** الجمهورية الجزائرية الديمقراطية الشعبية**

**وزارة التكوين والتعليم المهني**

**المعهد الوطني لتكوين المهني عبد الحق حمودة – المدية –**

**Expliquer de projet collectif sur git et Git Hub**

**Nom : Prénom :**

* Soualhi Riham
* Hamidi Ikram
* Debieb Amina
* Bettache Nahrawan
* Slama Maria

**Section**: S3 DWM

**Prof**: Bachen

**Introduction**

​ Dans l'écosystème actuel du développement logiciel, la capacité à travailler efficacement en équipe est tout aussi cruciale que la compétence technique individuelle. La gestion de projet ne se limite plus à l'écriture de code, mais s'étend à la coordination, à l'intégration continue et à la gestion des versions. C'est dans ce contexte que s'inscrit notre projet collectif intitulé "Site Collectif".

**Objectifs**

* ​Maîtrise de l'Architecture Client-Serveur (Git/Git Hub) : Comprendre l'interaction entre les dépôts locaux (sur les machines de chaque membre) et le dépôt distant (Remote).
* ​Application du "Feature Branch Workflow" : S'assurer que chaque fonctionnalité est développée de manière isolée sur une branche spécifique pour protéger le code principal (main).
* ​Répartition et Réalisation des Tâches Techniques :
* ​Développement de la structure principale (index.html) pour la page d'accueil.
* ​Création d'un formulaire fonctionnel (contact.html) pour la page de contact.
* ​Mise en place d'une charte graphique harmonieuse via les feuilles de style (style.css).
* ​Création de la page de Services (service.html).
* ​Conception de la page À Propos (about.html).
* Collaboration et Revue de Code : Utiliser les Pull Requests pour valider le travail de chaque membre avant son intégration.
* ​Gestion de Projet et Livraison : Coordonner les fusions (Merges) et marquer la version finale du projet (Tagging v1.0).

1. **Réalisation Technique et Méthodologie**

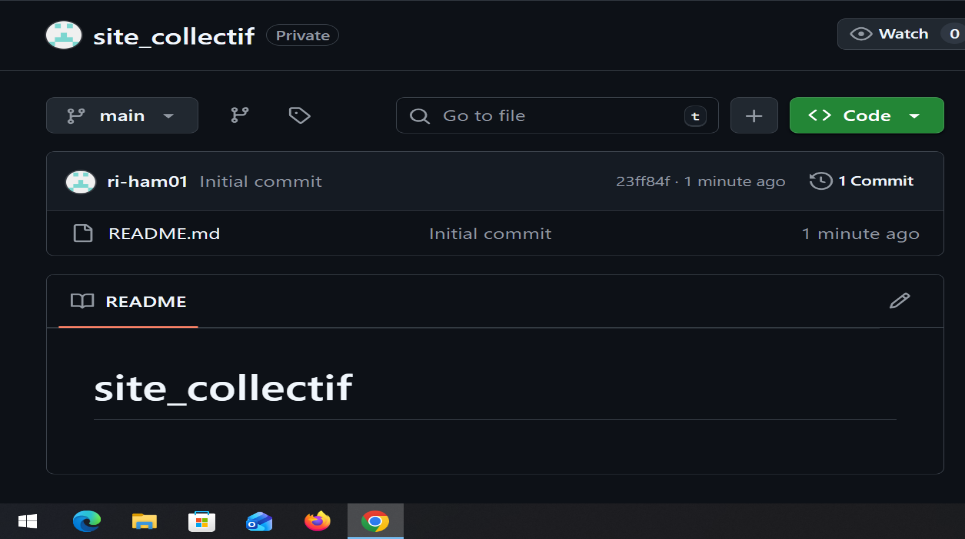
* **L’administrateur :**

1. ​**Phase 1** **: Initialisation et Préparation :**

Commencer par créer l'espace de travail centralisé.

​**Action** : Création du dépôt public site\_collectif sur Git Hub et ajout d'un fichier README.md.

**But** : Mettre en place le dépôt distant (Remote) pour permettre à l'équipe de cloner le projet.

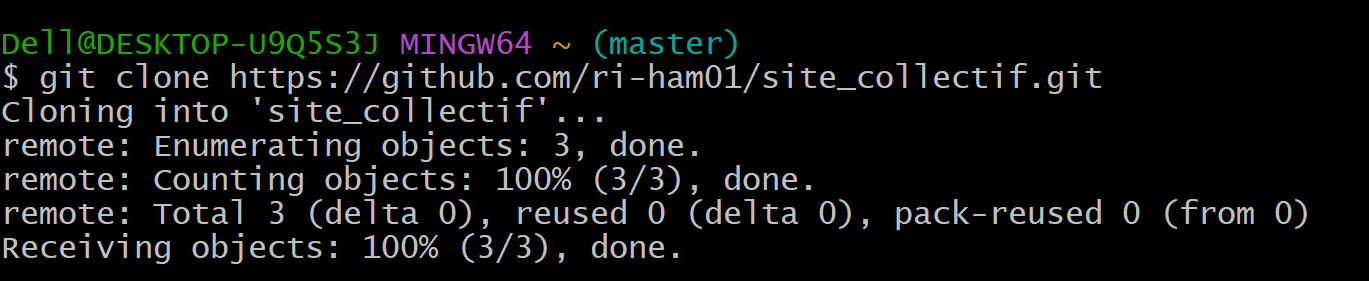


Le Responsable a commencé le travail en obtenant une copie locale du dépôt distant sur laquelle travailler.

* **Commande Utilisée** :

Git clone https://github.com/ri-ham01/site\_collectif.git

* **Explication** : La commande git clone est une commande fondamentale utilisée pour créer une copie miroir (une copie locale) d'un dépôt Git distant (Remote Repository) existant sur une plateforme comme Git Hub.

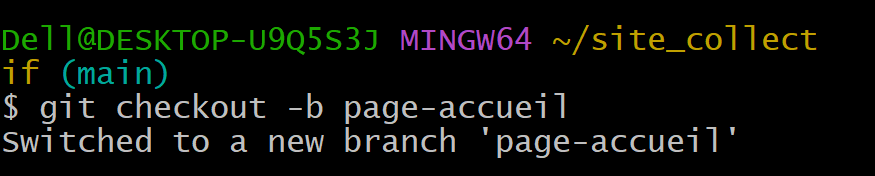


* **Rôle** : Cette commande effectue les actions suivantes :
* Télécharge tous les fichiers du projet et l'historique complet des modifications (Commit History).
* Crée un dossier local nommé site\_collectif.
* Initialise un environnement Git à l'intérieur de ce dossier, en configurant automatiquement la référence distante (Remote) sous le nom origin, qui pointe vers l'URL du dépôt cloné.
* **Résultat** : Le Responsable dispose désormais d'un environnement de travail local parfaitement synchronisé avec le dépôt central.

1. **Phase Deux : Isolation et Développement**

Pour garantir de ne pas perturber le travail sur la branche principale, le Responsable a créé une nouvelle branche pour développer sa fonctionnalité.

* **Commande Utilisée :**  git checkout -b page accueil
* **Explication** : La commande git checkout -b est une commande composée qui exécute deux actions simultanément.
* -b page-accueil : Crée une nouvelle branche (Branch) nommée page-accueil.
* git checkout : Bascule immédiatement vers cette nouvelle branche.

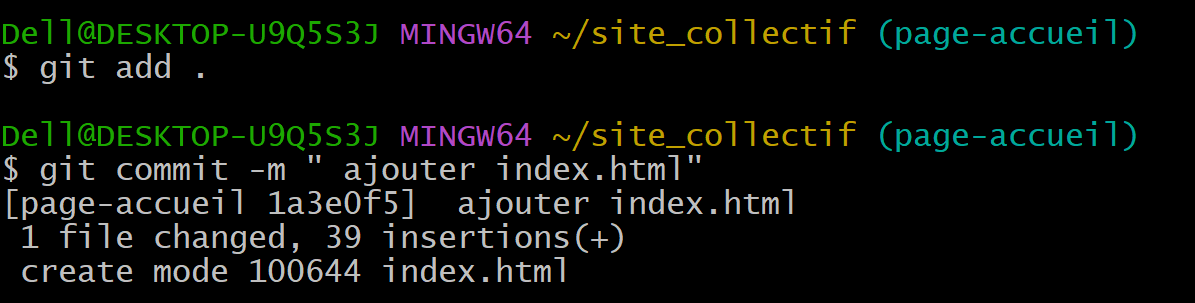


* **Rôle :**
* Création de Branche : Le système de branche de Git permet à plusieurs développeurs de travailler sur des fonctionnalités ou des corrections différentes en parallèle et de manière isolée.
* Isolation : Toute modification apportée par le Responsable sera désormais confinée à la branche page-accueil et n'affectera pas le code stable de la branche main.
* **Travail sur les Fichiers :**
* **Commande** : touch index.html
* **Explication et Rôle** : Commande système permettant de créer un nouveau fichier vide nommé index.html. Dans le contexte du développement web, le fichier index.html est généralement la page d'entrée principale de tout site web.

1. **Phase Trois : Versionnement Local des Modifications**

Après avoir terminé le développement, le Responsable a enregistré ces modifications dans l'historique local de Git.

