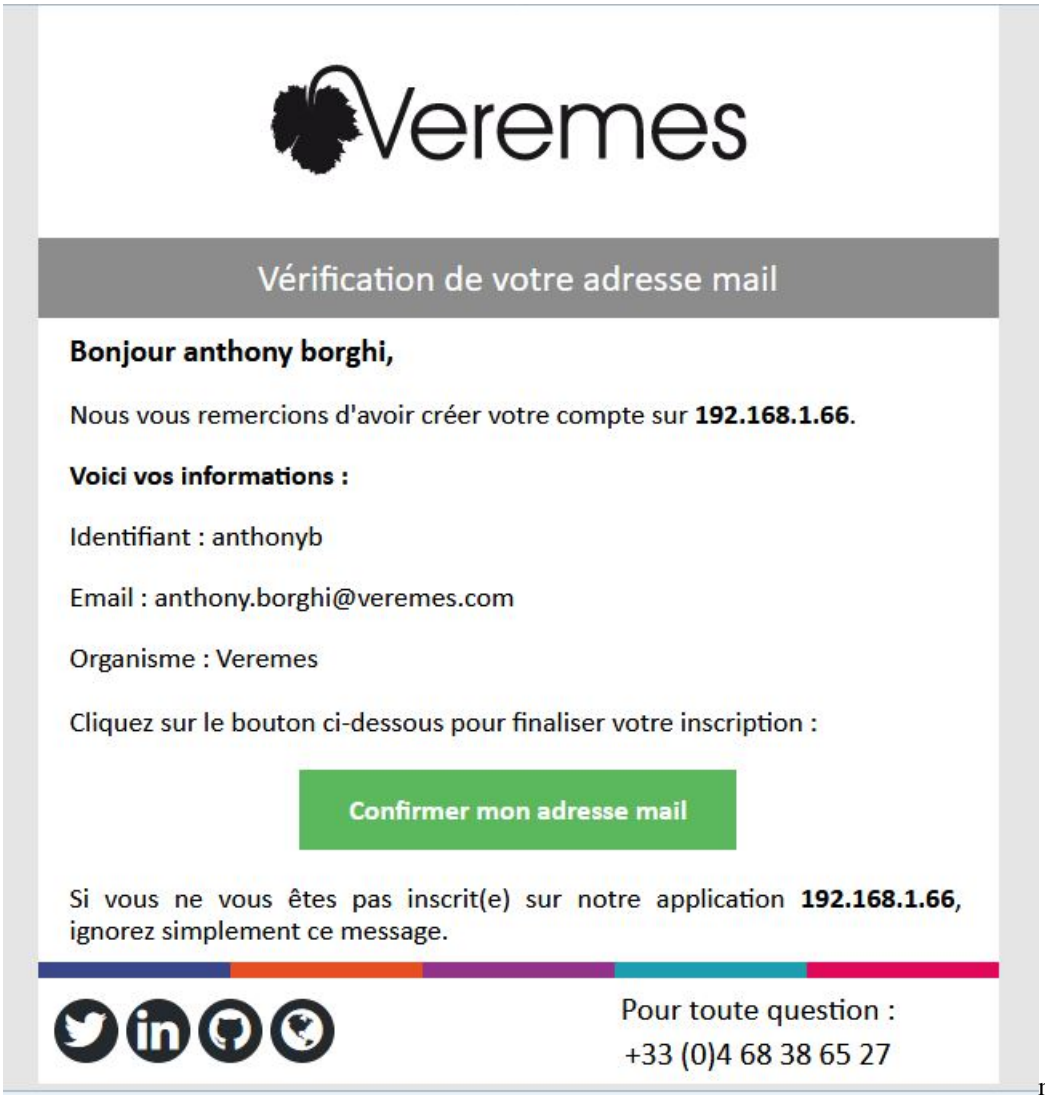
login-form

3.4.3 Personnalisation du template de mail

Personnaliser vos mails nécessite une certaine connaissance des langages HTML, Javascript, et PHP. Dans le cas où vous ne connaîtriez pas ces langages de programmation, nous pouvons le faire durant une prestation, qui vous sera facturée.

