**BELABED Yacine**

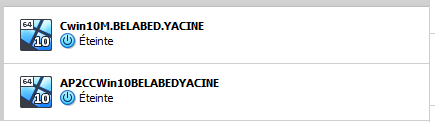
**1TSSIOA**

**15/01/2024**

**MISSION 1 - PROJET 1:**

**COMPTE-RENDU - INSTALLATION DE LA MACHINE VIRTUEL**

→ Les machines virtuelles ont déjà été installées auparavant (semestre 1) :

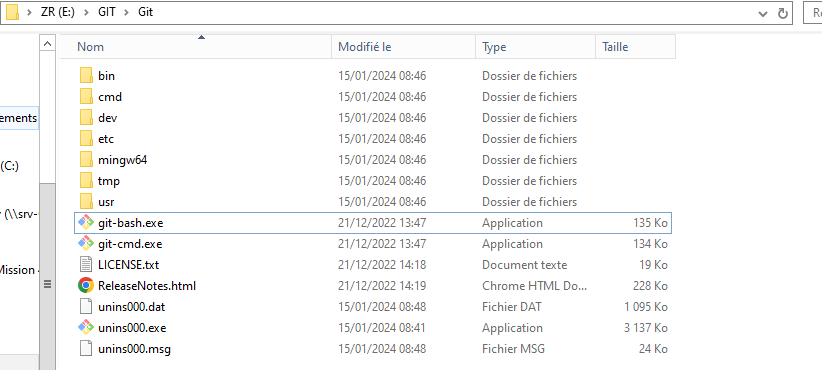


Je les ai donc renommés pour l'avancée du projet. Les paramètres sont conformes pour l’avancée de la mission.

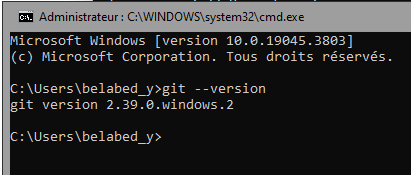
**MISSION 1 - PROJET 2:**

**COMPTE-RENDU - INSTALLATION DE GIT**

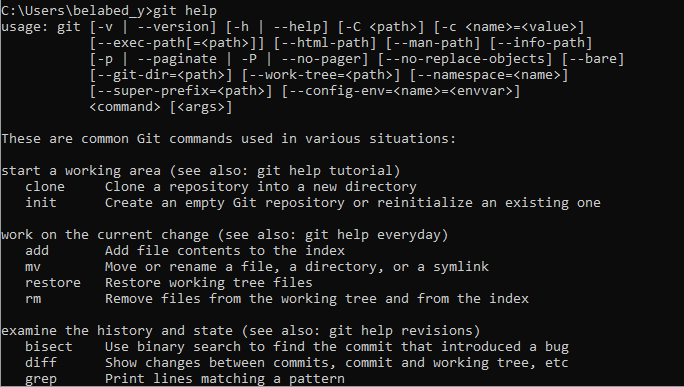
→ L’environnement de travail a été installé dans le disque dur :



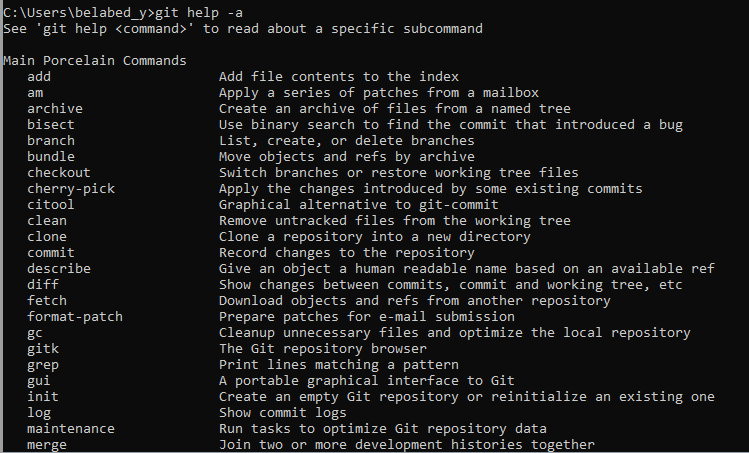
##### Vérification de la version de GIT : usage de la commande “git –version”



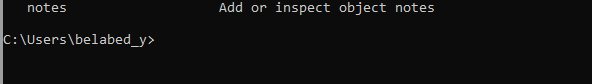
Test de la commande “git help” pour l’obtention de l’aide générale :