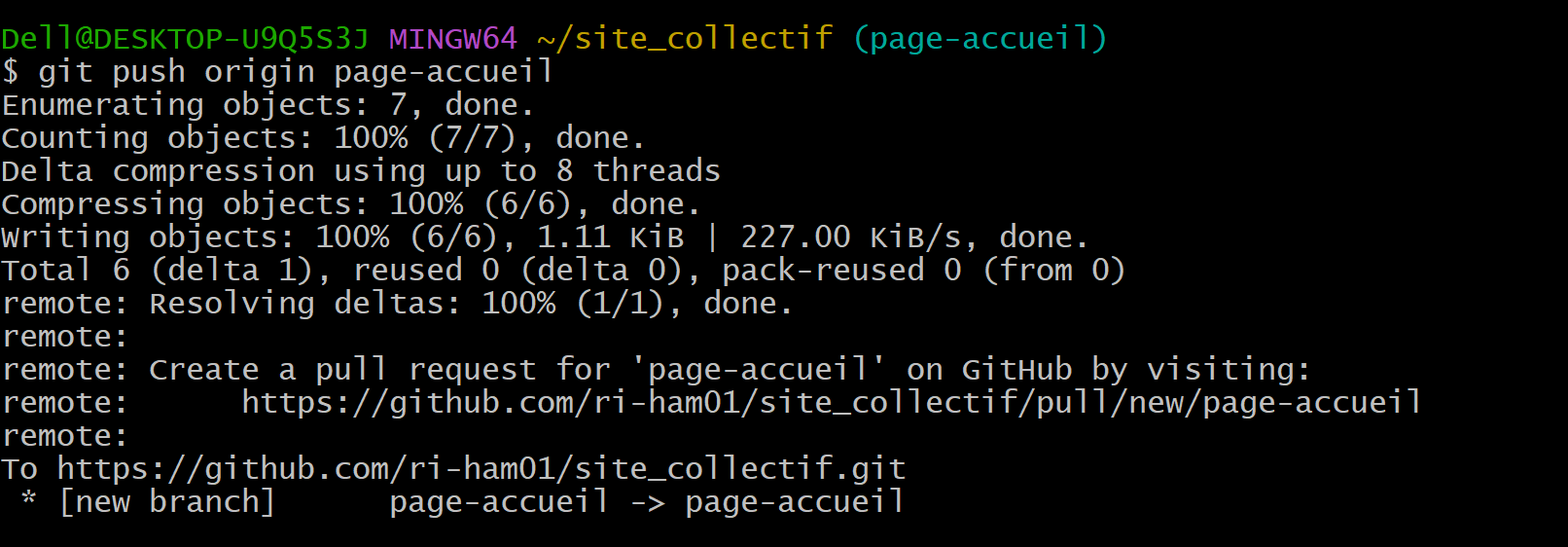
* **Commande Utilisée** : git add .
* **Explication** : La commande git add déplace les fichiers d'un état "non suivi" (Untracked) ou "modifié" (Modified) vers la zone de "préparation" (Staging Area).
* **Rôle** : Permet de spécifier les modifications que l'on souhaite inclure dans le prochain "instantané" de sauvegarde (Commit). Le point (.) signifie "tous les fichiers nouveaux et modifiés dans le dossier courant et ses sous-dossiers".
* **Commande Utilisée** : git commit -m "ajouter index.html"
* **Explication** : La commande git commit prend tous les fichiers de la zone de préparation (Staging Area) et les stocke de manière permanente dans la base de données de Git avec une description de ces changements.
* **Rôle** :
* Créer un "instantané" (Snapshot) de l'état du projet à cet instant.
* L'option -m permet d'écrire le message de commit directement. Le message "ajouter index.html" décrit précisément le contenu de ce commit.
* **Résultat** : Le travail sur la nouvelle fonctionnalité est sauvegardé en toute sécurité dans l'historique local sur la branche page-accueil.



1. **Phase Quatre : Envoi des Modifications et Invitation à la Collaboration**

Pour transférer le travail de la machine locale vers le dépôt central et informer l'équipe des changements, le Responsable a poussé la branche.

* **Commande Utilisée :** git push origin page-accueil
* **Explication** : La commande git push transfère les commits de la branche locale vers le dépôt distant (Remote Repository).
* **Rôle :**
* Origin : Le nom de la référence distante (le dépôt sur Git Hub).
* Page-accueil : Le nom de la branche à pousser.
* Cette action crée une nouvelle branche sur Git Hub nommée page-accueil contenant le code de la page d'accueil.



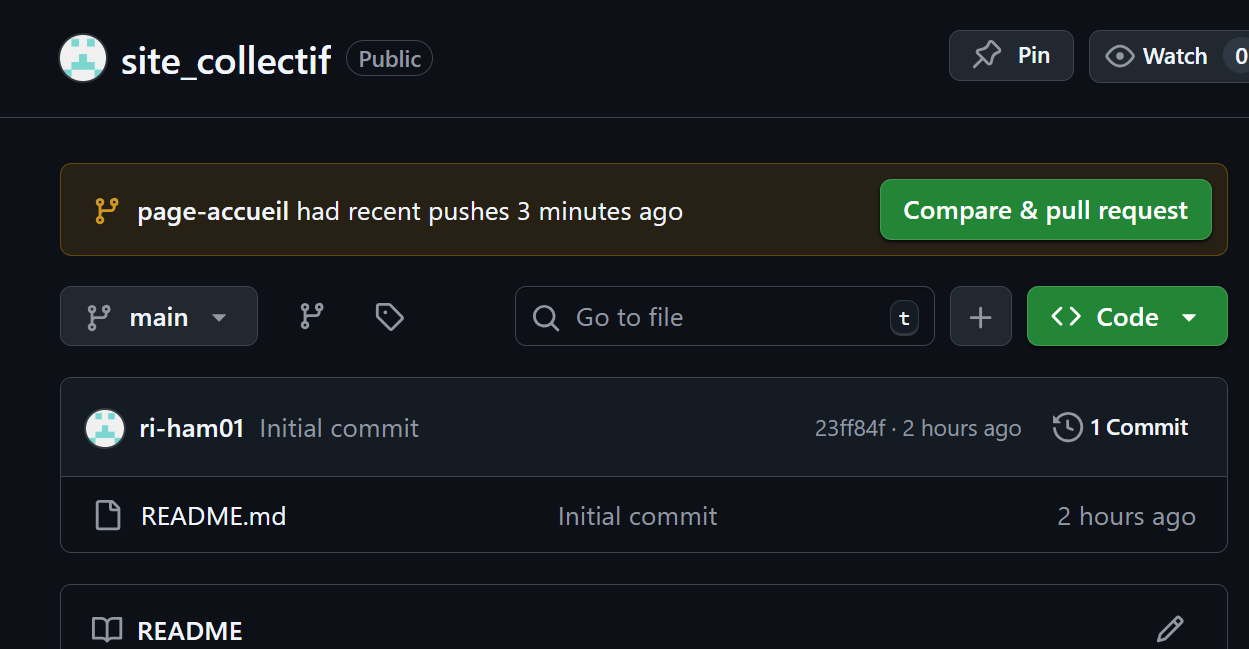
Après avoir poussé la branche, le Responsable peut se rendre sur l'interface Git Hub et créer une "Pull Request" (Demande de tirage). Une Pull Request est un mécanisme formel pour proposer des modifications à un projet, permettant aux autres membres de réviser le code (Code Review), d'en discuter et de faire des commentaires avant son intégration dans la branche principale main. Ceci constitue "l'invitation" effective des membres à examiner le travail.

1. **Phase Cinq : Intégration des Contributions de l'Équipe**

Après avoir examiné les Pull Requests des membres et donné son approbation, le Responsable les intègre dans la branche principale.

Opérations effectuées (généralement via l'interface GitHub) :

* **Accepter la Pull Request** : Cliquer sur le bouton "Merge pull request" dans l'interface GitHub pour chaque demande de tirage ayant été revue et approuvée.
* **Fusion (Merge) :** GitHub exécute automatiquement une opération git merge sur les serveurs distants, fusionnant le code de la branche (par exemple page-about ou page-contact) dans la branche main.
* **Mise à jour du Dépôt Local** : Après avoir finalisé toutes les fusions sur GitHub, le Responsable doit exécuter à nouveau git pull origin main sur sa machine locale pour récupérer tous ces changements fusionnés récemment.



1. **Phase Sexe : Préparation à l'Intégration des Contributions**

Pendant que les autres membres créent leurs propres branches et soumettent leurs Pull Requests, le Responsable (ou toute personne responsable de l'intégration) doit s'assurer que sa version locale de la branche principale est à jour.

* **Commande Utilisée** : git checkout main

Explication et Rôle : Revenir à la branche principale main pour se préparer à intégrer les changements.

* **Commande Utilisée** : git pull origin main
* **Explication :** La commande git pull est une commande composée qui exécute deux actions :
* Git fetch : Récupère les derniers changements du dépôt distant.
* Git merge : Fusionne ces changements dans la branche locale actuelle (main).

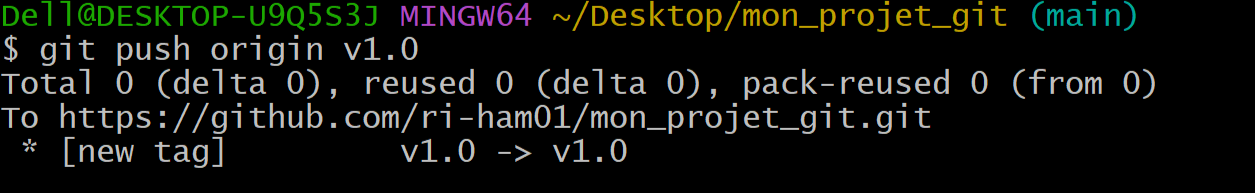


* **Rôle** : Cette commande garantit que la branche main locale est synchronisée avec l'état le plus récent du dépôt central. Ceci est crucial pour éviter les conflits de fusion (Merge Conflicts) lors de l'intégration des demandes des membres.

1. **Phase Sept : Marquage d'une Version (Tagging à Release)**

Après avoir fusionné toutes les fonctionnalités principales et considéré le projet comme stable, le Responsable a créé un tag pour la première version.

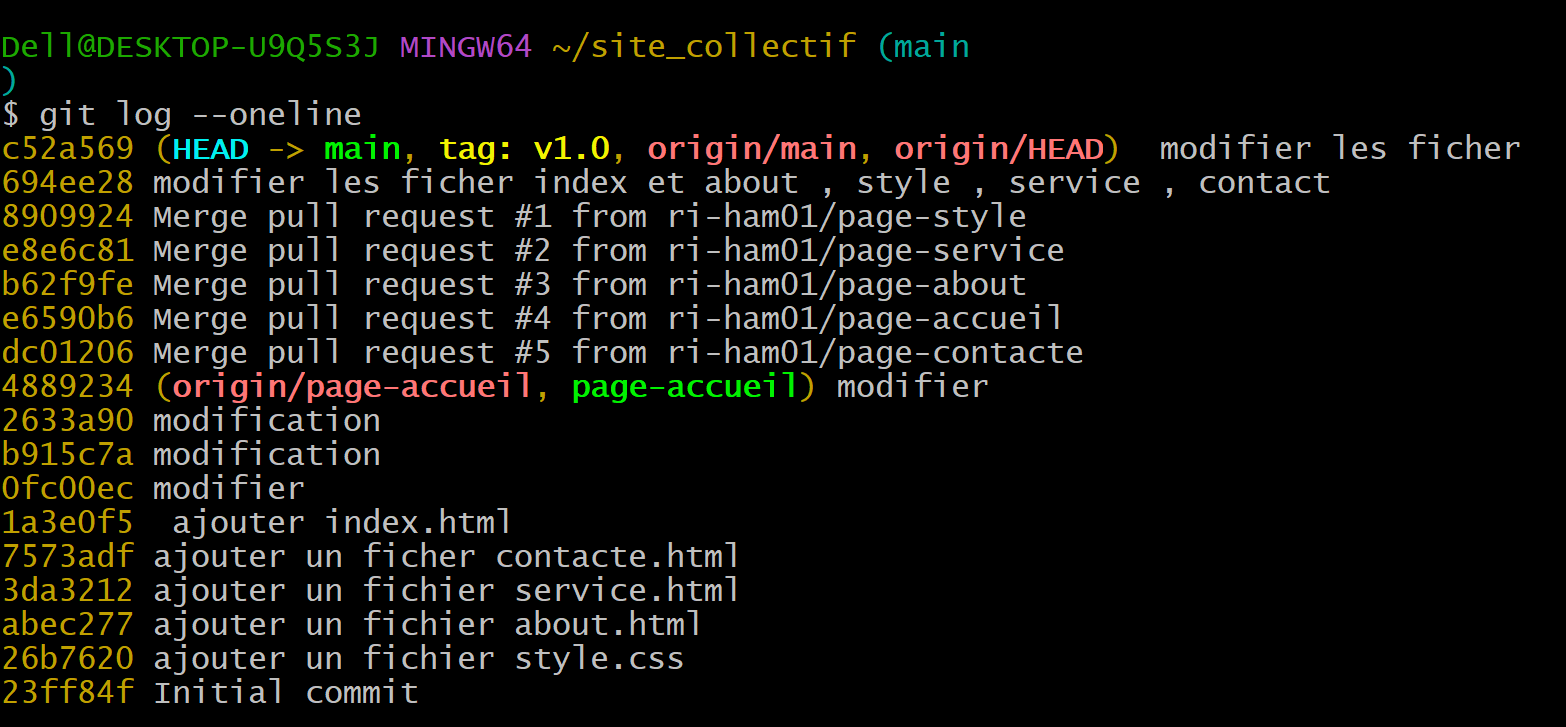
* **Commande Utilisée :** git tag v1.0
* **Explication** : Les tags dans Git sont utilisés pour marquer un point spécifique dans l'histoire du projet comme étant important, généralement pour des versions (ex : v1.0, v2.1.3).
* **Rôle** : Créer un tag local nommé v1.0 sur le dernier commit de la branche main actuelle. Ce tag indique que l'état du projet à cet instant est la "Version 1.0", prête pour la distribution. Les tags ne se déplacent pas automatiquement comme les branches ; ils sont statiques.



