



# Le Versionning

## Table des matières

I-Introduction.....	2
II-GitHub .....	3
III-Tortoise.....	4
IV-Déposer un projet .....	5
V-Partager un projet et travailler en commun .....	7
VI-Conclusion.....	9

## I-Introduction

Le versioning permet, grâce à un système de clone (copie du projet sur un réseau local ou sur le cloud), d'avancer simultanément sur différentes tâches. Il permet également d'archiver un ensemble de fichiers en conservant la chronologie de toutes les modifications qui ont été effectuées dessus. De cette façon, il n'y a aucun risque d'altérer ou de perdre les données déjà enregistrées sur le dépôt commun (ou *repository*).

Les développeurs peuvent avoir accès à tout ou partie du projet commun, en fonction de la sensibilité des données qui y sont traitées.

Certains logiciels anticipent même les conflits : ils avertissent l'utilisateur qu'un autre développeur travaille en même temps que lui sur la même ligne de code et évitent ainsi que les deux travaux se parasitent.

Le versioning est utilisé tout au long du cycle de vie d'un logiciel, le long des grandes étapes que sont la maquette, le prototype, la version alpha, la version bêta, la release candidate et la version finale.

## II-GitHub

Git est un logiciel de gestion de version. Git va nous permettre d'enregistrer les différentes modifications effectuées sur un projet et de pouvoir retourner à une version précédente du projet.

Dans le langage des systèmes de gestion de version, la copie de l'intégralité des fichiers d'un projet et de leur version située sur le serveur central est appelé un dépôt. Git appelle également cela "repository" ou "repo" en abrégé.

GitHub est un service en ligne qui permet d'héberger des dépôts ou repo Git. C'est le plus grand hébergeur de dépôts Git du monde.

Une grande partie des dépôts hébergés sur GitHub sont publics, ce qui signifie que n'importe qui peut télécharger le code de ces dépôts et contribuer à leur développement en proposant de nouvelles fonctionnalités.

Pour récapituler, et afin d'être bien clair sur ce point : Git est un logiciel de gestion de version tandis que GitHub est un service en ligne d'hébergement de dépôts Git qui fait office de serveur central pour ces dépôts.

### III-Tortoise

TortoiseSVN est un client Windows open source gratuit pour le système de contrôle de version *Apache™ Subversion®*. TortoiseSVN gère des fichiers et répertoires dans le temps. Les fichiers sont stockés dans un *dépôt* central. Le dépôt est un peu comme un serveur de fichiers ordinaire, sauf qu'il se rappelle chaque changement fait à vos fichiers et répertoires.

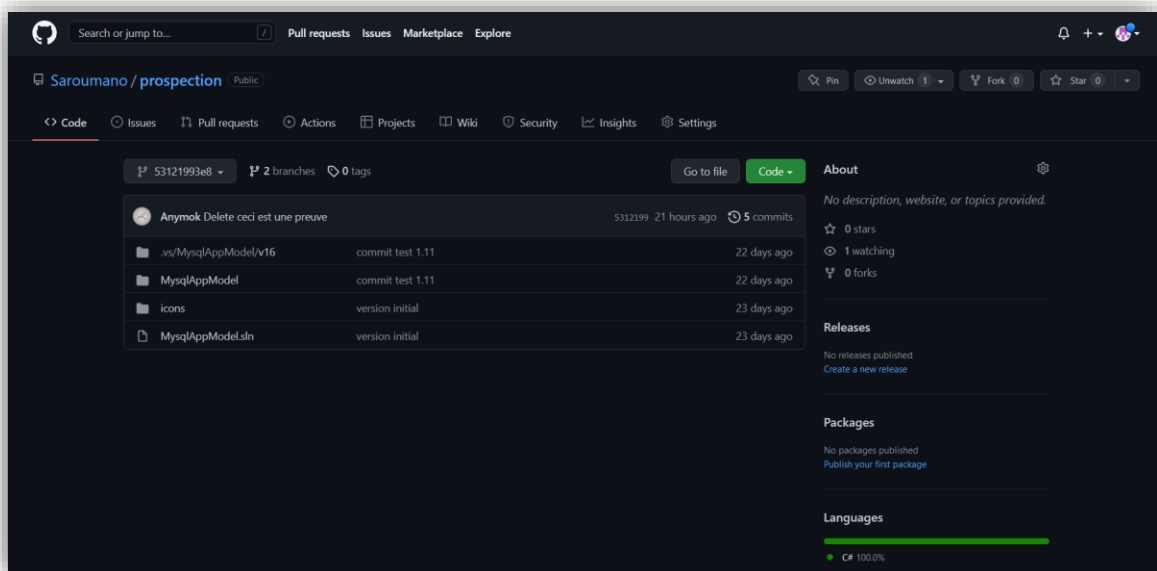
Cela vous permet de récupérer les anciennes versions de vos fichiers et examiner l'histoire de comment et quand vos données ont changé, et qui les a changés. C'est pourquoi beaucoup de gens voient Subversion et des systèmes de contrôle de versions en général, comme d'une sorte de « machine à remonter le temps ».