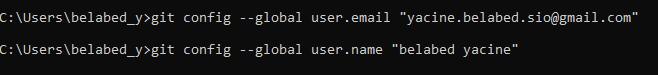
Test de la commande “git help -a” pour l’obtention de l’aide détaillée sur une commande connue :



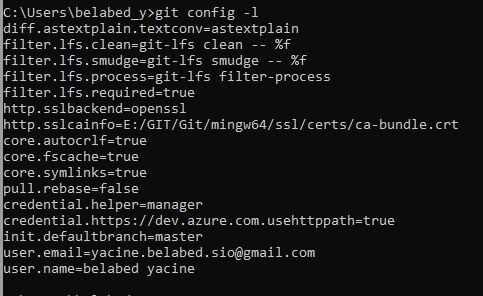
Saisi de la lettre ***[q]*** pour quitter ce mode :



Saisi des identifiants ***(mail + nom,prénom)*** afin de pouvoir utiliser les commandes GIT vers le serveur distant. Utilisation également de la commande ***“- -global”*** permettant donc de conserver les informations fournies pour tous les prochains projets :



Usage de la commande ***git config -l*** permettant de vérifier l'ensemble des paramètres de configuration :



→ On voit bien ici que le mail ainsi que le nom/prénom ont bien été enregistrés dans la configuration.

**MISSION 1 - PROJET 3:**

**COMPTE-RENDU - UTILISATION DU LOGICIEL GIT**

**EXPLICATION THÉORIQUE (déroulement) :**

1. La commande ***git init*** permet de générer une base locale dans le répertoire de travail. La base est dans le répertoire caché .git.

2. La commande ***git add*** permet d’ajouter les fichiers et dossiers à synchroniser.

3. Lorsque les fichiers à synchroniser sont modifiés, il faut enregistrer une version, en utilisant la commande ***git commit***.

4. La commande ***git commit avec l’option -a*** ajoute l’ensemble des fichiers et dossiers du répertoire de travail à la synchronisation et gère les modifications de la base de données.

5. Pour pouvoir envoyer le projet sur un serveur, il faut utiliser la commande de configuration ***git remote add*** qui associe un label à l'URL du projet hébergé.

6. L'envoi du projet se fait avec la commande ***git push***.

###### **...A.1.1 Schéma de flux depuis un serveur distant**

Lorsqu'un utilisateur veut récupérer un projet depuis Internet, le schéma est différent :

Les étapes sont les suivantes :

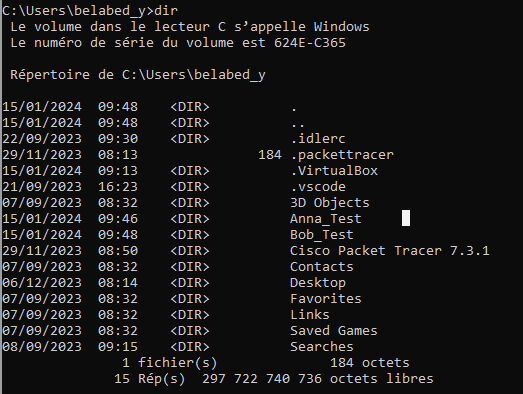
1. Il faut ***se placer dans le répertoire local de travail***. La base sera téléchargée en même temps que les fichiers, avec la commande ***git clone***.

2. Avant de commencer à travailler, un codeur doit toujours récupérer les dernières mises à jour du projet, par la commande ***git fetch***.

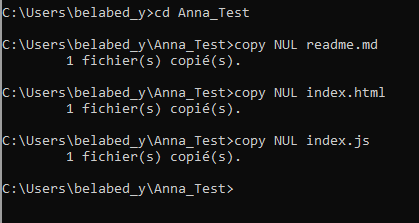
3. La commande ***git merge*** tente de fusionner les différentes versions, pour ne pas écraser de modifications (du développeur local ou du projet distant).