* **Commande Utilisée :** git push origin v1.0
* **Explication et Rôle** : Cette commande pousse le tag v1.0 du dépôt local vers le dépôt distant sur GitHub. Cela rend le tag visible par tous sur la plateforme GitHub et permet de se référer au code de cette version à tout moment dans le futur.

1. **Phase huitième Revue de l'Historique du Projet :**

* **Commande** : git log --oneline
* **Explication et Rôle** : Cette commande affiche l'historique des commits (Commit History) de manière concise, montrant pour chaque commit son hash abrégé et son message. C'est utile pour visualiser la séquence de travail effectuée par tous les membres et s'assurer que tous les changements sont présents dans l'historique.



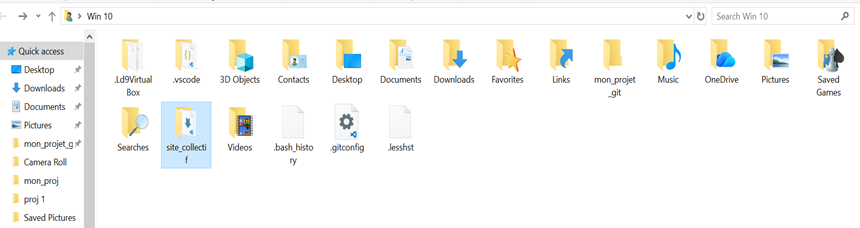
**contacte.html**

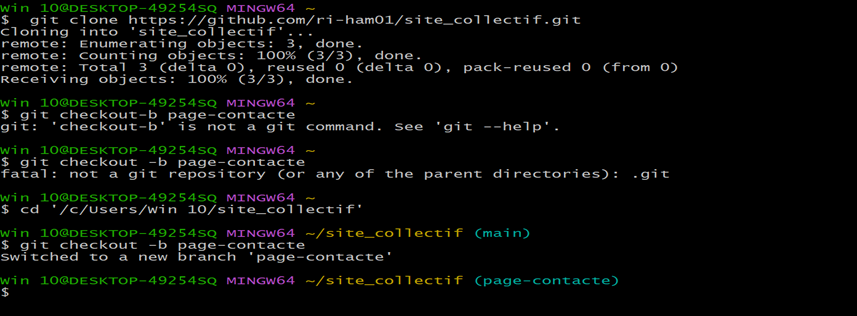
1) J’ai accepté l'invitation du responsable afin de pouvoir contribuer au projet.

2) J’ai utilisé la commande **git clone** pour télécharger l'intégralité du projet depuis Git Hub vers mon ordinateur personnel.

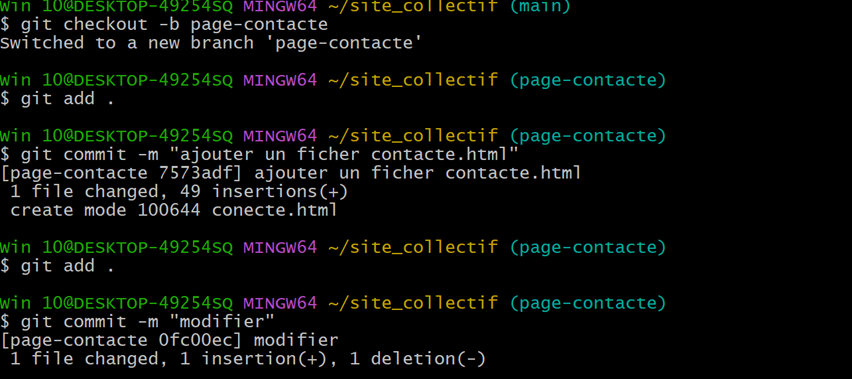
3) Je me suis rendu dans le répertoire du projet en utilisant la commande **cd** **site\_collectif.**

4) J’ai créé une nouvelle branche nommée **page-contacte** à l'aide de la commande **git** **checkout -b page-contacte**, puis j'ai commencé à y travailler.

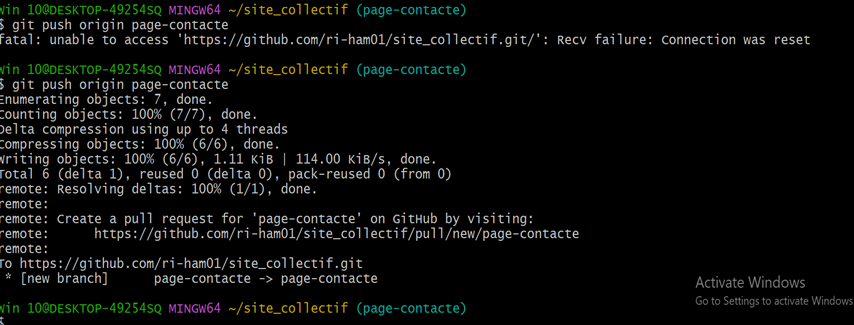




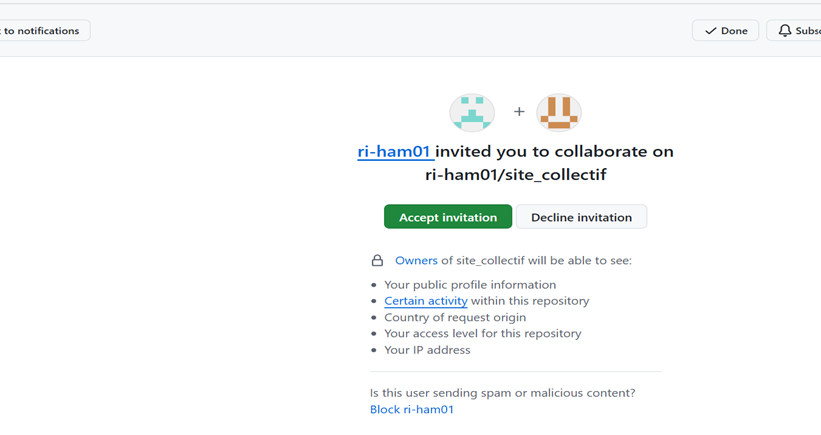
5) J'ai utilisé la commande **git add** pour ajouter les nouveaux fichiers aux changements indexés.

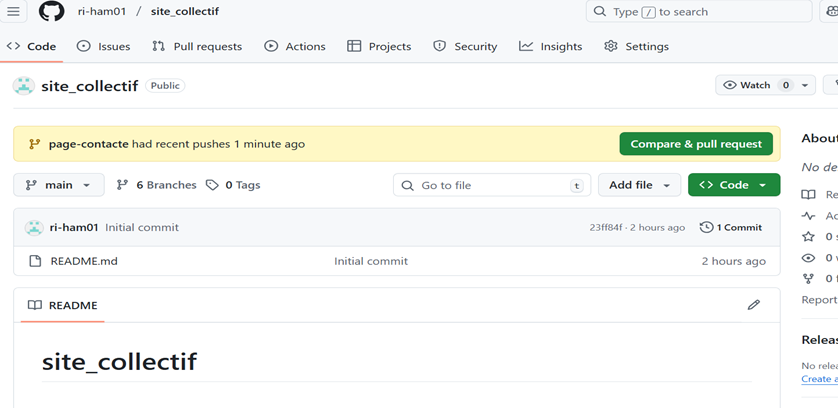
6) J’ai utilisé la commande **git commit** pour enregistrer et documenter de manière permanente les fichiers ajoutés. 

7) J’ai utilisé la commande **git push** pour envoyer ces enregistrements depuis mon ordinateur vers Git Hub, afin que le responsable puisse les examiner.



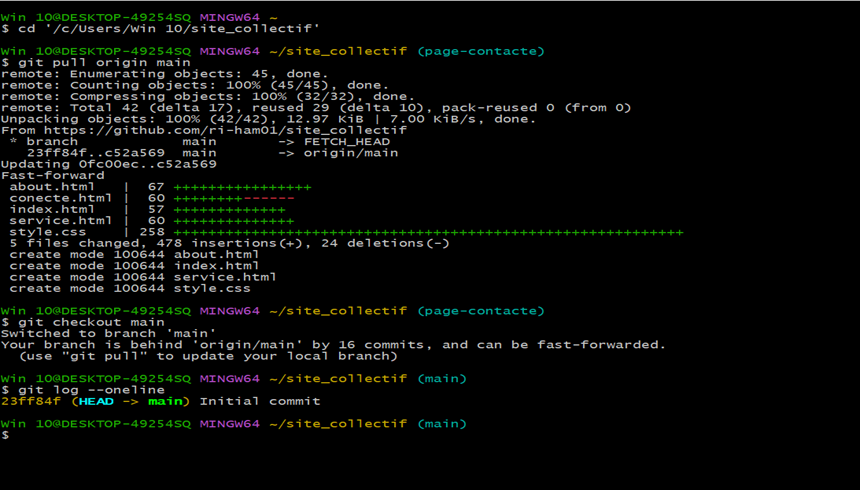
8) La plateforme Git Hub a confirmé que le travail avait bien été reçu.





