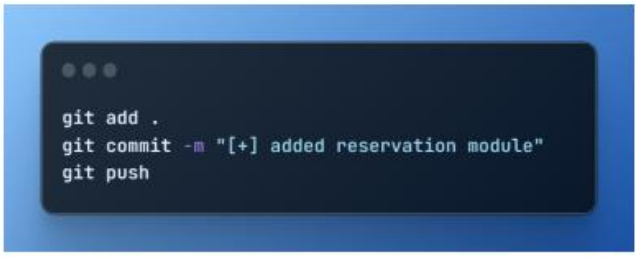
AP 1ere année OBarbeuc

Gérer le patrimoine informatique

Exploiter des normes et référentiels :

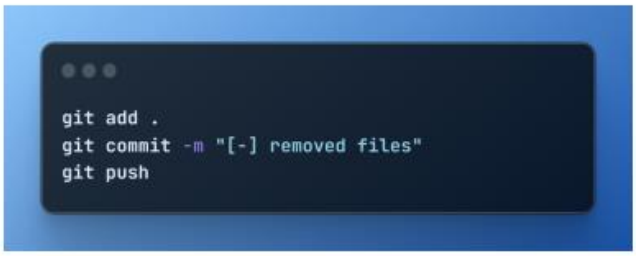
Lors de notre projet de restauration en AP, on a pu utiliser des normes, notamment lors de changements dans les codes quand nous envoyons nos modifications

Pour les ajouts de code :



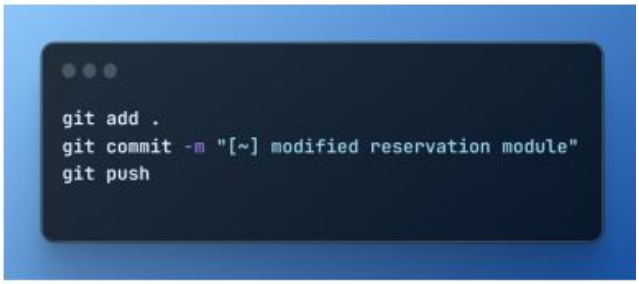
```
git add .
git commit -m "[+] added reservation module"
git push
```

Pour les suppression de code :



```
git add .
git commit -m "[-] removed files"
git push
```

Pour les modification de code :



```
git add .
git commit -m "[~] modified reservation module"
git push
```

Nous avons également utilisé les normes Pascal et Camel.

Pour les fonction et procedure : première lettre du premier mot en minuscule
ensuite première lettre des autres mots en Majuscule

Pour les classes : Première lettre en majuscule

Développer la présence en ligne de l'organisation

Bien que cela était fictif, nous avons pu simuler la mise en ligne d'un site web de réservation pour l'organisation de restauration de Gaston Berger.

Travailler en mode projet

La méthode d'organisation sur GitHub pour le projet Obarbeuc consiste à utiliser les fonctionnalités intégrées de GitHub pour organiser et gérer le développement du site web de manière efficace. Pour ce faire, nous avons utilisé les outils suivants : Les issues : pour identifier les problèmes, les bugs, les demandes de fonctionnalités et les améliorations à apporter au projet. Les issues ont été attribuées aux membres de l'équipe pour qu'ils puissent travailler sur les tâches correspondantes.

Planifier les activités :

Nous avons pu dans un premier temps planifier les activités nécessaires à la réalisation du projet et les attribuer aux différents membres du groupe.

Planning détaillé

A partir du 1er février :

-Établir le planning détaillé du projet (jalon 2) : 1 séance → Samuel | Rédouane ✅ (séance du 1er février)

-Établir la structure du site (jalon 2) : 1 séance → Rédouane |Yannis ✅ (séance du 1er février)

-Réaliser le site vitrine du restaurant (jalon 3) : 2/3 séances → Yannis | Samuel

-Concevoir la politique RGPD du site (jalon 3) 1 séance → Yannis | Samuel | Rédouane

-Effectuer le suivi du planning (jalon 3) : 1 séance → Samuel | Rédouane

-Schématiser la base de données (jalon 4) : 1 séance → Yannis|Rédouane

-Créer la base de données avec un jeu de d'essai (jalon 4) : 2 séances → Samuel|Yannis

-Réaliser des maquettes pour chaque fonctionnalité (jalon 4) : 2 séances → Rédouane | Yannis

-Effectuer le suivi du planning (jalon 4) : 1 séance → Samuel | Rédouane

-Rééditer la base de données et corriger les potentielles erreurs (jalon 5) : 1 séance → Samuel|Yannis

Fonctionnalités (Jalon 5) : 1 séance

Mettre à disposition des utilisateurs un service informatique

Réaliser les tests d'intégration et d'acceptation d'un service :

Lors de la réalisation du site de restauration de Gaston Berger nous avons pu mettre en place des tests unitaires permettant de s'assurer du bon fonctionnement du site.

Un test est divisés en deux parties :

- SetUp : Initialise les méthodes et les données à tester
- Run : Lance le test et le compare avec le résultat attendu.

Un test est en fait une classe personnalisé composé de deux méthode qui hérite de la classe TestCase

```python
class TodayLunchTestCase(TestCase):
    def setUp(self):
        TodayLunch.objects.create(title="Cheeseburger", price=35)
        TodayLunch.objects.create(title="SioBurger", price=25)
        TodayLunch.objects.create(title="BigBurger", price=10)


    def test_true_price(self):
        todayLunch_0 = TodayLunch.objects.get(title="Cheeseburger")
        self.assertEqual(todayLunch_0.getPrice(), 35)
    ....
```

Les tests ont été organisés dans un fichier nommé test.py.

Voici un exemple d'un test unitaire réalisé

```python
class UrlTestCase(TestCase):
    def test_url(self):
        response = self.client.get('/reservation')
        self.assertEqual(response.status_code, 200)
```

J'essaye d'accéder à la page réservation sans être connecté ce qui va me bloquer et me renvoyer en code de redirection 302.

```
======================================================================
FAIL: test_url (main.tests.UrlTestCase)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\chill\Desktop\dev\OBarbeuc\obarbeuc\main\tests.py", line 12, in test_url
    self.assertEqual(response.status_code, 200)
AssertionError: 302 != 200

----------------------------------------------------------------------
Ran 1 test in 0.033s
```

Comme nous le voyons avec "AssertionError: 302 != 200"
La page nous a renvoyé le code 302 au lieu du code 200 qui était attendu.