**MISE EN PRATIQUE :**

*PARTIE 1 - ANNA :*

→ Utilisation de la commande ***git init*** afin de créer les répertoires Anna\_Test ainsi suivi du répertoire Bob\_Test pour gagner du temps :

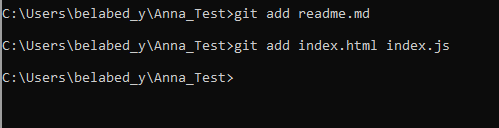
→ Création des fichiers dans le répertoire ***“Anna\_Test”*** :



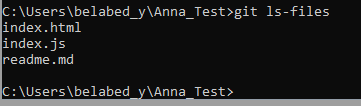
***Préparation de la base locale GIT :***



***Ajout des fichiers à surveiller :***

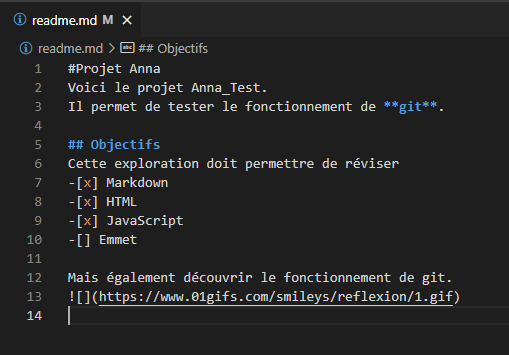


***Vérification de l’ajout des fichiers :***

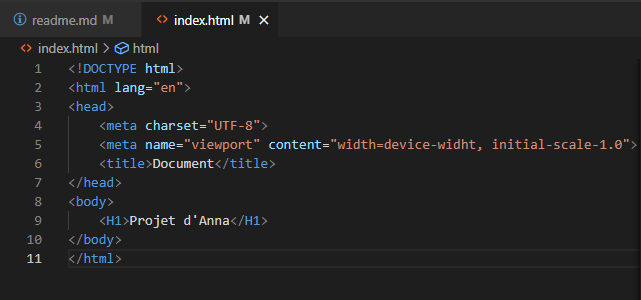


***MODIFIER LES FICHIERS :***

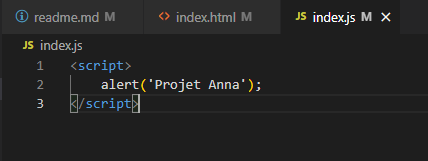
readme.md :



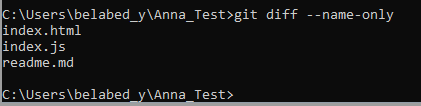
index.html :



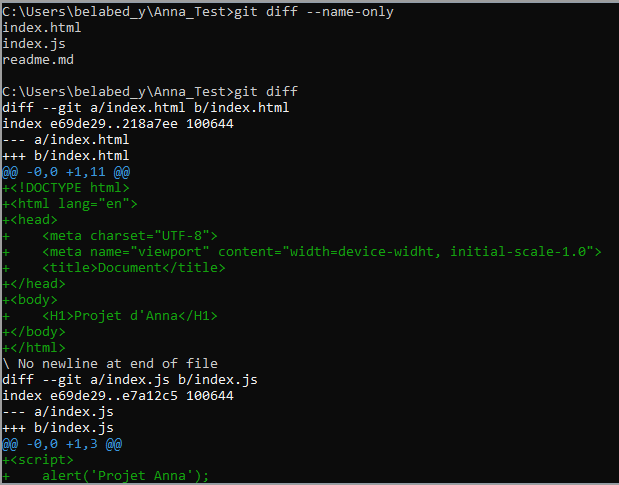
index.js :



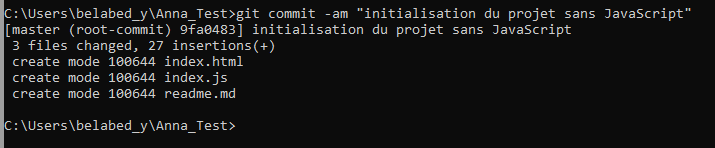
→ Utilisation de la commande ***“git diff --name-only”*** pour voir les fichiers modifiés :



→ Vérification du contenu des changements dans les fichiers via la commande ***“git diff”*** .