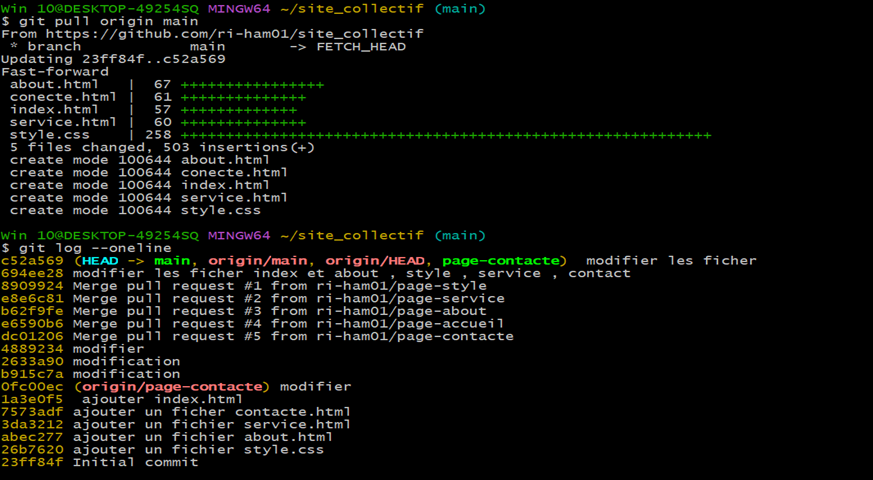
9) J'ai utilisé la commande **git pull origin main** pour récupérer tous les fichiers et les mises à jour de la branche principale sur Git Hub.

10) Je suis retourné à la branche originale main en utilisant **git checkout main**, puis j'ai utilisé **git log** pour vérifier le dernier enregistrement officiel disponible sur ma machine locale.



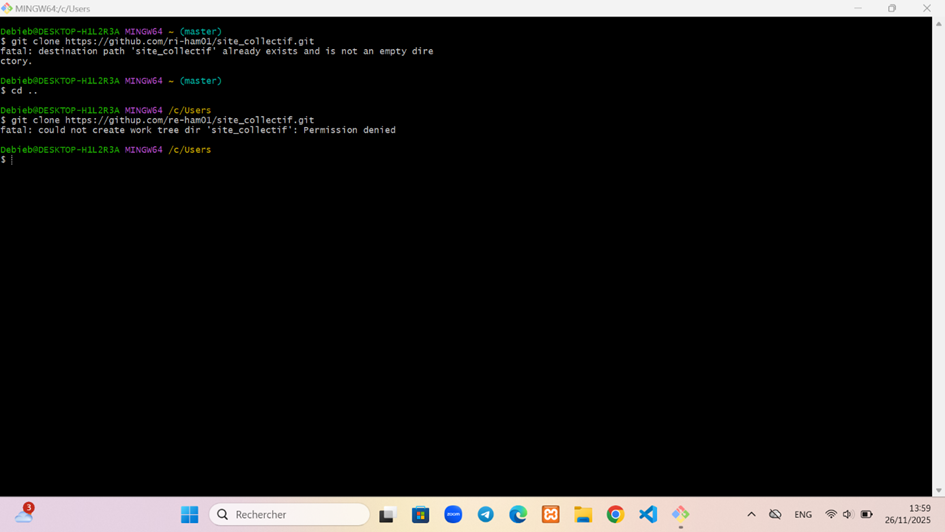
11) Étant sur la branche principale main, j'ai réutilisé la commande **git pull origin main** pour télécharger les mises à jour effectuées par mes collègues et ainsi mettre ma version à jour.

12) J'ai utilisé **git log** et j'ai constaté que ma branche locale main est parfaitement synchronisée avec la branche distante origin/main, confirmant que je possède la version la plus récente du projet.

******

**Page-style**

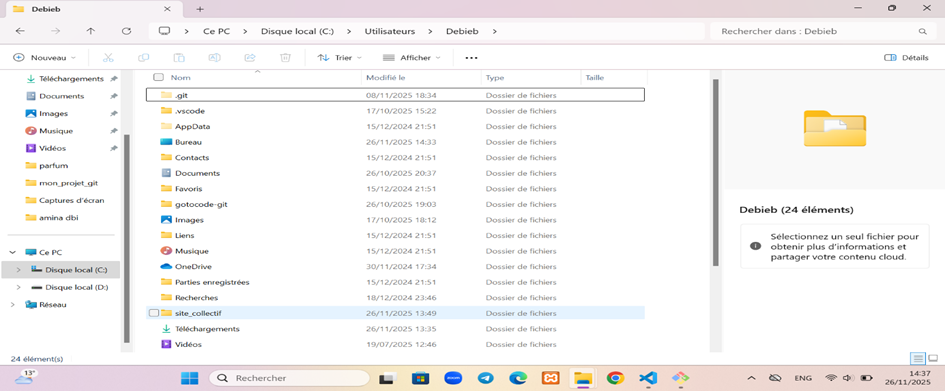
1. **Local Repository:**

******

Cette capture d’ecran montre le clonage du depot distant

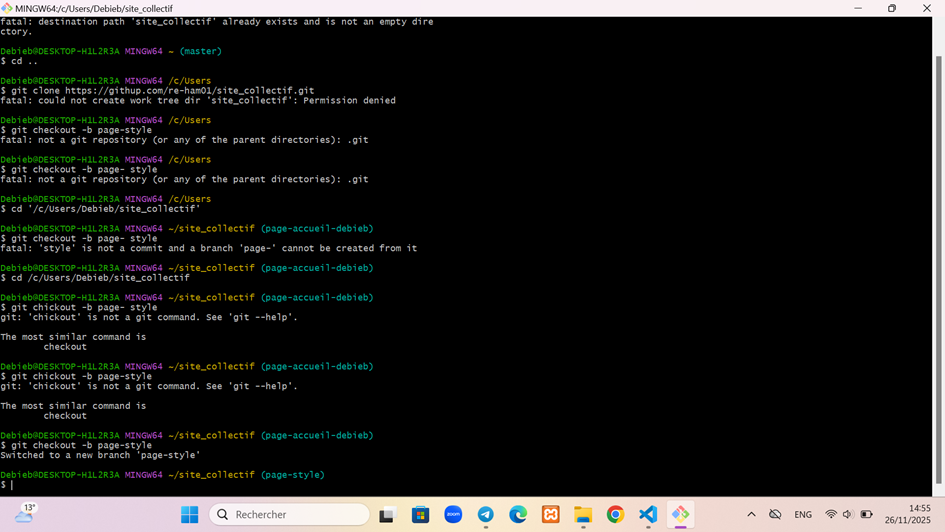
Clone <https://githup.com/ri-ham01/site_collectif.git>

Il s’est avere qu’un dossier site\_collectif existait déjà.



Cette photo montre l’explorateur de fichiers (File Explorer) sous Windows, ou il apparait dans le dossies du projet site\_collectif lui-meme situe dans le dossier utilisateur(c:\utilisateurs\debieb).cela confirme que le processus de clonage a reussi et que le dossier est prêt a etre utilise.

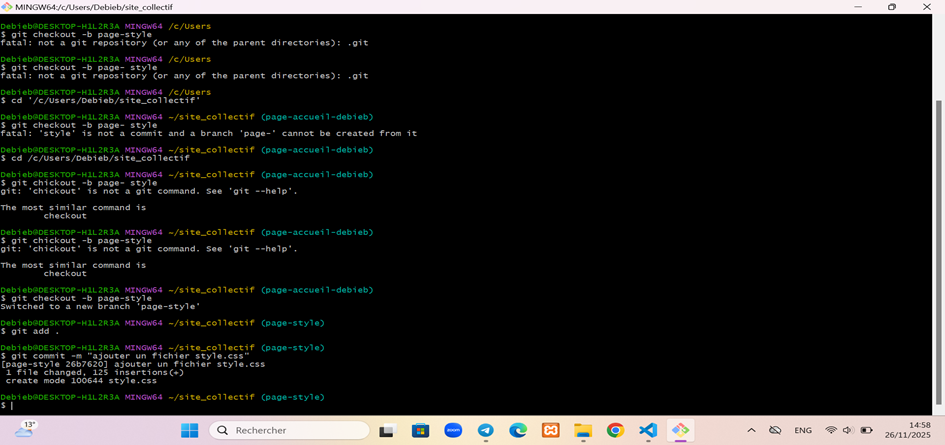
**2-Branching :**



Apres m’etre connecte au dossier du projet, j’ai cree une branch Style.Css en utilisant la commande git checkout -b page-style.

Et le changement a donc été effectue le passage a une nouvelle branche a donc été effectue Page-Style.

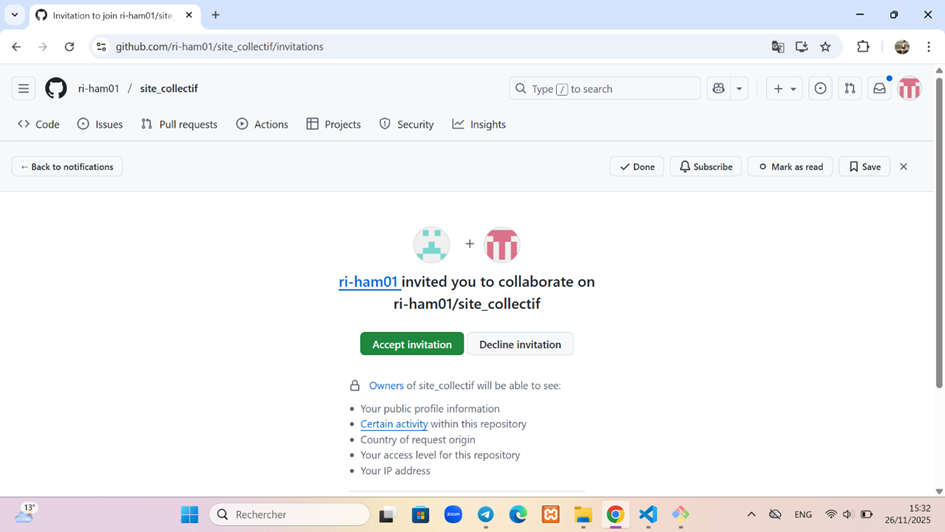
1. **Modification :**

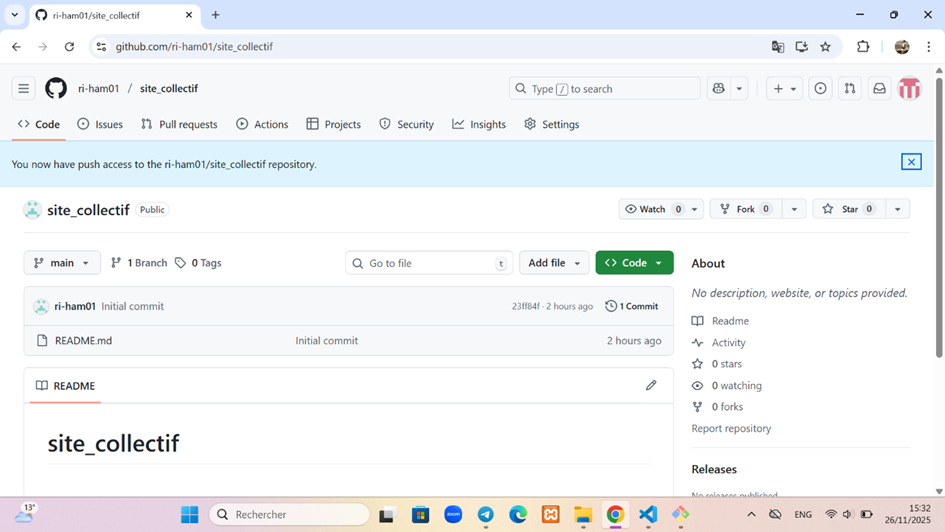
******

J’ai ajoute un fichier css qui contient les modification Style et je l’ai enregistre comme un Commit dans la branche Page-Style.

