# Rapport TP

1. **Quels changements cela a-t-il apporter dans les répertoires/fichiers de notre projet ?**

Lors de l’exécution de la commande, on remarque que dans notre package.json, un module a été rajouté automatiquement pour husky. On remarquera également que un fichier node\_modules c’est créé, ainsi qu’un fichier .husky, comportement les éléments dont un autre package.json.

On notera également qu’un package\_lock.json c’est créé à notre racine du projet.

1. Une image contenant texte, capture d’écran, Police

   Le contenu généré par l’IA peut être incorrect.**Tenter de faire un commit et indiquer ce que vous constatez**

On remarque donc que l’on ne peut pas commit, et que le message affiché et celui de ‘test’ dans notre package.json

1. **Modification du .husky et du contenu de pre-commit**

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.Maintenant, lorsque nous exécutons un commit, nous ne lançons plus le message de test du package, mais nous affichons simplement le echo que nous y avons mis, avec le message ‘je m’exécute avant un commit ‘

Resultat après commit :

VOIR EN DESSOUS 🡪

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.On voir bien maintenant que le message est exécuté et afficher avant de réaliser le commit, sans erreur cette fois, car on n’exécute plus le npm test qui affichait le message d’erreur.

1. **Quels changements cela a-t-il apporter dans les répertoires/fichiers de notre projet ?**

Après exécution des deux commandes, on remarque donc la création de deux nouveaux fichiers :

Une image contenant texte, Police, nombre, capture d’écran

Le contenu généré par l’IA peut être incorrect.Dans les node\_modules :

Une image contenant texte, Police, capture d’écran

Le contenu généré par l’IA peut être incorrect.Dans les fichiers racine du projet :

1. **Quels changements cela a-t-il apporter dans les répertoires/fichiers de notre projet ?**

VOIR EN DESSOUS 🡪

Une image contenant texte, capture d’écran, Police, logiciel

Le contenu généré par l’IA peut être incorrect.

Cela m’a créé un fichier temporaire avec mon historique de mes changelogs dans mon fichier .changeset

1. **Générer le changelog automatiquement : quels changements cela a-t-il apporter dans les répertoires/fichiers de notre projet ?**

Je remarque premièrement que cela m’a supprimé mon ancien fichier temporaire qui contenais mon historique de changelogs.

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.**Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.**Je remarque également la création d’un nouveau fichier intitulé CHANGELOG.md à la racine de mon projet avec le contenu suivant :

Ce sera donc mon fichier définitif de mon historique de tous mes changelogs

1. **Expliquer ce que chaque ligne fait (PRETTIER)**

VOIR PLUS BAS 🡪

**"printWidth": 80,**  
→ Définit la largeur maximale d’une ligne de code avant que Prettier ne la découpe sur plusieurs lignes (80 caractères ici, ce qui est une convention courante pour améliorer la lisibilité du code).

**"tabWidth": 2,**  
→ Définit le nombre d’espaces utilisés pour une indentation (ici 2 espaces par niveau d’indentation).

**"useTabs": false,**  
→ Spécifie si Prettier doit utiliser des tabulations (true) ou des espaces (false, ici, il utilisera des espaces).

**"semi": true,**  
→ Indique si Prettier doit ajouter un point-virgule (;) à la fin de chaque instruction JavaScript (true signifie qu’il les ajoutera).

**"singleQuote": true,**  
→ Spécifie l’utilisation des guillemets simples (') au lieu des guillemets doubles (") pour les chaînes de caractères.

**"trailingComma": "all",**  
→ Ajoute une virgule finale à la fin des listes et objets multi-lignes.

**"bracketSpacing": true,**  
→ Ajoute un espace entre les accolades et leur contenu.

**"arrowParens": "always",**  
→ Force l’utilisation des parenthèses autour des paramètres des fonctions fléchées, même lorsqu’il n’y a qu’un seul paramètre.

**"endOfLine": "crlf"**  
→ Définit le type de retour à la ligne utilisé dans les fichiers formatés.

* "lf" (Linux/macOS)
* "crlf" (Windows, ici choisi)  
  Cela garantit une cohérence du format des fichiers selon le système utilisé.

1. **Exécuter Prettier pour formater le code : décrivez et détaillez les changements observés dans votre rapport.**

J’ai remarqué que le fichier coucou.js avait été légèrement modifier, avec des espaces, ou des guillemets changés également :

VOIR PLUS BAS 🡪

Une image contenant texte, capture d’écran, Police, logiciel

Le contenu généré par l’IA peut être incorrect.

Notre fichier de configuration .prettierrc.json est en fait la pour formatter le code selon des paramètres mis au préalable. La commande npm run format permet de formatter les fichiers selon les paramètres enregistrés.

1. Une image contenant texte, capture d’écran, Police

   Le contenu généré par l’IA peut être incorrect.**Exécutez la commande npm run knip et décrivez ce que vous voyez :**

On peut voir que la commande permet de retourner tous ce qui ne sert à rien dans les codes, les fichiers qui ne sont pas utilisés, les dépendances inutiles etc.

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.UNE FOIS LES ELEMENTS INUTILES SUPPRIMES :

Une fois que l’on a supprimé tout ce qui était inutile, knip ne trouve aucune issues d’erreurs, donc il nous retourne que tout vas bien.