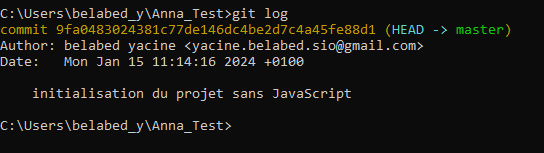


→ Enregistrement du projet dans la base locale de Anna\_Test :



**VÉRIFICATION DES ENREGISTREMENTS :**

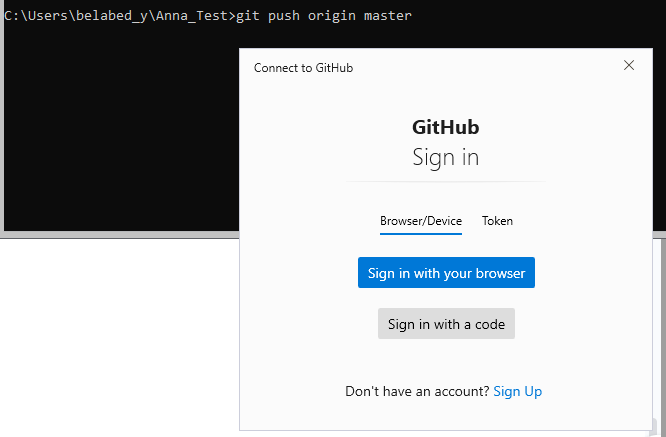
→ Vérification de la conformité de l’enregistrement grâce à la commande ***“git log”*** :



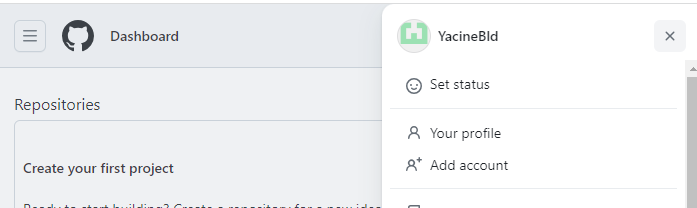
→ Création du projet sur un serveur (via un lien) :

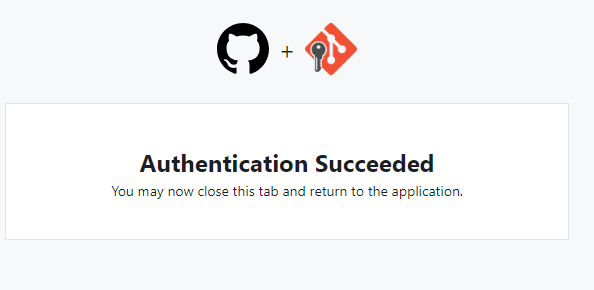


→ Utilisation de la commande ***“git push”*** pour l’envoi du projet. Une fenêtre GitHub s’ouvre alors :

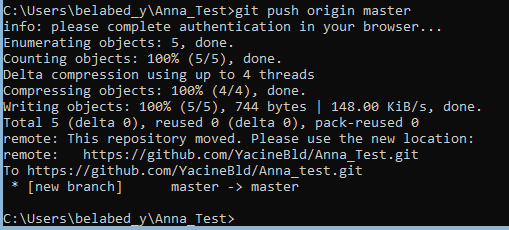


→ Création d’un compte GitHub :



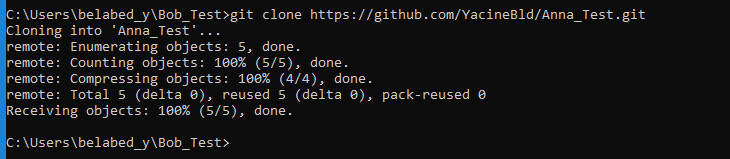
→ Authentification réussie avec le lien : 

Résultat de la commande :

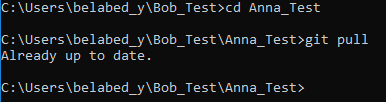


*PARTIE 2 - BOB :*

***Créer le projet (clonage) :***



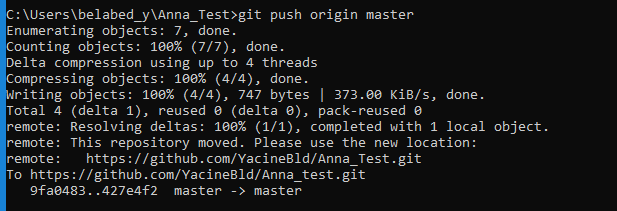
***Mettre à jour le projet local :***

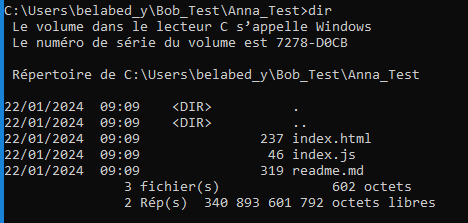


***Modifier le projet :***



Résultat :