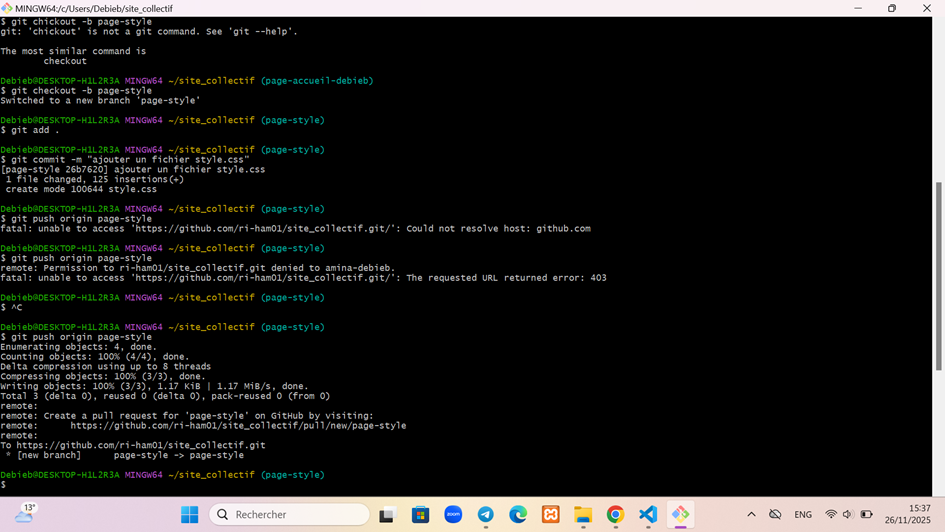
**3-Push :**

Avant de tenter de pousser vers le dépôt distant, je dois avoir la permission d'écrire dans le dépôt GitHub du collègue (ri-ham01/site\_collectif).



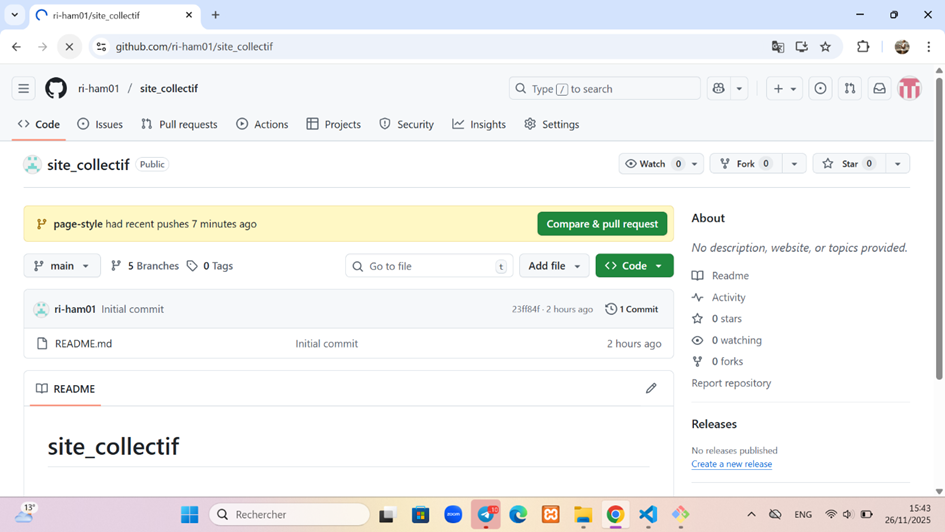


L'image montre que j'ai l'autorisation Push sur l'entrepôt.

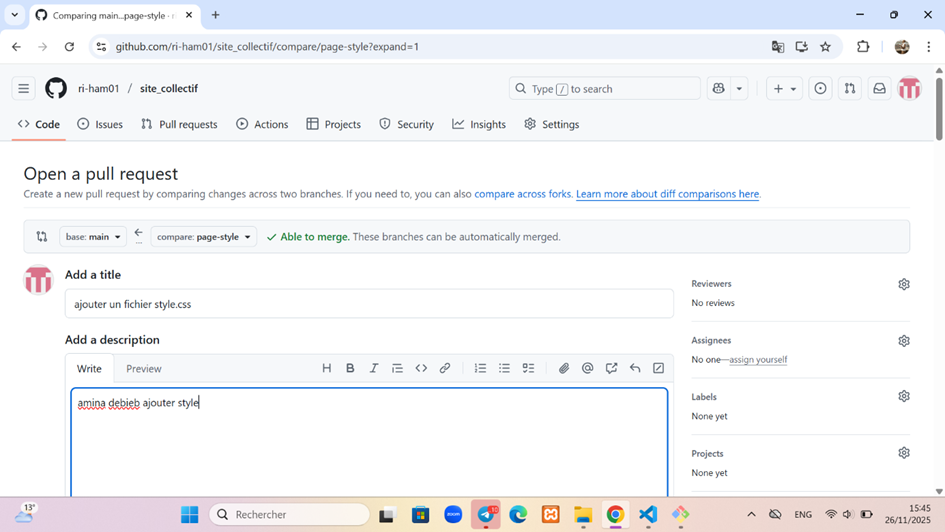


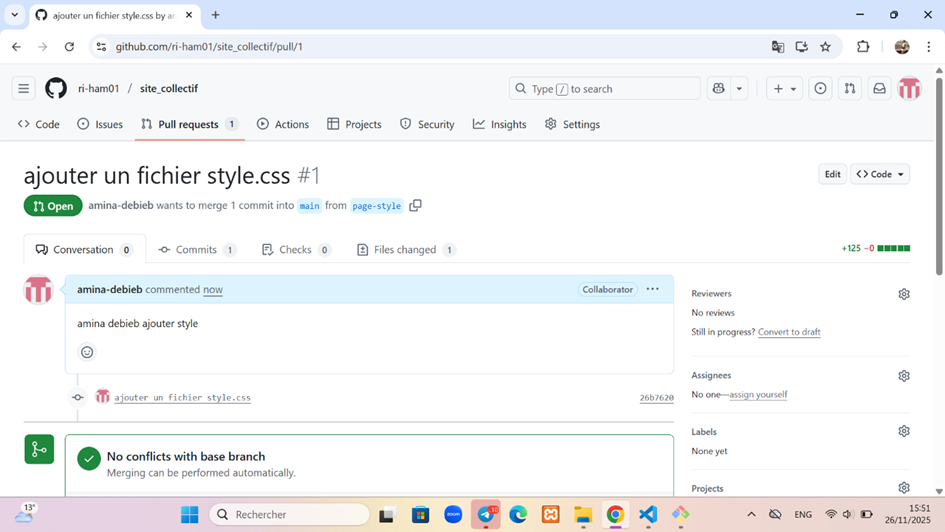
J'ai transféré les modifications de votre branche locale (style page) vers le dépôt distant sur GitHub. Cela indique que la branche est désormais disponible sur GitHub.

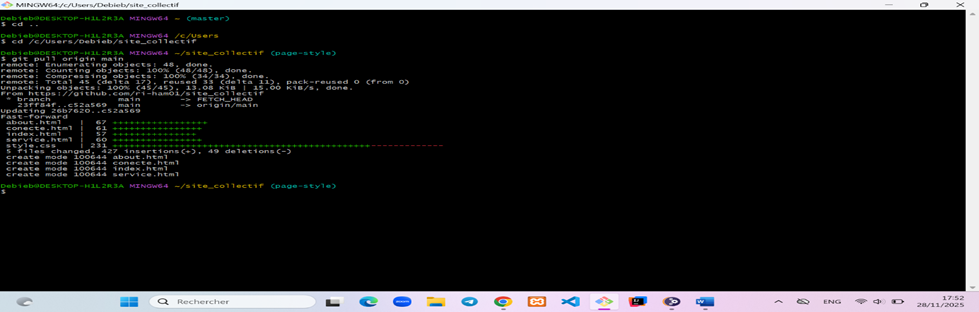
**4-Pull Request :**



Après le chargement, une notification apparaît sur la page d'accueil du dépôt GitHub pour créer une demande de fusion Pull Request pour la branche « page-style ». J'ai cliqué sur le bouton Compare & Pull request.





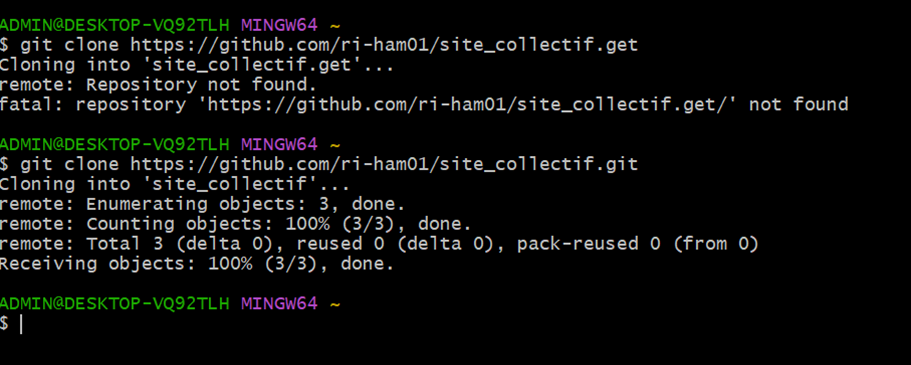
**5- Pull :** 

**Service.html**

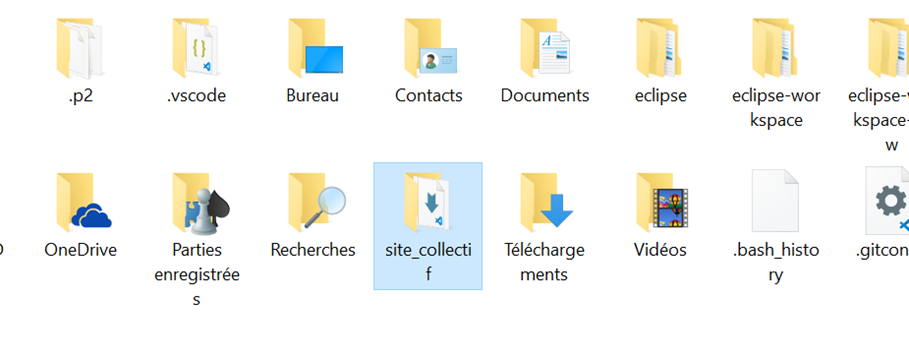
1/configuration de l`identité de l`utilisateur Git pour les futurs commits .



