Travail pratique #4: Outils de Développement collaboratif

École Polytechnique de Montréal

Trimestre : Automne 2020

Équipier1: Sami Bourai *#2041659*
Équipier2: *Fanilotiana Guy Randriamanalinarivo#2040625*

Équipe:312

Présenté à : William Harvey et Nicolas Verbaere

Polytechnique Montréal
Remis le 17 novembre 2020

4.1 Gestion des bogue

1. Qu'est-ce que le logiciel Sia et à quoi sert-il ?

Sia est une plateforme de stockage cloud. Il sert a regrouper des hotes qui offre des services de stockage cloud.

2. Comment est déterminé le niveau d'importance d'un bogue (Issue) ?

Le niveau d'importance d'un bogue est déterminé par des étiquettes de priorités et des codes de couleurs. Ces priorités peuvent être vu dans la section Issue→Label

ex : High Priority, Hotfix, Low Priority.

3. Sur la barre latérale gauche du projet, sélectionnez la section Issues puis la colonne All,et entrez dans la barre de recherche l'id suivant :#3772 En quoi consiste le bogue lié à cet id ?

Le bogue concerne une impossibilité d'upload des fichiers, le téléchargement ralentit chaque 40mo puis se bloque totalement et du cote client toute requête TCP génère une erreur, Sia n'accepte plus des données de HTTP

4. Quel est le nom de l'utilisateur qui a rapporté ce bogue?

Robin Letourneur

5.Quel est le niveau d'importance de ce bogue ?

High Priority

6.Combien de personnes ont participé à la discussion reliée à ce bug ?

4 participants

7. Combien de Merge Requests sont reliés à ce bogue ? Combien ont été acceptés ?

2 Merge Requests sont reliés à ce bogue et les 2 ont été acceptes

8. Est-ce que la communauté a confirmé la validité du bogue ? Si oui, comment ?

Oui la communauté a confirmé la validité du bogue en ajoutant l'étiquette Bug

9. Est-ce que le bogue a été résolu ? Qu'est-ce qui vous l'indique ?

Non le bogue n'a pas encore été résolu puisque l'issue est encore en état « Open »

10. Refaire les étapes 3 à 9 inclusivement, mais pour l'Issue avecl'id suivant : #3969

10.3 Le bogue consiste en une erreur lorsque l'on télécharge un répertoire sur Skynet et l'utilisateur reçoit un message d'erreur lorsqu'il accède à Skylink comme quoi le format n'est pas spécifié et il n'y en a pas par défaut

10.4 Peter-Jan Brone

10.5 Le niveau d'importance de ce bogue est passé de Low Priority à High Priority

10.6 3 personnes ont participé à la discussion liée à ce bogue

10.7 1 Merge Request est relié à ce bogue et il fut accepté

10.8 Non, la communauté n'a pas validé ce bogue

10.9 Oui le bogue a été résolu, puisque le merge request a fait que le bogue soit « Closed »

4.2 Révision technique de code source

1. Que représente le projet open source Commento et à quoi sert-il ?

C'est une plate forme pouvant être intégrée à un site web et ainsi permettre aux utilisateurs de commenter ou encore de voter.

2. Dans un contexte de contribution à un projet, qu'est-ce qu'une révision (patch) ?

Le patch c'est un fichier contenant la correction qu'on veut apporter à notre fichier original.

3. Sur la barre latérale gauche du projet, sélectionnez la section Merge Requests puis la colonne All, et entrez dans la barre de recherche le nom suivant : WIP : Set cookies server-side. Décrivez brièvement ce que fait cette révision.

Cette révision apporte une correction aux Cookies du serveur. Selon le commentaire, le rédacteur supprime le code lié à la configuration, manipulation et lecture de Cookies.

4. Qui est l'auteur de cette révision ?

Michael Bryan

5. WIP signifie Work in Progress. Pourquoi l'auteur du commit a choisi de le préciser ?

Comme les corrections ne sont pas finies on indique avec WIP. Ainsi, d'autres utilisateurs peuvent apporter les changements necessaires s'ils le désirent.

6. Combien de fichier(s) a/ont subi des changements ? Dans quel(s) répertoire(s) se trouvaient ces fichiers ?

44 fichiers ont subi des changements.

Ces derniers se trouvent tous dans le répertoire api.

7. Quel est le statut actuel de la révision ? Que signifie-t-il ?

Open comme quoi ce dernier est encore en WIP.