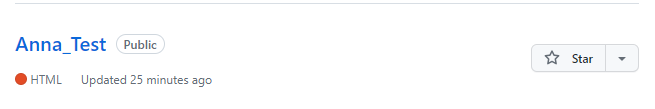


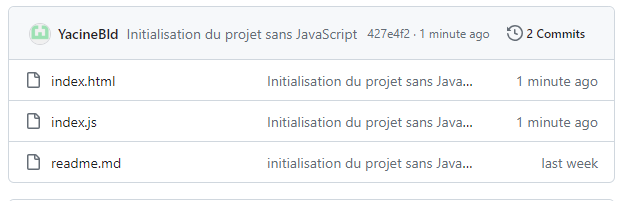


***Évolutions du projet :***

Les branches dans les systèmes de contrôle de version comme Git sont en effet une excellente solution pour permettre à différents membres de l'équipe de travailler sur des fonctionnalités ou des corrections de bogues séparées sans interférer les uns avec les autres.

##### **Création du projet GITHUB → Vérification de leur intégrité sur GITHUB :**





**MISSION 1 - PROJET 4:**

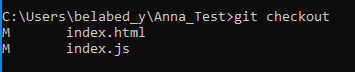
**COMPTE-RENDU - UTILISATION DES BRANCHES AVEC GIT**

##### **Gestion des fonctionnalités avec GIT :**

→ Utilisation de la commande git branch :



→ Utilisation de la commande git checkout :

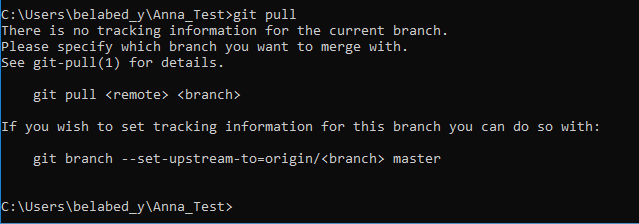


→ Utilisation de la commande git merge :

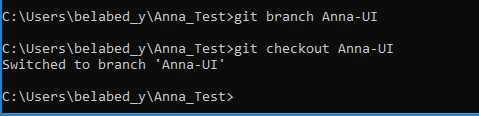
##### **MISE EN PRATIQUE :**

*1ER PARTIE - ANNA :*

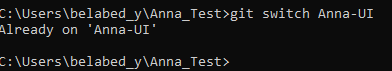
→ Utilisation de la commande git pull pour synchroniser le projet :



→ Création et placement sur la branche à l’aide des commandes git branch et git checkout :

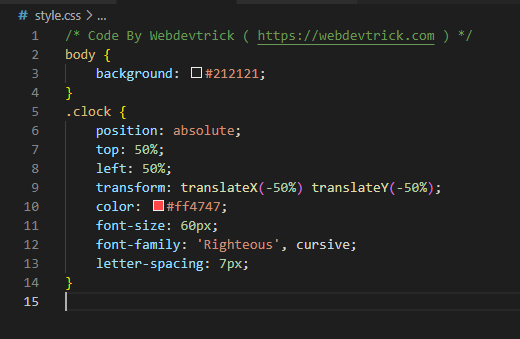


→ Utilisation de la commande switch qui remplace la commande checkout pour basculer de branche :

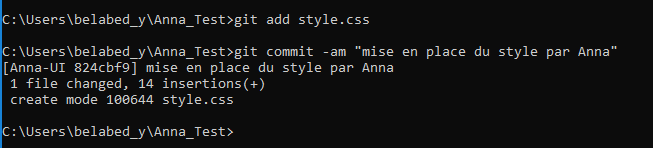


***Ajout d’un fichier de style :***

Ajout et paramétrage du du nouveau fichier style.css :

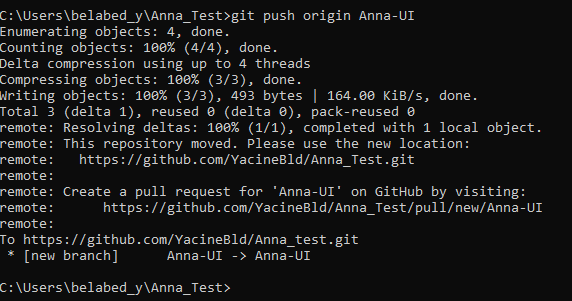


Ajout du nouveau fichier à la base locale GIT (et versionné) :