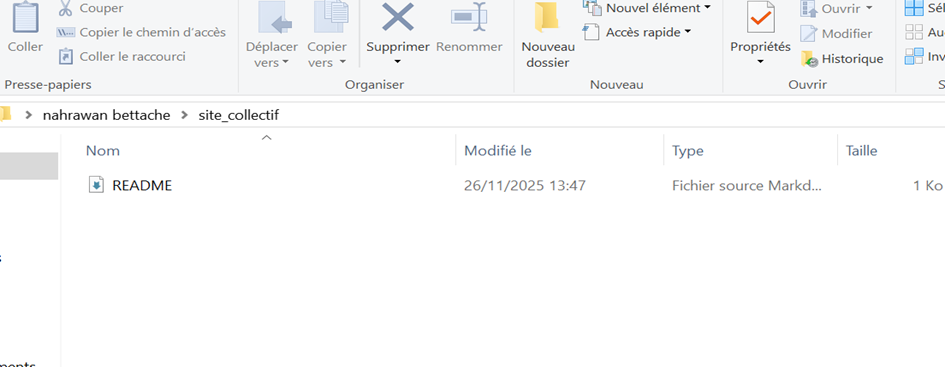
2/-utilisation de la commande git clone pour obtenir une copie locale du projet « site\_collectif » .



3/- vérification du répertiore locale créé par le clonage.



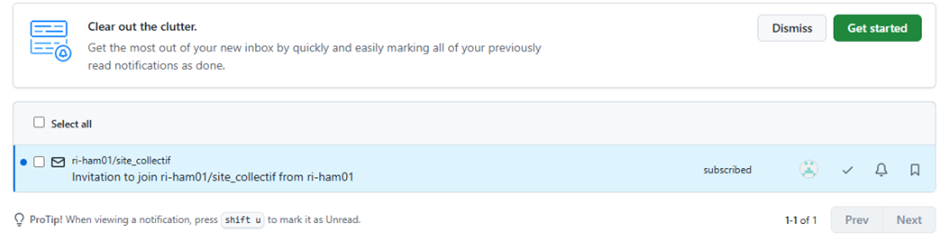
4/-exploration de contenu initial du dépôt.



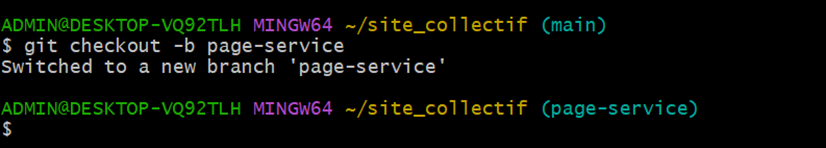
5/-utilisation de la commande cd pour se positionner dans le répertoire du dépôt « site-collectif » .



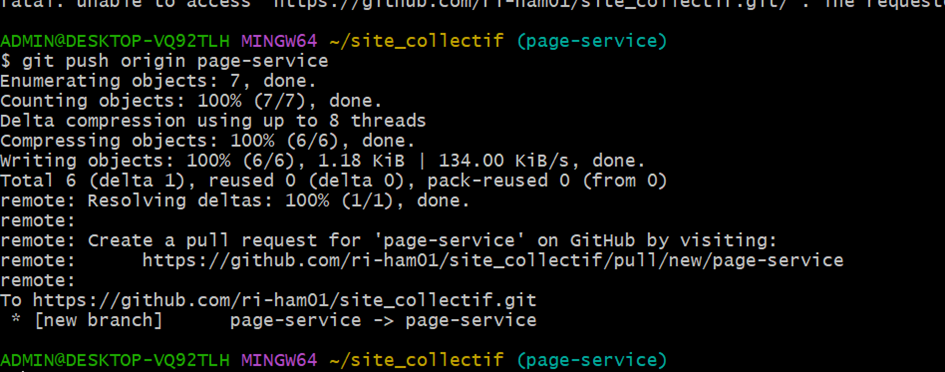
6/-réception de l`invitation a rejoindre un dépôt distant sur GitHub.



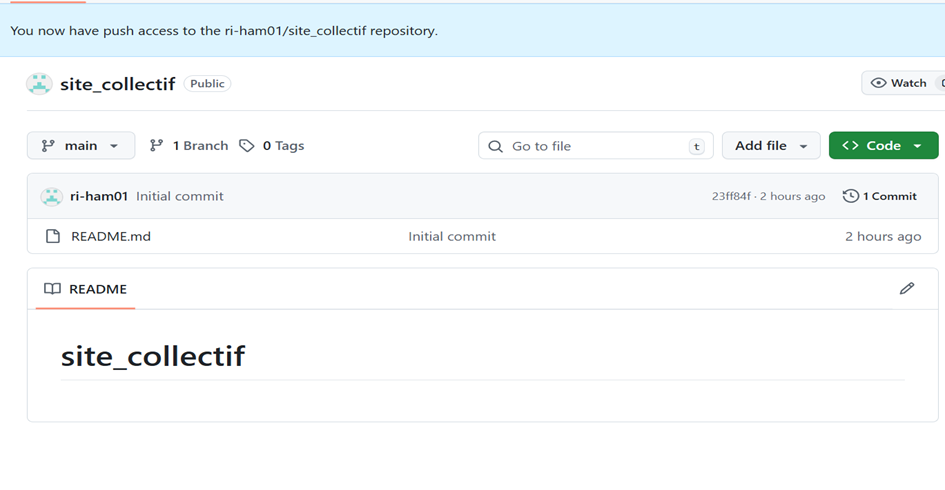
7/-création et basculement réussi vers une nouvelle branche «page -service »



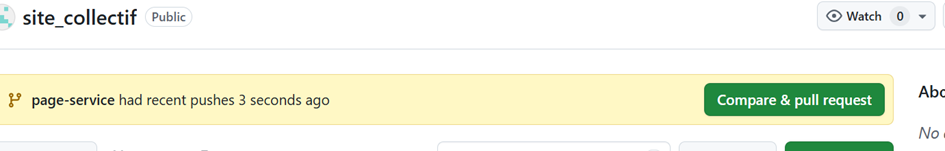
8/-déploiement effectif(push) du nouveau travail vers le dépôt distant .



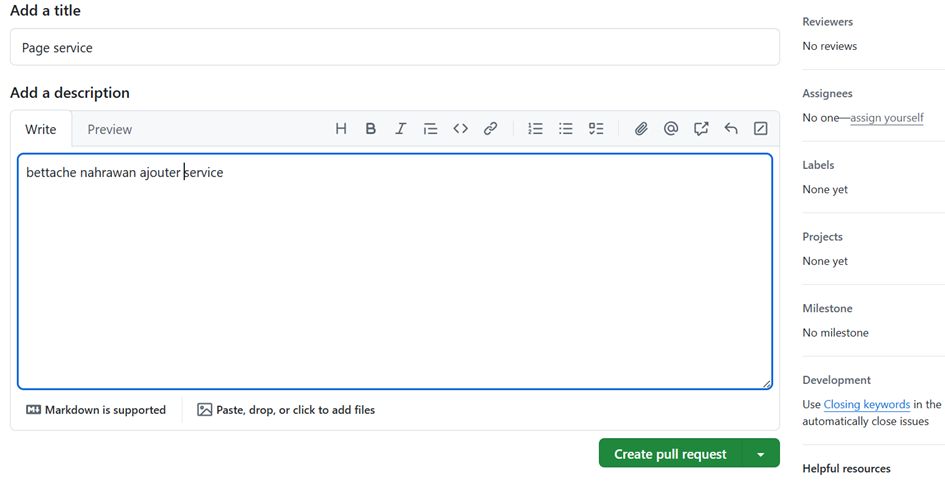
9/-confirmation de l`accès et vérification de l`état du dépo distant.



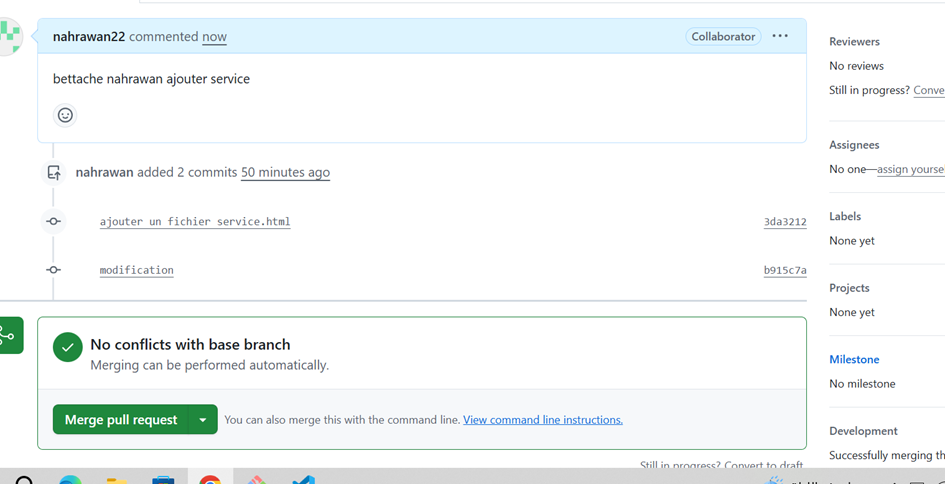
10/-publication (push) réussi de la nouvelle branche sur le dépôt distant.



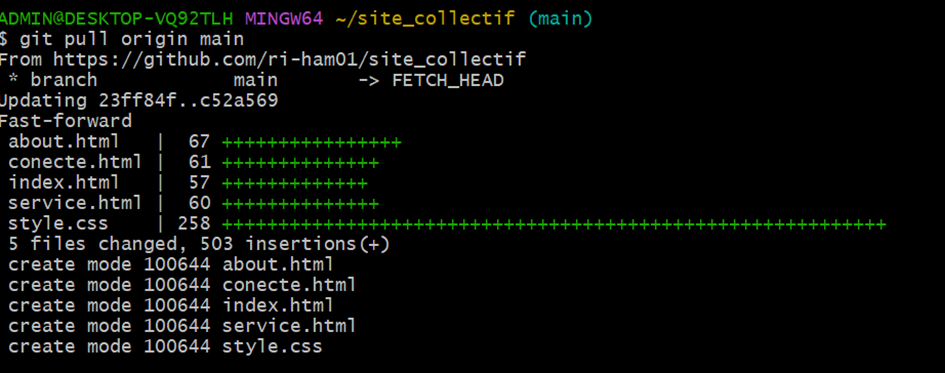
11/-ouverture d`une demande de fusion (pull request) pour intérger les modification .