Quelques systèmes de contrôle de version sont aussi des systèmes de gestion de configuration logicielle (GCL). Ces systèmes sont spécifiquement conçus pour gérer des arborescences de code source et ont beaucoup de fonctionnalités spécifiques au développement de logiciel - comme la compréhension de langages de programmation en natif, ou des outils d'approvisionnement pour construire le logiciel. Subversion, cependant, n'est pas un de ces systèmes ; c'est un système général qui peut être utilisé pour gérer *n'importe quelle* collection de fichiers, y compris du code source.

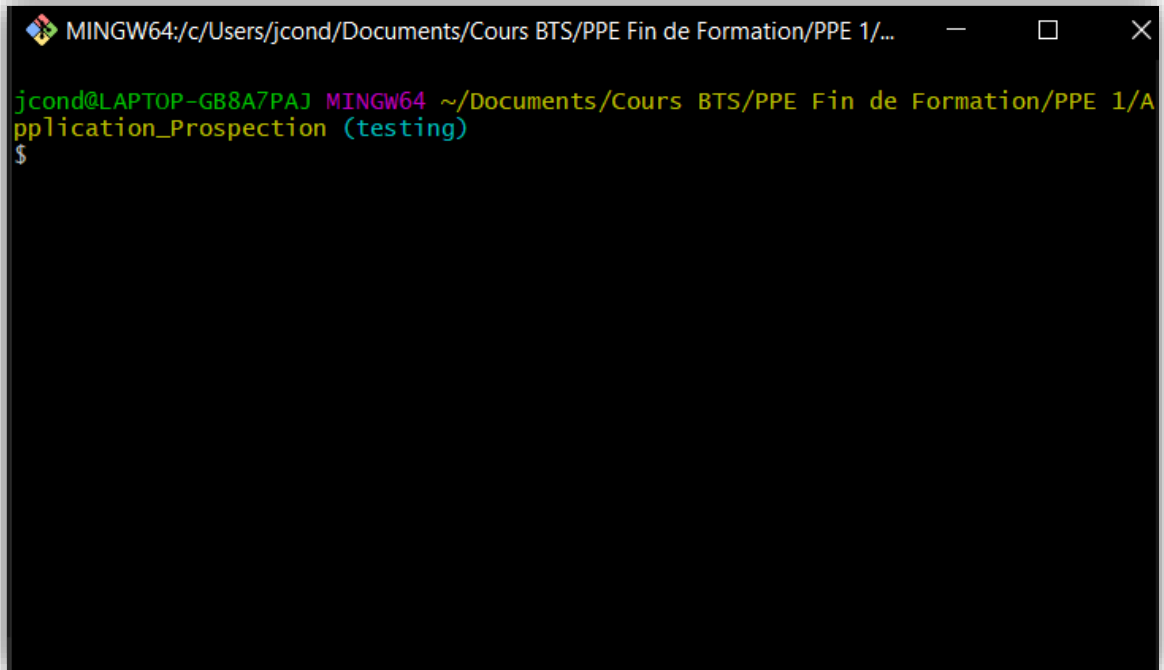
## IV-Déposer un projet

On va ici prendre en main GitHub, sur lequel on va tout d'abord déposer un projet complet pas à pas puis dans un second temps, nous partagerons un autre projet sur lequel une personne externe va pouvoir influencer et au besoin modifier les différentes composantes de celui-ci.