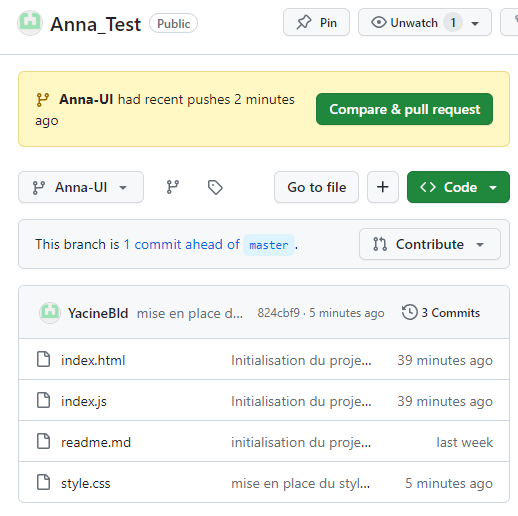
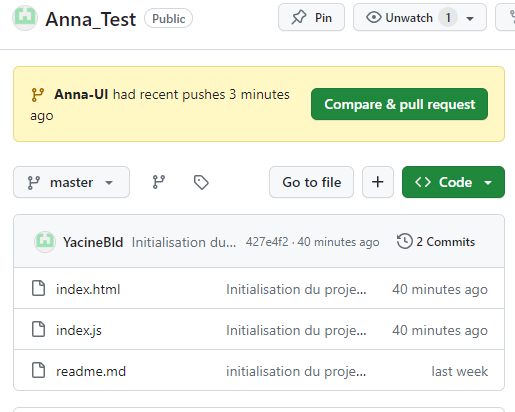


***Envoi vers le serveur :***

Synchronisation avec le serveur :



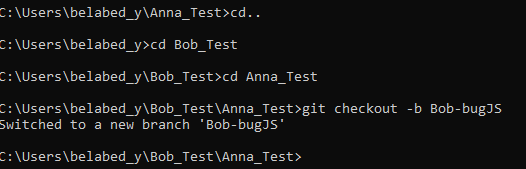
Résultats depuis le navigateur :



*1ER PARTIE - BOB :*

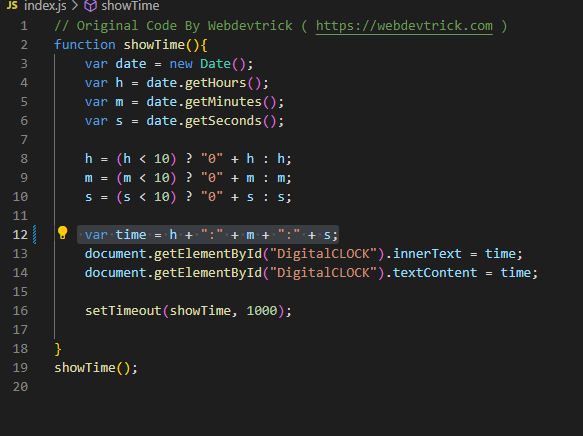
***Créer la branche Bob-bugJS :***

→ Utilisation de l’option -b de la commande checkout (raccourci) :



***Corriger le script :***

→ Correction apporté au script JS (ligne 12) :

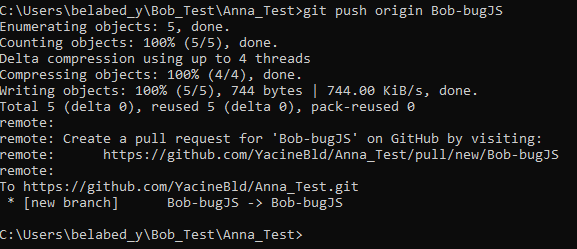


Résultat :