Lors de la mise à jour de votre application, ce fichier sera écrasé par la nouvelle version de notre template. Nous vous conseillons donc de garder une sauvegarde de votre travail en dehors de l'application.

Par défaut, l'application utilise ce template de mail :



mail-confirm

Pour personnaliser vos mails, il faut retoucher le fichier **Accounts.class.mail.inc**.

Dans ce fichier, vous trouverez un tableau contenant plusieurs clés :

- **newUserTitle** : Objet et titre du mail en mode 'contrôlé'.
- **newUserTitleAutomated** : Objet et titre du mail en mode 'classique'.
- **newUserBody** : Corps du mail d'inscription. Ce texte contiendra certaines données qui remplaceront les clés suivantes :
  - **TITLE** : Titre précédemment défini.
  - **HOSTNAME** : Nom du serveur récupéré depuis les properties.
  - **LOGIN** : Login de l'utilisateur ayant fait la demande.
  - **MAIL** : Adresse mail de l'utilisateur ayant fait la demande.
  - **COMPANY** : Entreprise de l'utilisateur ayant fait la demande.
  - **URL** : URL vers l'API rest pour permettre l'inscription (nécessaire au fonctionnement de la fonctionnalité).
- **confirmationSignUpTitle** : Objet du mail de confirmation d'inscription.

- **genericBody** : Mise en forme du corps. Il doit comporter une balise :
  - **MESSAGE** : Sera remplacé par le corps du mail souhaité (nécessaire au fonctionnement de la fonctionnalité).
- **SignUpOk** : Corps du mail de confirmation d'inscription, si l'inscription a réussi (Mode contrôlé, reçu par l'admin).
- **SignUpError** : Corps du mail de confirmation d'inscription, si l'inscription a échoué (Mode contrôlé, reçu par l'admin).
- **signUpConfirmation** : Corps du mail de confirmation d'inscription, si l'inscription a réussi (Mode contrôlé, reçu par l'utilisateur).
- **fpwdTitle** : Objet et titre du mail de récupération de compte.
- **fpwdUpdateOk** : Affiché si le mot de passe a pu être mis à jour.
- **fpwdUpdateError** : Affiché si le mot de passe n'a pas pu être mis à jour.
- **alreadyUpdate** : Affiché si l'utilisateur réutilise le même mail pour changer une deuxième fois son mot de passe.
- **fpwdBody** : Corps du mail envoyé à l'utilisateur pour changer son mot de passe. Ce texte contiendra certaines données qui remplaceront les clés suivantes :
  - **HOSTNAME** : Nom du serveur récupéré depuis les properties.
  - **LOGIN** : Login de l'utilisateur ayant fait la demande.
  - **URL** : URL vers l'API pour accéder au formulaire de récupération de mot de passe (nécessaire au fonctionnement de la fonctionnalité).
- **fpwdForm** : Contient le formulaire permettant le changement de mot de passe. Ce texte contiendra certaines données qui remplaceront les clés suivantes :
  - **URL** : URL vers l'API pour finaliser la récupération de mot de passe (nécessaire au fonctionnement de la fonctionnalité).
  - **TOKEN** : Remplacé par le token (token de la session contenant les informations de la demande de récupération de compte) dans le formulaire pour pouvoir l'envoyer au serveur lors de la validation du formulaire (nécessaire au fonctionnement de la fonctionnalité).
- **passwordScript** : Javascript utilisé pour la récupération de compte (vérification de concordance entre les deux mots de passe).
- **mainBody** : Corps principal du mail, sur l'image ci-dessus c'est le fond, ainsi que le logo Veremes en haut, et tout ce qui est sous le bandeau multicolore en bas. Ce template doit contenir certaines clés :
  - **CONTENT** : Remplacé par genericBody (nécessaire au fonctionnement de la fonctionnalité)
  - **SCRIPT** : Remplacé par le JavaScript (nécessaire au fonctionnement de la fonctionnalité)

3.5 Services web

3.5.1 1. Définition

Les web services sont la partie back-end de l'application ; ils se composent de plusieurs ressources qui permettent au client d'interroger la base de données, de lire/modifier des fichiers et d'effectuer des opérations sur la machine physique du serveur.