Afin de pouvoir déposer gratuitement sur les serveurs GitHub un projet, il est tout d'abord nécessaire de créer un compte, une fois cela fait il faut définir ce que l'on appelle une repository ou répertoire en français afin de délimiter le lieu de dépôt de notre projet. On va ici déposer un projet d'application de prospection client :



Afin de déclarer un projet dans GitHub, il nous est possible de soit l'ajouter directement depuis le menu comme montrer ci-dessus ou alors depuis les lignes de commandes à l'aide du Bash :

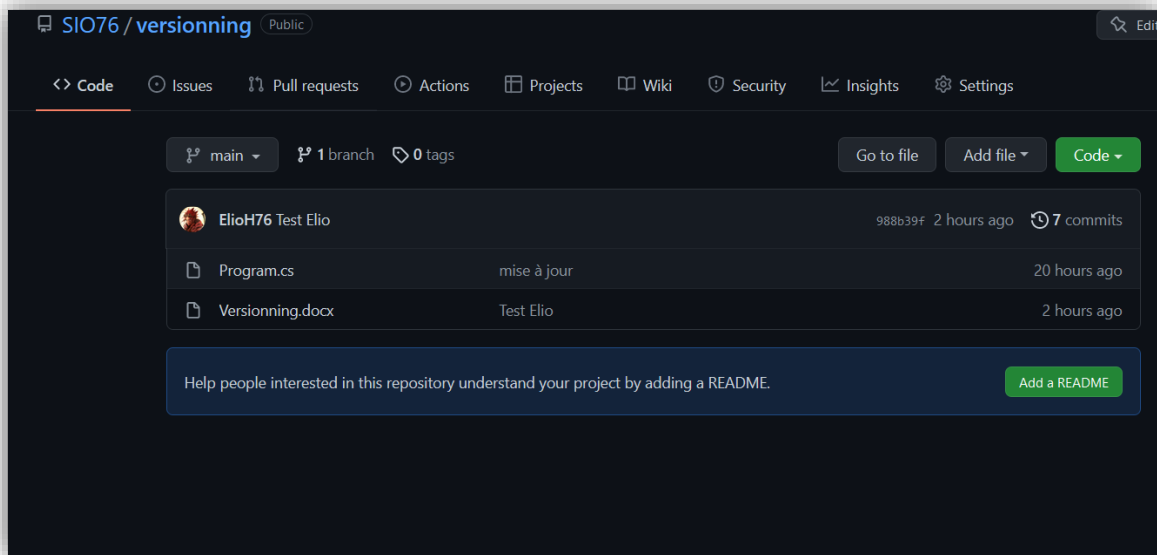


```
MINGW64:/c/Users/jcond/Documents/Cours BTS/PPE Fin de Formation/PPE 1/...
jcond@LAPTOP-GB8A7PAJ MINGW64 ~/Documents/Cours BTS/PPE Fin de Formation/PPE 1/A
pplication_Prospexion (testing)
$
```

