Workflow Dedicate

1 – Trello

Screens screenshot of a computer screen

Description automatically generated with medium confidence

Les grosses tâches à faire se situent dans la partie « Epic »

A picture containing text, screenshot, software, web page

Description automatically generated

Choisir le modèle de carte qui correspond au type de tâche que l’on veut effectuer

A screenshot of a computer

Description automatically generated with medium confidence

Dans le cas où la tâche « enfant » est créée après la « parent », ajouter celle-ci directement depuis la carte « enfant »

N’oubliez pas d’assigner la personne correspondante (souvent vous-même à la tâche en question).

Mettre aussi la tâche une fois finie dans la liste correspondante.

2 – Code

2.1 – Branche sur Git/Github

Lorsque vous créez votre branche pour la feature ou la task en question, le nom de la branche utilise cette nomenclature :

**{numéro de tâche}-{nom de la tache}**

2.2 – Commit

A screen shot of a computer

Description automatically generated with medium confidence

Utilisation de la commande **yarn commit** ou **npm run commit.**

Il suffit de suivre les différentes étapes.

Cela va lancer la commande avec un package: **git-cz**, cela permet d’avoir une syntaxe logique et cohérente dans les différents commit sur github.

2.3 – PR

2.3.1 – Rebase

Si vous avez du code en retard par rapport à l’actuelle branche dev, pour vous mettre à jour vous avez 2 possibilités différentes :

* En restant sur votre branche, vous faites :
  + Git pull origin dev --rebase
  + Possible conflits à résoudre
* Si vous craignez de faire une bêtise :
  + Git checkout dev
  + Git pull
  + Git checkout [votre branche]
  + Git rebase origin/dev

Au moment de push vous allez avoir une erreur, il faut force le push dans ce cas-là

à push –force

à les différences majeures entre le rebase et le merge : <https://codeur-pro.fr/difference-entre-git-merge-et-git-rebase/>

2.3.2 – Github

A screenshot of a computer

Description automatically generated with medium confidence

Une fois que votre code est push et rebase, le but est de faire un PR en faisant une comparaison entre la base « dev » et votre branche.

Il est intéressant de mettre des reviewers à votre PR pour que d’autres personnes puissent voir le code et comprendre ce que vous avez fait voir même vous conseiller s’il/elle voit un point d’amélioration possible.

2.3.3 – Pipeline CI

Une fois la PR demandée, la pipeline CI va se lancer et tester à nouveau le projet sur un environnement dédié, on veut que notre code fonctionne sur une autre machine que la notre.

Les différents outils :

[Husky](https://www.npmjs.com/package/husky)

permet de trigger les hooks de git, par exemple ici on trigger le hook « pre-commit » qui permet de lancer une commande avant de commit.

[Lerna](https://lerna.js.org/)

C’est un gestionnaire de monorepo, cela nous permet de gérer le repos et c’est différents services qui sont situés dans le dossier « packages ».

Il va faire tous les tests situés dans tous différents packages.

[Git-cz](https://www.npmjs.com/package/git-cz)

[Eslint](https://eslint.org/)

Utilisation des règles [airbnb](https://www.npmjs.com/package/eslint-config-airbnb). Un linter permet d’avoir des règles de syntaxes strictes et faire en sorte que tout le monde travaille de la même manière.