Dans vMap et autres produits Veremes, ils sont mis en place par une API-REST, ce qui signifie que l'on accède aux données selon des règles bien spécifiques.

Exemple de requête permettant de lister les cartes vMap :

```
https://corbieres/vmap_rest/vmap/maps
```

Exemple de requête permettant de voir les informations de la carte ayant pour identifiant '15' :

```
https://corbieres/vmap_rest/vmap/maps/{15}
```

L'API-REST retourne au client un résultat JSON ou XML contenant les informations demandées :


```
{
  "maps": [
    {
      "theme_name": "Thème Géobretagne",
      "theme_description": "Cartes Géobretagne avec fond OSM",
      "crs_name": "[EPSG:2154]-RGF93.Lambert-93",
      "map_id": 15,
      "crs_id": "EPSG:2154",
      "name": "Carte OSM Géobretagne",
      "description": "Carte Geobretagne avec un fond osm",
      "extent": "140807|6725441|303007|6799494",
      "catalog_index": 3,
      "thumbnail": "https://corbieres/vmap_ws_data/vitis/vmap_admin_map_vmap_admin_
↪map/documents/15/thumbnail/geobret.png?d=1499068782",
      "maptheme_id": 1,
      "groups": "",
      "groups_label": ""
    }
  ],
  "status": 1
}
```


3.5.2 2. Utilisation

2.1. En-têtes

Il y a divers en-têtes essentiels à l'utilisation des ressources.

2.1.1. Accept

Valeurs possibles :

- application/json
- application/xml
- application/x-vm-json

Définition

L'en-tête détermine le format de réponse demandé par le client. Les formats application/json et application/xml retournent un objet possédant un tableau qui porte le nom de la ressource (dans l'exemple ci-dessus, il s'agit de "maps"). Le format application/x-vm-json diffère en donnant comme nom du tableau "data", permettant de faire des requêtes génériques par le client.

2.1.2. Token

Le token de connexion identifie l'utilisateur de l'application, c'est grâce à lui que la ressource identifie si le demandeur possède les droits suffisants pour avoir un résultat, et c'est par son intermédiaire que se font les connexions à la base de données.

Pour des raisons de sécurité, il est strictement interdit de passer le token en tant que paramètre dans l'URL. Il faut donc le passer en en-tête : si une personne malveillante a accès au réseau (man in the middle), elle pourrait alors voir ce token et donc usurper l'identité d'un autre utilisateur.

2.1.3. X-HTTP-Method-Override

Lorsque l'on utilise régulièrement l'API-REST, il est possible que l'on soit confronté à des problèmes de longueur d'URL : à partir d'un certain nombre de caractères, les navigateurs refusent d'exécuter la requête et affichent l'erreur suivante :

```
414 URI Too Long
```

Pour pallier à cette contrainte, un en-tête X-HTTP-Method-Override a été mis en place pour envoyer une requête de type POST avec des paramètres figurant dans le body (sans limite de taille) et interprétables comme des requêtes GET :

```
General
    Request Method:POST

Request Headers
    X-HTTP-Method-Override: GET
```

2.2. Paramètres génériques

2.2.1. order_by

Permet de définir l'ordre d'affichage lorsqu'il y a plusieurs données. Par défaut, il vaut l'identifiant de la ressource.

2.2.2. sort_order

Couplé au paramètre "order_by", il permet de définir l'ordre avec les valeurs suivantes :

- asc : ordre ascendant
- desc : ordre descendant

2.2.3. limit

Si le paramètre “limit” est fourni, alors le tableau retourné se limite à “n” éléments.

2.2.4. offset

Souvent couplé avec les paramètres “limit” et “order_by”, il peut permettre, par exemple, d’effectuer une pagination sur une liste.

2.2.5. attributs

Définit les attributs qui seront retournés par le client. Pour les renseigner, il faut écrire ces attributs en les séparant par le caractère “|”.

2.2.6. distinct

True/false permet de distinguer les valeurs résultantes.

2.2.7. filter

Donne la possibilité à l’utilisateur de filtrer les données. Il faut écrire un objet JSON composé de **relations** et d’**opérateurs**.