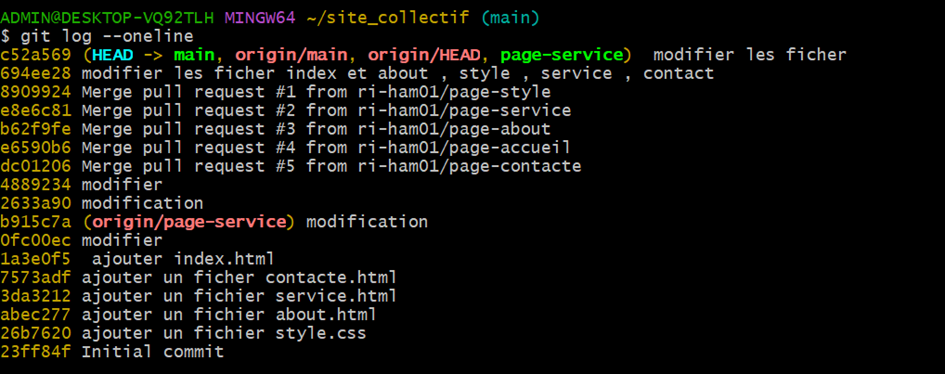
12/-vérification de l`absence de conflits e préparation a la fusion (merge).



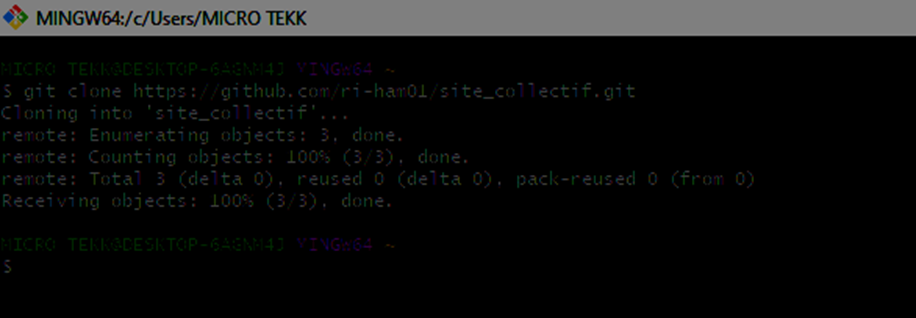
13/-mise a jour du dépôt local pour récupérer la derniére version



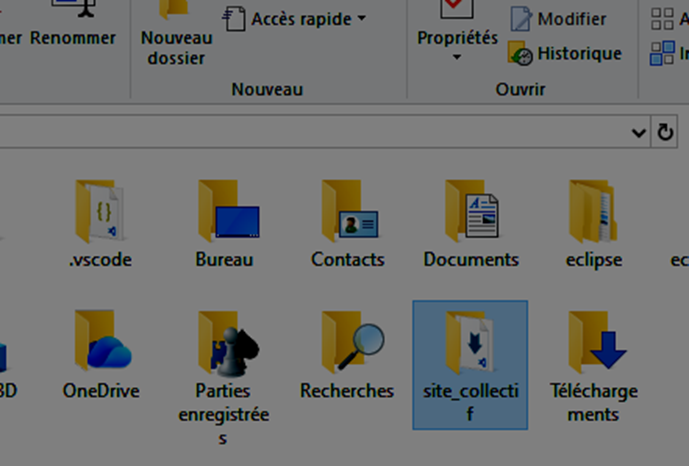
14/-consultation de le résultat des commits sur la branche principale.



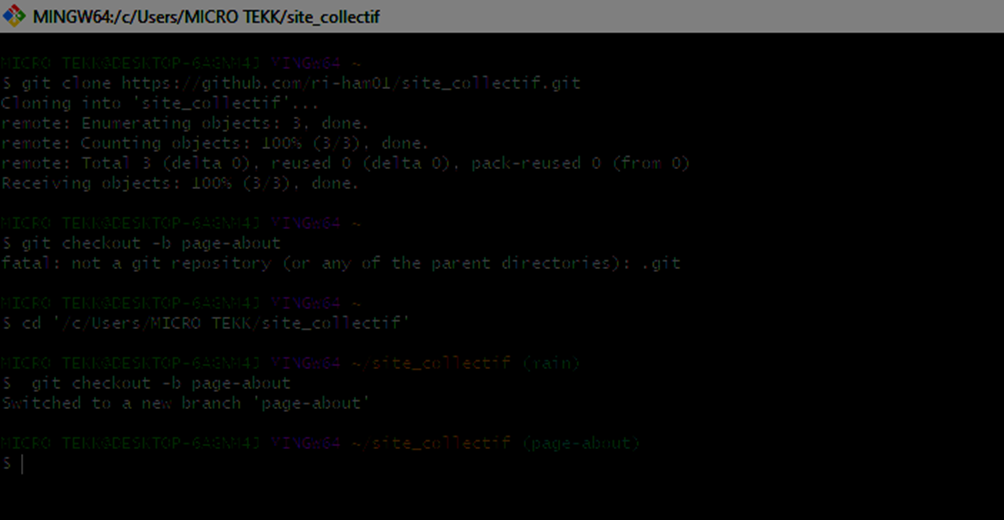
**About.html**

****

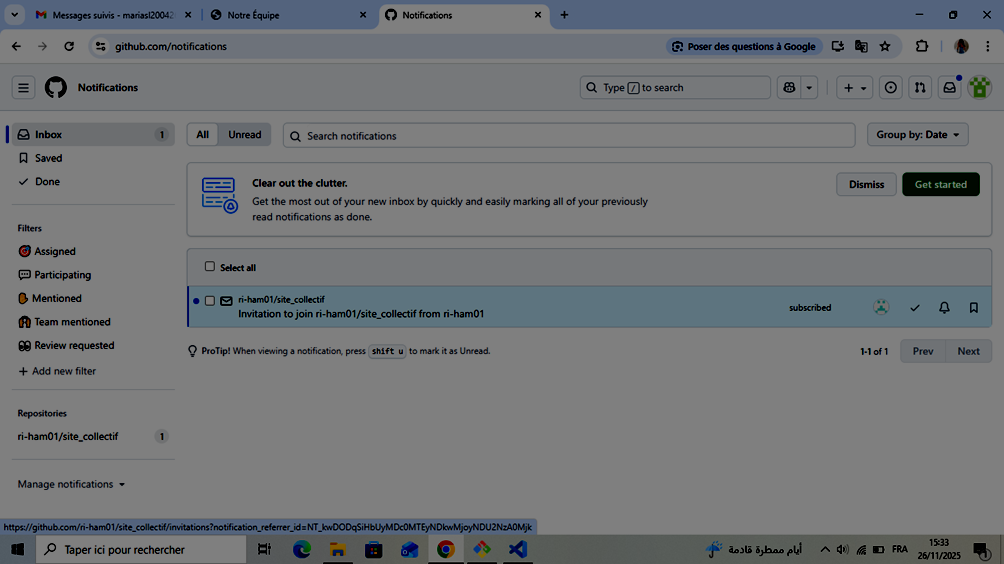
**1/L’opération de clonage du projet :** L’instruction git clone a été utilisée pour télécharger une copie de travail compléte du dépôt distant site\_collectif hébergé sur GitHub, en préparation du démarrage du travail local sur les fichiers du projet et de leur synchronisation ultérieure.



**2/Dossier du projet:** Suite à l’opération git clone, le dossier site\_collectif est apparu sur le disque local. Il représente désormais le dépôt local du projet et contient tous ses fichiers pour commencer les modifications.



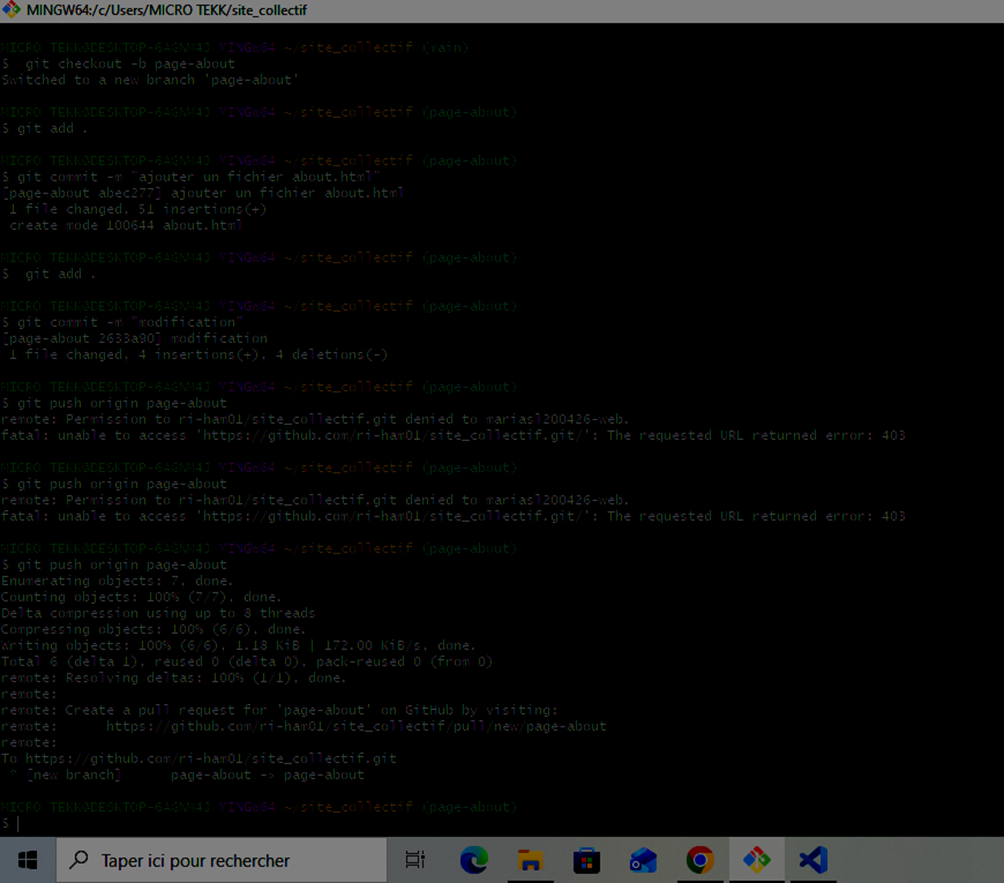
**3/Création d’une nouvelle branche :** La commande git checkout -b page-about a été utilisée pour créer et basculer immédiatement vers une nouvelle branche de travail nommée page-about. Cela me permet de développer les fichiers de la page « about »isolément du flux principal (main), garantissant la stabilité du projet avant l’intégration des changements.



**4/Adhésion à l’équipe de projet :** J ‘ai reçu une notification d’invitation via GitHub pour rejoindre le dépôt partagé ri-ham01/site\_collectif. L’acceptation de cette invitation confirme mon statut de « contributrice » dans l’équipe, m’autorisant à pousser (Push) mes changements et à les intégrer au projet commun.