8. Refaire les étapes 3 à 6 inclusivement, mais en entrant dans la barre de recherche le nom suivant : Fix twitter profile photo import bug

8.3 Il y a présence d'un bug lors de l'importation d'une photo de profil du réseau social Twitter.

8.4 **Souradip Mookerjee**

8.5 Comme le bug est fixed, il n'y pas l'etiquette WIP. Donc le bug est réparéé.

8.6 Un fichiers seulement (Commenter_photo.go) et ce dernier se trouve dans api aussi.

4.3 Intégration continue

1. Sur la barre latérale gauche du projet, sélectionnez la section Merge Requests puisla colonne All,et entrez dans la barre de recherche le nom suivant : Anti-Feature icons. Décrivez brièvement ce que fait cette révision
L'arrière-plan est animé par un effet Android par défaut, on aimerait améliorer l'animation

2. En cliquant sur l'onglet Pipelines, combien de commits passent les tests ? Combien échouent les tests ? Combien ont été annulés ?
4 commits passent les tests, 6 commits échouent les tests, 3 commits ont été annulés

3. En cliquant sur l'onglet Commits, combien de commits passent le pipeline ?
1 commit passe le pipeline

4. Quel est le rôle des pipelines sur Gitlab ?
Les pipelines définissent les actions à faire par étape, les actions d'une même étape se feront en parallèle, si elles réussissent toutes, on passe à l'étape suivante

5. Quel est le lien entre les tests et les pipelines sur Gitlab ?
Les tests sont habituellement dans les pipelines pour revérifier à chaque fois que les modifications apportées n'affectent pas ce qui fonctionnait déjà avant

6. Pourquoi les pipelines comportant plusieurs étapes (comme https://gitlab.com/uniqx/fdroidclient/pipelines/109834613) sont utiles pour l'intégration continue?
Pour s'assurer à chaque fois du bon fonctionnement avant la livraison ou le déploiement du projet après les modifications qui ont été apportées, et de suivre le programme si jamais il y a de nouveaux bogues on pourra mieux les repérer

Pour s'assurer à chaque fois de livrer un programme de qualité
Pour gagner du temps puisque les pipelines exécutent les taches dans l'ordre indiqué et on n'a pas forcément besoin de réécrire les étapes à chaque fois
Dans l'exemple, on a 3 tests à passer avant de faire le déploiement

Partie 2 :

Les deux équipe :

2)- Quelle(s) commande(s) avez-vous utilisée ?
Git clone https://gitlab.com/polytechnique-montreal-log1000/20203/311-312

Equipe 1 :

4. Ajoutez vos modifications à l'entrepôt Git. Quelles commandes avez-vous utilisée ?

Les modifications ont été ajoutées et nous avons utilisé les mêmes commandes que d'habitude.

Git checkout equipe2 -> pour aller sur notre branche.
Git add .
Git commit -m " "
Git push

1. Qu'est-ce qu'un Merge Request?

Une demande de fusion dans un travail de collaboration sur gitlab.

2. Quelle est l'utilité d'un Merge Request? Cliquez sur « New Merge Request », afin de créer un nouveau Merge Request.
L'utilité de ce dernier est de faire en sorte que les autres membres de l'équipe valide les modifications ajoutées et quelle soient ensuite ajoutées au master.
3. Quelles sont les branches source et destination de votre Merge Request ?

La branche source : equipe2 pour notre cas.

La branche destination : master

Equipe 2 :
1. Quel(s) est/sont les commits associées ?