2.2.7.1. Relations

Les relations définissent le type de condition à utiliser selon la structure JSON suivante :

```
{
    "relation": "AND",
    "operators":[{
        "..."
    }, {
        "..."
    }]
}
```

Dans cet exemple, on demande d’ajouter les filtres définis par les opérateurs selon la relation “AND”. On peut également utiliser une relation “OR”.

Il est aussi possible de faire dans une même requête du “AND” et du “OR” en incorporant une relation comme si c’était un opérateur :

```
{
    "relation": "AND",
    "operators":[{
        "..."
    }, {
        "relation": "OR",
        "operators": [{
            "..."
```

(continues on next page)

(continued from previous page)

```
        }, {
            "..."
        }]
    }]
}
```

Ainsi, on obtient une requête constituée de “AND” et de “OR” (voir l’exemple ci-après).

2.2.7.2. Opérateurs

Plus simples à comprendre, les opérateurs se composent de trois ou quatre arguments :

- **column** : nom de la colonne sur laquelle appliquer le filtre.
- **value** : valeur du filtre.
- **compare_operator** : type de comparaison (“=”, “!=”, “<>”, “>=”, “<=”, “>”, “<”, “IN”, “NOT IN”, “IS NULL”, “IS NOT NULL”, “LIKE”, “INTERSECT”).
- **compare_operator_options (optionnel)** : ajoute des options suivant le type de compare_operator.

La structure est la suivante :

```
{
    "column": "...",
    "compare_operator": "...",
    "value": "...",
    "compare_operator_options": {
        "..." : "..."
    }
}
```

2.2.7.3. Exemples

Pour être plus parlant, voici quelques exemples avec leur équivalent sous forme SQL.

En utilisant une relation AND, on peut filtrer sur plusieurs opérateurs :

```
{
    "relation": "AND",
    "operators":[{
        "column": "auteur",
        "compare_operator": "=",
        "value": "Laurent"
    }, {
        "column": "allume",
        "compare_operator": "=",
        "value": "true"
    }, {
        "column": "route_id",
        "compare_operator": "=",
        "value": 10
    }]
}
```

Équivalent SQL

```
auteur='laurent' AND allume='true' AND route_id=10
```

Si un seul opérateur est utilisé, alors il n'est pas nécessaire de renseigner de relation :

```
{
    "column":"auteur",
    "compare_operator":"=",
    "value":"laurent"
}
```

Équivalent SQL

```
auteur='laurent'
```

En utilisant des relations imbriquées, on peut effectuer des filtres complexes :

```
{
    "relation": "AND",
    "operators":[{
        "column":"auteur",
        "compare_operator":"=",
        "value":"laurent"
    }, {
        "relation": "OR",
        "operators": [{
            "column":"allume",
            "compare_operator":"=",
            "value":"true"
        }, {
            "column":"route_id",
            "compare_operator":"=",
            "value":10
        }]
    }]
}
```

Équivalent SQL

```
auteur='laurent' AND (allume='true' OR route_id=10)
```

On peut utiliser "compare_operator" = "IN" en utilisant des valeurs situées dans un tableau :

```
{
    "relation": "AND",
    "operators":[{
        "column":"auteur",
        "compare_operator":"=",
        "value":"laurent"
    }, {
        "relation": "OR",
        "operators": [{
            "column":"allume",
```

(continues on next page)

(continued from previous page)

```
            "compare_operator":"=",
            "value":"true"
        }, {
            "column":"route_id",
            "compare_operator":"IN",
            "value":[5,10]
        }]
    }]
}
```

Équivalent SQL

```
auteur='laurent' AND (allume='true' OR route_id IN (5, 10))
```

Il est possible d'utiliser "compare_operator" = "LIKE" avec des valeurs suivies ou précédées du caractère "%" :

```
{
    "column":"auteur",
    "compare_operator":"LIKE",
    "value":"laur%"
}
```

Équivalent SQL

```
auteur LIKE 'laur'%
```

En utilisant "compare_operator_options.case_insensitive" sur un type "LIKE", on peut rendre le filtre insensible à la casse :

```
{
    "column":"auteur",
    "compare_operator":"LIKE",
    "compare_operator_options":{
        "case_insensitive": true
    },
    "value":"%laur%"
}
```