## V-Partager un projet et travailler en commun

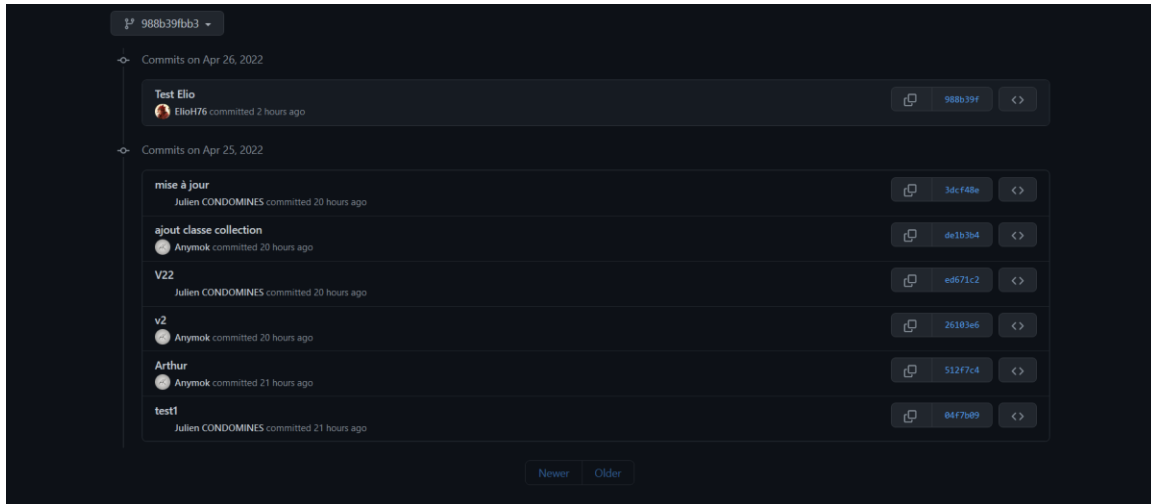
La partie la plus intéressante du versionning est la possibilité de travailler à plusieurs sur un même projet où que l'on soit placé géographiquement, au nombre que l'on souhaite en même temps. Il faut cependant faire attention à ce que l'on appelle les problèmes de versions, en effet, si deux personnes soumettent en théorie une modification sur un projet en même temps, alors celle-ci ne pourra se faire ; Pour pallier à cela, il faut utiliser la technologie du push et pull mis en place par le site, on va donc soumettre une demande de mise à niveau lors de nos commit, ce ceux vont être analysés et confrontés pour savoir si une mise à niveau du projet principale est possible de suite ou s'il faut d'abord mettre à niveau notre version du programme.

Nous allons ici déposer un projet en commun sur GitHub, que nous allons chacun pouvoir modifier après avoir mis à jour notre version puis le redéposer sur le serveur :





## LE VERSIONNING



On va donc pouvoir à partir d'ici télécharger la dernière version du programme, la modifier et ensuite la soumettre à nouveau au serveur pour mettre à jour la version finale du programme.

```
MINGW64:/c/Users/jcond/Documents/Cours BTS/Cours 2 Lundi Aprem Mercy/v...
jcond@LAPTOP-GB8A7PAJ MINGW64 ~/Documents/Cours BTS/Cours 2 Lundi Aprem Mercy/ve
rsionning (main)
$ git commit -m 'Mise à jour v1.1'
[main b811ba6] Mise à jour v1.1
1 file changed, 1 insertion(+), 1 deletion(-)

jcond@LAPTOP-GB8A7PAJ MINGW64 ~/Documents/Cours BTS/Cours 2 Lundi Aprem Mercy/ve
rsionning (main)
$ git push https://github.com/SIO76/versionning
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 349 bytes | 349.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/SIO76/versionning
  988b39f..b811ba6  main -> main

jcond@LAPTOP-GB8A7PAJ MINGW64 ~/Documents/Cours BTS/Cours 2 Lundi Aprem Mercy/ve
rsionning (main)
$
```

### VI-Conclusion

Nous avons donc vu ici comment le versionning va permettre d'aider les développeurs lors de projet en communs, ceux-ci vont pouvoir facilement communiquer leurs modifications sur les projets qu'elles soient réalisées en même temp ou non, qu'importe la zone géographique ou le nombre de personne travaillant en même temps sur le projet.