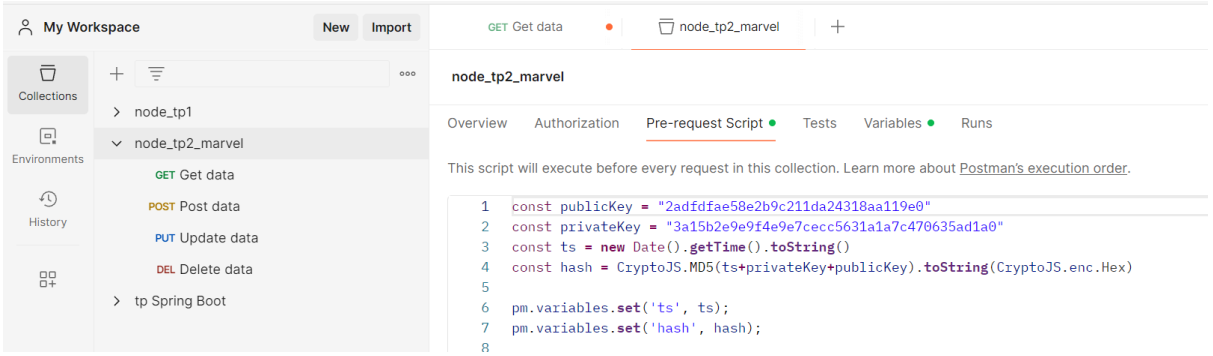
# Compte-rendu tp 2

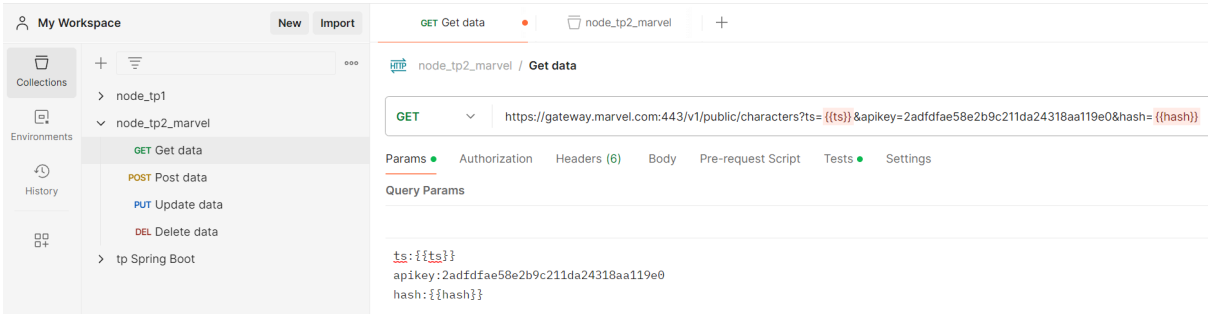
Ctrl+Alt+L ⇒ reformater

**les explications nécessaire pour comprendre démarche**

- Pour précalculer le hash dans Postman, c'est dans pre-request-script de la collection




- Comment les utiliser




- Afficher une image à partir de l'URL imageUrl ⇒ obtenir une URL complète
imageUrl: `${character.thumbnail.path}/portrait_xlarge.${character.thumbnail.extension}`

- Utilisation de fetch

```
url='https://gateway.marvel.com:443/v1/public/characters'
const response = await
fetch(`${url}?ts=${timestamp}&apikey=${publicKey}&hash=${hash}`);
//${variable}
```

ou

```
const reponse = await fetch("http://example.com/films.json")
```

- Rediriger la page vers l'index avec la variable characters
res.view('../templates/index.hbs', {characters})

- Fichier .hbs avec des variables entre {{ }}

- Commandes Docker :
    - docker rm -vf $(docker ps -aq) // Supprimer tous les conteneurs
    - docker rmi -f $(docker images -aq) // Supprimer toutes les images
    - docker build . -t marvels // Construire une image nommée "marvels"
    - docker run -p 3000:3000 marvels // Exécuter

**vos choix, ce qui a marché facilement**
- Au début, l'utilisation de express() est rapide et pratique pour effectuer des requêtes HTTP.

- Fastify est bien adapté pour l'affichage de pages web en HTML

**vos difficultés, comment vous les avez surmonté**
1. Problème d'affichage sur la page web
⇒ Oubliez le "return" avant res.view

2. Après build en « multistage », la taille de l'image n’a rien changé
⇒ Je n'ai toujours pas réussi. J'ai essayé pendant une demi-journée. Je pense qu'il faut peut-être installer certaines dépendances npm pour réussir le build.

**qu'est qu'on pourrait améliorer**
- styles de la page HTML
- taille de l'image Docker, qui reste grande