Équivalent SQL

```
LOWER(auteur) LIKE LOWER('%lAur%')
```

Utilisation de "IS NOT NULL"

```
{
    "column": "nom",
    "compare_operator": "NOT NULL"
}
```

Équivalent SQL

```
nom IS NOT NULL
```

On peut effectuer des intersections géométriques utilisant PostGIS :

```
{
    "column":"geom",
    "compare_operator":"intersect",
    "value":"SRID=3857;POINT(349627.744690664 5237367.243157785)"
}
```

L'option "source_proj" utilisée ici n'est pas obligatoire mais conseillée si on connaît le système de projection de la table :

```
{
    "column":"geom",
    "compare_operator":"intersect",
    "compare_operator_options":{
        "source_proj": 2154
    },
    "value":"SRID=3857;POINT(349627.744690664 5237367.243157785)"
}
```

On peut utiliser un buffer lors de l'intersection, et même spécifier sur quel type de géométrie s'applique le buffer :

```
{
    "column":"geom",
    "compare_operator":"intersect",
    "compare_operator_options":{
        "source_proj":"2154",
        "intersect_buffer":8.31909066896183,
        "intersect_buffer_geom_type":"point|line"
    },
    "value":"SRID=3857;POINT(349643.2709620344 5237383.963757724)"
}
```

3.5.3 3. Exemple de création d'un web service et de ses ressources

Dans une installation classique, les web services se trouvent sous forme de dossiers dans le répertoire vmap/vas/rest/ws. Dans ces dossiers se trouvent les fichiers indispensables ainsi que les ressources des web services.

Dans cet exemple, nous allons créer un web service "customWS" dans lequel créer une ressource "villes".

3.1. Création du dossier et des fichiers indispensables

Parmi les fichiers indispensables se trouvent les fichiers suivants :

- **overview.phtml** : permet d'afficher la ressource dans la page d'aide au développement.
- **CustomWS.class.inc** : classe mère du projet.
- **CustomWS.class.sql.inc** : fichier contenant les requêtes SQL du projet. Il doit contenir au moins les requêtes "Définition des requêtes de l'api Vitis".

3.2. Création de la première ressource

Dans cet exemple, nous cherchons à créer la ressource “villes” qui permet de lister les villes contenues dans la table “f_villes_l93” installée par défaut avec vMap.

Chaque ressource est définie par deux fichiers PHP :

- l’un pour la définition unitaire d’un objet (ici Ville.class.inc),
- l’autre pour agir sur une liste complète d’objets (ici Villes.class.inc). Le “s” (obligatoire) permet de faire la différence entre la liste et l’unitaire.

3.2.1 La ressource unitaire (Ville.class.inc)

Il s’agit d’une classe PHP qui doit au moins contenir les éléments suivants :

3.2.1.1 Inclusions des fichiers

```
require_once 'CustomWS.class.inc';
require_once __DIR__ . '/../../class/vitis_lib/Connection.class.inc';
```

Inclusion de la classe mère du web service ainsi que la classe permettant d’effectuer des connexions à la base de données.

3.2.1.2 Classe

```
class Ville extends CustomWS {
    ...
}
```

Définition de la classe Ville.

3.2.1.3 Constructeur

```
/**
 * construct
 * @param type $aPath url of the request
 * @param type $aValues parameters of the request
 * @param type $properties properties
 * @param type $oConnection connection object
 */
function __construct($aPath, $aValues, $properties, $oConnection) {
    $this->aValues = $aValues;
    $this->aPath = $aPath;
    $this->aProperties = $properties;
    $this->oConnection = $oConnection;
    $this->aSelectedFields = Array(...);
}
```

Constructeur de la classe. La variable **$this->aSelectedFields** définit les attributs à afficher lors des requêtes.