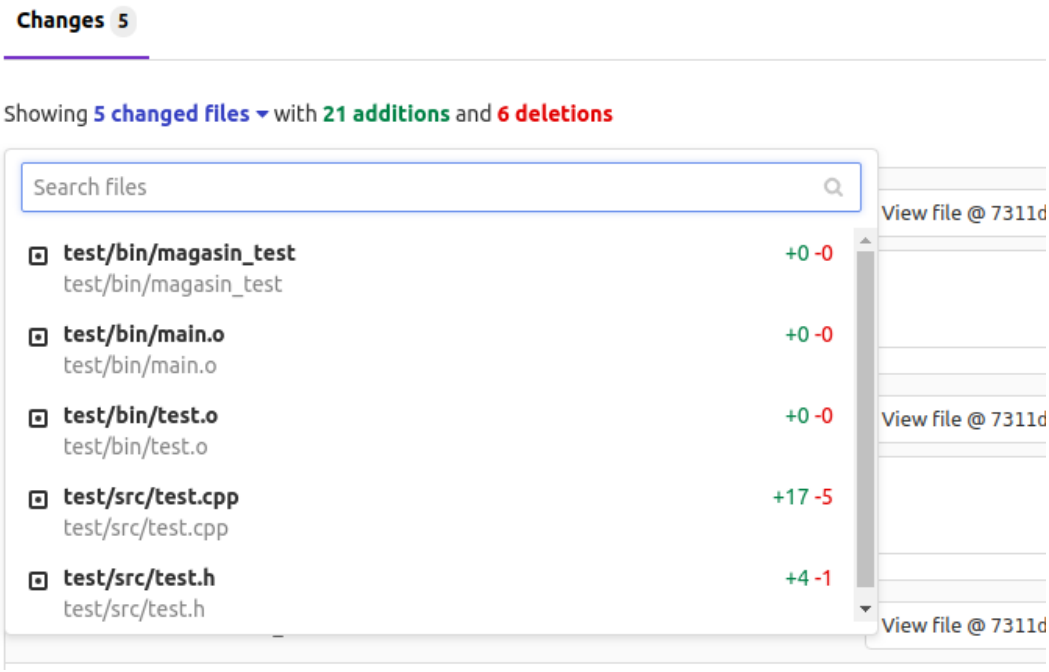
L'équipe 311 à fait deux commits

le premier étant : Ajout Makefile

le deuxième étant : ajout surcharge opérateur = pour la classe Client et 1 test unitaire

2. Quel(s) est/sont les fichiers qui ont été modifiés ?

Il y a 14 fichiers qui ont été modifiés pour le premier commit. Les voici :
Makefile Client.cpp Panier.cpp Client.o Panier.o Produit.o Rayon.o magasin magasin_test main.o test.o main.cpp test.cpp test.h
Il y a 5 fichiers qui ont été modifiés pour le deuxième commit. Les voici :



3. Est-ce que les modifications demandées sont correctes (tests unitaires fonctionnement et passent + modification du code) ?

Voici les 2 tests intégré par l'équipe 1 :

```
class Test : public CppUnit::TestFixture
{
    CPPUNIT_TEST_SUITE(Test);
    CPPUNIT_TEST(identifiantChangeTest);
    CPPUNIT_TEST(testSurchargeOperateurEgal);
    CPPUNIT_TEST_SUITE_END();
```

Ainsi les deux tests unitaires intégrés passent, en voici la preuve :

```
rm -rf bin/magasin bin/*.o test/bin/*.o test/bin/magasin_test
→  311-312 git:(equipe1) ✗ make
mkdir -p bin
g++ -o bin/main.o -c src/main.cpp
g++ -o bin/Client.o -c src/Client.cpp
g++ -o bin/Panier.o -c src/Panier.cpp
g++ -o bin/Produit.o -c src/Produit.cpp
g++ -o bin/Rayon.o -c src/Rayon.cpp
g++ -o bin/magasin bin/main.o bin/Client.o bin/Panier.o bin/Produit.o bin/Rayon.o
→  311-312 git:(equipe1) ✗ make test
mkdir -p test/bin
g++ -o test/bin/main.o -c test/src/main.cpp
g++ -o test/bin/test.o -c test/src/test.cpp
g++ -o test/bin/Client.o -c src/Client.cpp
g++ -o test/bin/Panier.o -c src/Panier.cpp
g++ -o test/bin/Produit.o -c src/Produit.cpp
g++ -o test/bin/Rayon.o -c src/Rayon.cpp
g++ -o test/bin/magasin_test test/bin/main.o test/bin/test.o test/bin/Client.o test/bin/Panier.o test/bin/Produit.o test/bin/Rayon.o -lcppunit
./test/bin/magasin_test
..


OK (2 tests)


→  311-312 git:(equipe1) ✗
```

4. Est-ce qu'il y a des objets (. o) et/ou l'exécutable du projet présents sur le Git? Est-ce que c'est une bonne ou mauvaise pratique de mettre ces fichiers dans un entrepôt Git ?

Effectivement, il y a bel et bien des fichier .o et des exécutables notamment le fichier magasin et magasin_test

c'est plutôt une mauvaise pratique de mettre des fichiers .o dans git puisque ces derniers seront recrées par la compilation. La bonne pratique étant plutôt de faire un gitignore pour ces fichiers là.

5. Attribuez un score aux modifications et justifiez votre choix.

On attribut un score de 5/5 aux modifications puisque ces dernières ont répondu aux exigences.