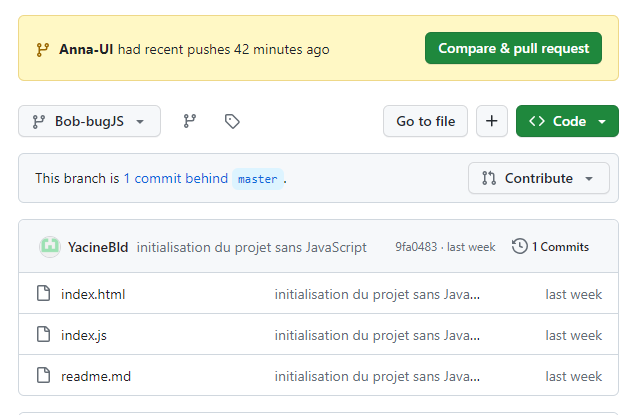




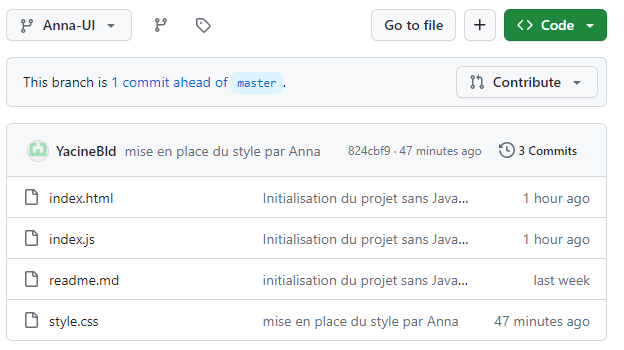
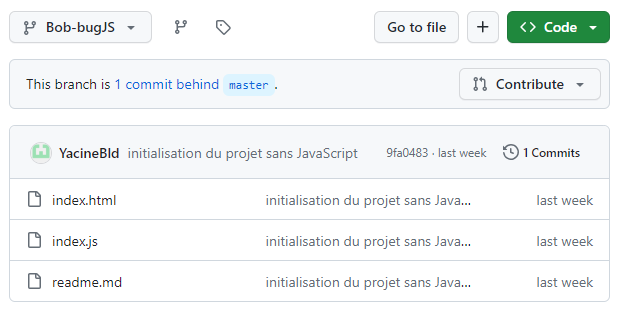
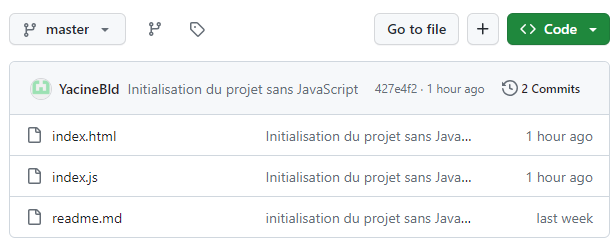
***Pousser la modification sur le serveur :***



Résultat depuis le navigateur :



***Comparaison des branches :***

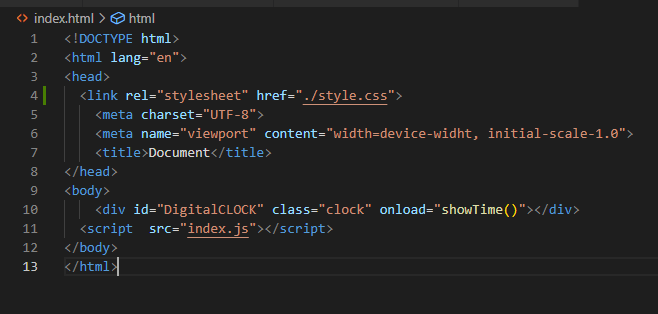


→ Lorsque deux personnes travaillent sur le même fichier, des conflits peuvent survenir lorsqu'elles tentent de fusionner leurs modifications. Git est conçu pour gérer ces conflits de manière efficace et fournir un moyen de les résoudre.

***SECONDE PARTIE ANNA :***

***Edition et publication du fichiers index.html :***

→ Ajout de la ligne (ligne 4) dans le fichier index.html :



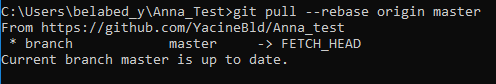
***Fusion avec la branche master :***

→ Utilisation de la commande git checkout master :

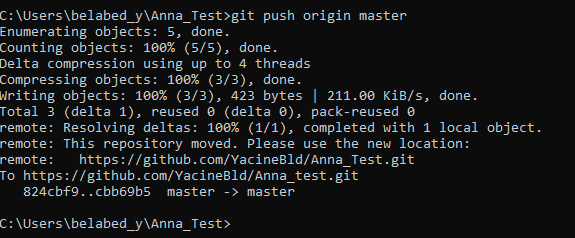
→ Utilisation de la commande git merge pour effectuer la fusion :