3.2.1.4 Fonction GET

```
/**
 * @SWG\Get(path="/villes/{code}",
 *   tags={"villes"},
 *   summary="Get Ville",
 *   description="Request to get Ville by id",
 *   operationId="GET",
 *   produces={"application/xml", "application/json", "application/x-vm-json"},
 *   @SWG\Parameter(
 *     name="token",
 *     in="query",
 *     description="user token",
 *     required=true,
 *     type="string"
 *   ),
 *   @SWG\Parameter(
 *     name="code",
 *     in="path",
 *     description="",
 *     required=true,
 *     type="integer"
 *   ),
 *   @SWG\Response(
 *         response=200,
 *         description="Poprerties Response",
 *         @SWG\Schema(ref="#/definitions/villes")
 *     )
 *  )
 */

/**
 * get informations about villes
 */
function GET() {
    require $this->sRessourcesFile;
    $this->aFields = $this->getFields('sig', 'f_villes_l93', 'code');
}
```

Deux commentaires se trouvent au-dessus de cette fonction :

- le premier est utilisé par swagger pour générer la documentation en ligne interactive,
- le second est le commentaire de la fonction utilisée pour décrire aux développeurs ce que fait la fonction.

Les paramètres décrits dans les commentaires swagger passés dans le chemin l'URL par la relation in="path"(comme ici "*code*") sont disponibles via la variable **$this->aPath**.

Les paramètres décrits dans les commentaires swagger passés dans l'URL par la relation in="query" (comme ici "*token*") sont disponibles via la variable **$this->aValues**.

La ligne **require $this->sRessourcesFile** permet de récupérer le contenu du fichier *CustomWS.class.sql.inc*.

La fonction **$this->getFields** permet de récupérer, en base de données, les informations de la ville en question en utilisant le paramètre "*code*" passé dans l'URL.

Le résultat stocké dans **$this->aFields** est retourné lors de la requête http.

3.2.2 La ressource multiple (Villes.class.inc)

3.2.2.1 Inclusions des fichiers

```
require_once 'Vmap.class.inc';
require_once 'Ville.class.inc';
require_once __DIR__ . '/../../class/vitis_lib/Connection.class.inc';
require_once __DIR__ . '/../../class/vmlib/BdDataAccess.inc';
```

Require de la classe mère du web service ainsi que la classe unitaire et les fichiers permettant l'utilisation de la base de données.

3.2.2.2 Classe

```
class Villes extends CustomWS {
    ...
}
```

Définition de la classe Villes.

3.2.2.3 Constructeur

```
/**
 * construct
 * @param type $aPath url of the request
 * @param type $aValues parameters of the request
 * @param type $properties properties
 */
function __construct($aPath, $aValues, $properties) {
    $this->aValues = $aValues;
    $this->aPath = $aPath;
    $this->aProperties = $properties;
    $this->oConnection = new Connection($this->aValues, $this->aProperties);
    $this->aSelectedFields = Array(...);
}
```

Contrairement à la ressource unitaire, la connexion est instanciée.

3.2.2.4 Fonction GET

```
/**
 * @SWG\Get(path="/villes",
 *   tags={"Villes"},
 *   summary="Get Villes",
 *   description="Request to get Villes",
 *   operationId="GET",
 *   produces={"application/xml", "application/json", "application/x-vm-json"},
 *   @SWG\Parameter(
 *     name="token",
 *     in="query",
 *     description="user token",
```

(continues on next page)

(continued from previous page)

```
*     required=true,
*     type="string"
*   ),
* @SWG\Parameter(
*     name="order_by",
*     in="query",
*     description="list of ordering fields",
*     required=false,
*     type="string"
*   ),
* @SWG\Parameter(
*     name="sort_order",
*     in="query",
*     description="sort order",
*     required=false,
*     type="string"
*   ),
* @SWG\Parameter(
*     name="limit",
*     in="query",
*     description="number of element",
*     required=false,
*     type="integer",
*     default="4"
*   ),
* @SWG\Parameter(
*     name="offset",
*     in="query",
*     description="index of first element",
*     required=false,
*     type="string"
*   ),
* @SWG\Parameter(
*     name="attributs",
*     in="query",
*     description="list of attributs",
*     required=false,
*     type="string"
*   ),
* @SWG\Parameter(
*     name="filter",
*     in="query",
*     description="filter results",
*     required=false,
*     type="string"
*   ),
* @SWG\Parameter(
*     name="distinct",
*     in="query",
*     description="delete duplicates",
*     required=false,
*     type="boolean"
*   ),
*   @SWG\Response(
*         response=200,
*         description="Poprerties Response",
*         @SWG\Schema(ref="#/definitions/villes")
```

(continues on next page)

(continued from previous page)

```
 *     )
 *  )
 */

/**
 * get Villes
 * @return the array of objects
 */
function GET() {
    $aReturn = $this->genericGet('sig', 'f_villes_l93', 'code');
    return $aReturn['sMessage'];
}
```

Tous les paramètres génériques sont listés dans les commentaires swagger, et sont disponibles sur les variables ** $this->aPath ** et ** $this->aValues **.