**5/Interface du dépôt sur GitHub** : cette image illustre la page d’accueil du dépôt partagé ri-ham01/site\_collectif sur la plateforme GitHub .il est clairement indiqué que le dépôt contient une seule branche (1 branch) , et que le dernier enregistrement (commit) effectué était le « initail commit » ,confirmant ainsi que j’ai accédé et vérifié l’état initial du projet avant de commencer mes propres modifications.



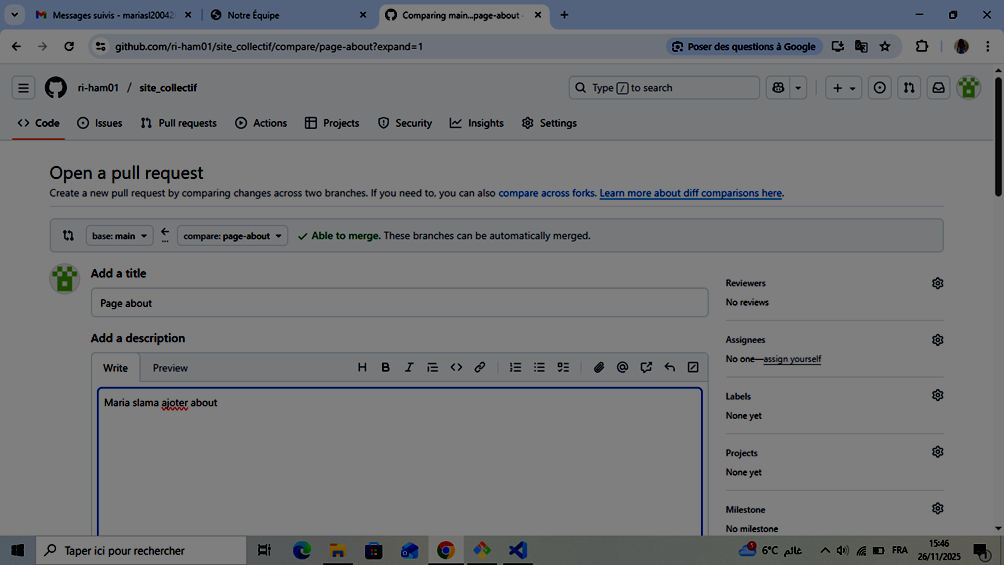
**6/1/Préparation de l’Environnement de travail sur une nouvelle branche** : J’ai commencé par isoler mon nouveau travail du code principal du projet. Pour cela, j’ai procédé comme suit :

•J’ai créé et basculé vers une nouvelle branche spécifiquement dédiée à cette fonctionnalité, que j’ai nommée page-about.

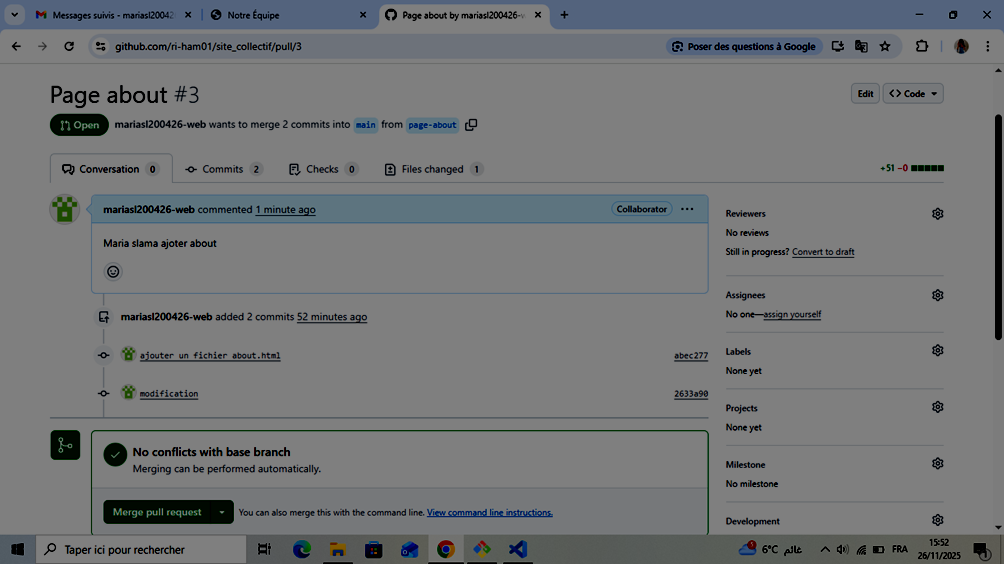
**2/Enregistrement des modifications locales (commit) :**

J’ai enregistré mes modifictions localement en deux temps : un premier commit pour l’ajout du fichier about.html, suivi d’un second commit intitulé simplement « modification » pour les ajustements ultérieurs.

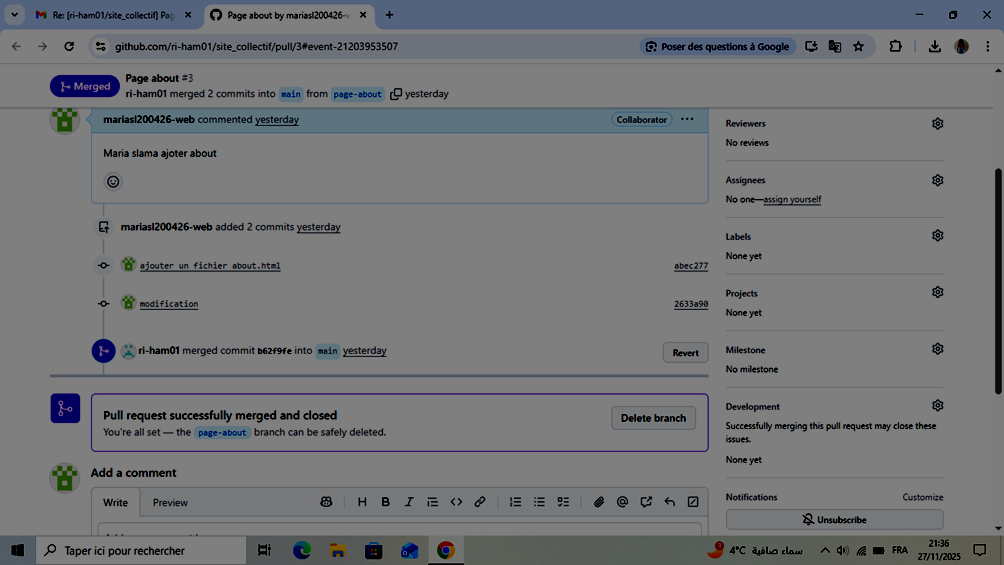
**3/Publication en ligne (push) :** J’ai ensuite poussé la branche page-about et tous ses commits vers le dépôt distant sur GitHub. L’opération a été un succés, mes changements sont désormais en ligne, et je suis prét à créer la pull request nécessaire à l’intégration.



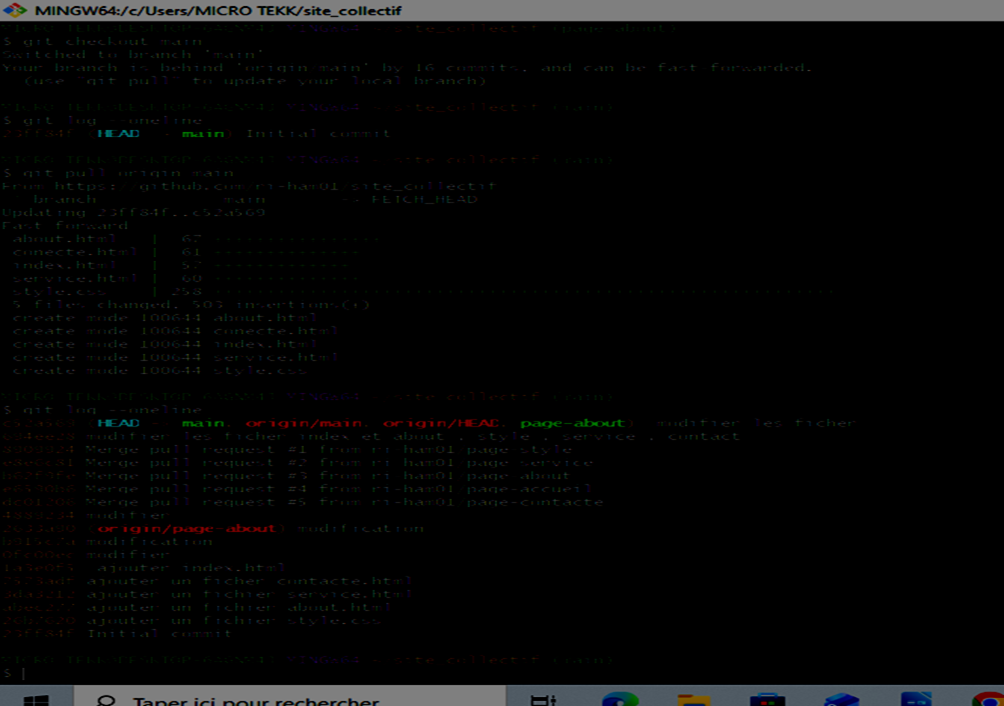
7/ J’ai suis actuellement sur l’interface GitHub, en train de créer un pull request (demande de fusion). Je compare ma branche de travail, page-about avec la branche principale main. Le système confirme que la fusion automatique est possible (« able to merge ») sans conflits.



8/ J’ai fini mon travail sur la branche web (en 2 commits), et j’ai créé cette demande (pull request) pour que mon code soit fusionné dans la branche principale.



9/ Mon travail a été fusionné (merge) avec succès.



10/ J’ mis à jour ma branche de travail (main) avec la dernières modifications du dépôt distant en utilisant la commande git pull origin main. Les changements ont été intégres avec succès et sans conflits.

1. **Conclusion et Synthèse**

À travers ce rapport, un cycle de travail de développement complet utilisant Git a été documenté, incarnant les bonnes pratiques du travail collaboratif sur les logiciels. Ce processus met en lumière plusieurs points essentiels :

1. Efficacité : Le système de branche de Git permet l'isolation des fonctionnalités, permettant à plusieurs équipes de travailler efficacement sans interference.
2. Contrôle et Responsabilisation : Le mécanisme de Pull Request fournit une couche de revue et de discussion avant la fusion, garantissant la qualité du code.
3. Traçabilité : Git conserve un historique complet et détaillé de chaque changement, identifiant la personne responsable, offrant une transparence totale.
4. Gestion des Versions : Les tags permettent une gestion organisée des versions logicielles, facilitant le déploiement et le retour à des états stables si nécessaire.
5. Ce workflow méthodique constitue une base solide pour tout projet logiciel collaboratif, en assurant sa pérennité, sa maintenabilité et la capacité de l'équipe à monter en charge.