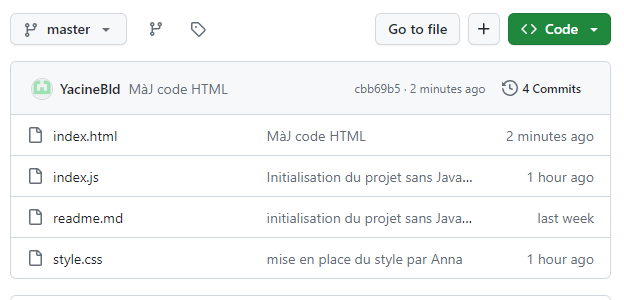
→ Utilisation de la commande git pull pour s’assurer que les modifications soient locales :



→ Synchronisation du serveur :

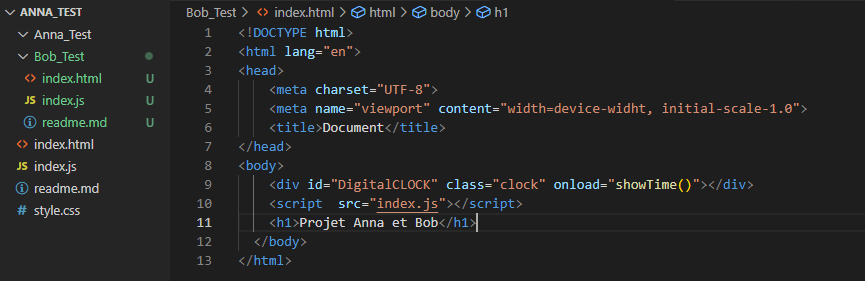


Résultat depuis le navigateur :

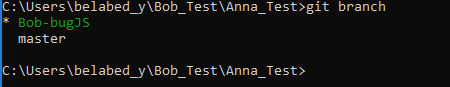


***SECONDE PARTIE BOB :***

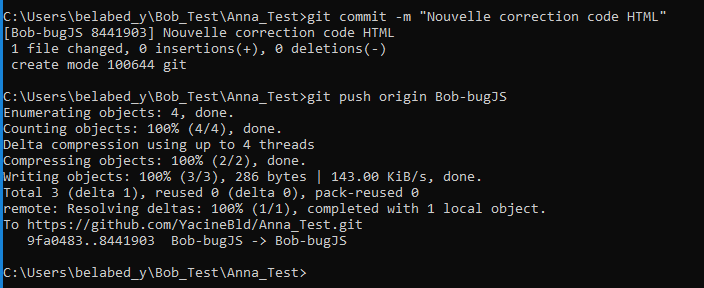
→ Ajout du titre manquant dans la section <body> du fichier de Bob :



→ Utilisation de la commande git branch afin qu’on vérifie bien qu’on travail dans la branche de Bob :



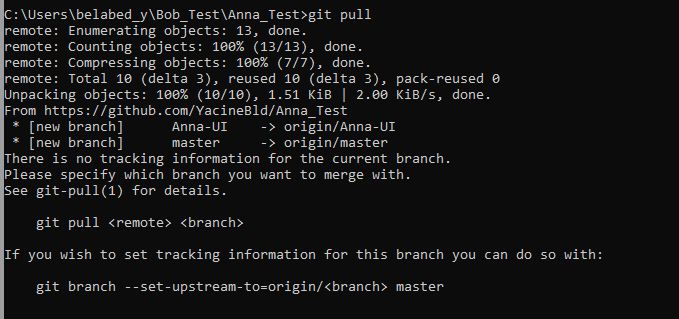
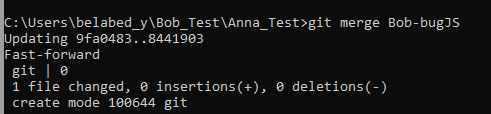
→ Utilisation des commandes commit et push pour valider la branche sur le serveur :

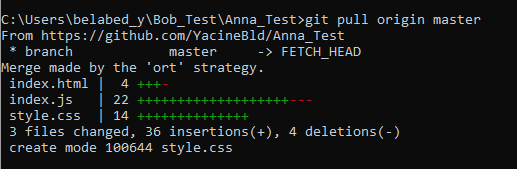
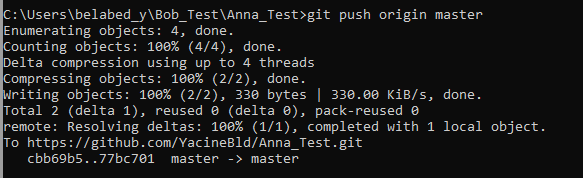


***Fusion avec la branche master :***

→ Utilisation de la commande git checkout master pour vérifier la synchronisation :







→ Résultats depuis Github :