Ici c'est la fonction **genericGet()** qui est utilisée et la fonction retourne du texte.

3.3. Ressource complexe avec executeWithParams()

Nous avons vu ci-dessus comment créer une ressource standard qui permet d'aller chercher en base de données les informations d'une table et de les renvoyer.

Imaginons que l'on veuille, dans la classe Ville, faire une deuxième requête en base de données (cette fois définie dans *CustomWS.class.sql.inc*) pour aller chercher les monuments associés à la ville.

CustomWS.class.sql.inc :

```
$aSql['getVilleMonuments'] = "SELECT * FROM sig.f_monuments WHERE \"code\"=[sCode]";
```

Ville.class.inc :

```
function GET() {
    require $this->sRessourcesFile;
    $this->aFields = $this->getFields('sig', 'f_villes_l93', 'code');

    $aSQLParams = array(
        'sCode' => array('value' => $this->aFields['code'], 'type' => 'string')
    );
    $oResult = $this->oConnection->oBd->executeWithParams($aSql['getVilleMonuments'],
↪$aSQLParams);
    if (gettype($oResult) == 'object') {
        $this->aFields['monuments'] = Array();
        while ($aLigne = $this->oConnection->oBd->ligneSuivante($oResult)) {
            array_push($this->aFields['monuments'], $aLigne);
        }
    }
}
```

Ci-dessus, la fonction **executeWithParams()** permet d'exécuter une requête SQL. Le résultat est alors rajouté dans $this->aFields['monuments'].

3.5.4 4. Fonction executeWithParams()

4.1 Définition

Pour effectuer des requêtes SQL en PHP, il est impératif d'utiliser la fonction executeWithParams() qui va exécuter une requête avec un tableau de paramètres passé en option.

Il ne faut surtout pas concaténer des variables à une requête SQL au risque d'exposer l'application à une faille de type SQLi.

Il faut écrire dans la requête une balise contenant le nom de la variable, et fournir un tableau de variables à executeWithParams().

Les différents formats sont :

- **string**, **number**, **integer** : pour les valeurs de variables à passer entre simple quotes.
- **group** : pour les valeurs à passer entre simple quotes et séparées par des virgules.
- **geometry** : pour les géométries à passer entre simple quotes.
- **quoted_string** : comme string mais pour intégrer des caractères spéciaux ; ex : 'ma lampe%'.
- **column_name**, **schema_name**, **table_name** : pour les noms de colonnes, tables, schémas. Attention car pour ces types de paramètres, executeWithParams() ne s'occupera pas des quotes, il faut donc les mettre à l'avance ; ex : SELECT "[column_name]" FROM [schema_name].[table_name] WHERE table='[table_name]'.

4.2 Exemples

```
$aSQLParams = array(
    'sSchema' => array('value' => $this->aProperties['schema_vmap'], 'type' =>
↪'column_name'),
    'sGroups' => array('value' => $sGroups, 'type' => 'group')
);
$sSql = "SELECT map_id, group_id FROM [sSchema].map_group WHERE \"group_id\" in␣
↪([sGroups])";
$oResult = $this->oConnection->oBd->executeWithParams($sSql, $aSQLParams);
```

```
$aSQLParams = array(
    'sSchema' => array('value' => $this->aProperties['schema_vmap'], 'type' =>
↪'column_name'),
    'sMapId' => array('value' => $map_id, 'type' => 'number')
);
$sSql = "SELECT * FROM [sSchema].map_layer WHERE \"map_id\" = [sMapId]";
$oResult = $this->oConnection->oBd->executeWithParams($sSql, $aSQLParams);
```

CHAPTER 4

FME