1 Création d'un boilerplate

Quand vous créez une application vous pouvez l'architecturer de la façon suivante :

- Vous créez le dossier de l'application avec

```
mkdir nom_de_votre_application
```

- Vous entrez dans le dossier

```
cd nom_de_votre_application
```

- Vous ouvez tout avec vscode

```
code .
```

- Vous créer les dossiers et fichiers qui doivent se trouver à la racine et initialiser le dépôt git et le package.json ainsi que les fichiers qui permettent d'exclure des dossiers.

```
mkdir src assets
touch index.html main.js styles.css
git init
npm init -y
touch .gitignore .prettierrc
echo "node_modules" >> .gitignore
echo "{}" >> .prettierrc
npm pkg set scripts.format="npx prettier --write ."
```

- Une fois créer les fichiers devant se trouver à la racine vous pouvez vous occuper des autres Nous n'avons pas encore utiliser le dossier utils ni le dossier service mais cela va arriver. Quant au dossier views il correspond à l'idée des pages

```
cd src
mkdir js css
cd js
mkdir components pages layouts services utils containers
cd components
touch button.js input.js anchor.js nav.js paragraph.js
    form.js
cd ..
cd pages
touch home.js login.js register.js
cd ..
cd layouts
touch header.js
```

```
cd ..
cd services
touch AuthService.js NotificationService.js
    LocalStorageService.js UsersService.js ProductsService
    .js PaymentService.js CartService.js
cd ..
cd containers
HomeContainer.js RegisterContainer.js LoginContainer.js
    PaymentContainer.js CartContainer.js
cd ..
```

```
cd css
mkdir components pages utils layouts base
cd components
touch button.css input.css form.css
cd ..
cd pages
touch home.css login.css register.css
cd ..
cd utils
touch keyframes.css
cd ..
cd layouts
touch header.css
cd ..
cd base
touch base.css variables.css fonts.css
```

2 Quelques commandes sh

Le langage sh est le langage de programmation utilisé pour communiquer dans des terminaux avec le système.

Quelques commandes sont importantes :

La commande cd permet de vous placer dans le dossier de l'utilisateur courant La commande cd / permet de vous placer dans le dossier racine La commande cd Desktop si vous êtes bien placé(e) vous permettra d'entrer dans le dossier Desktop

La commande ls -a vous permet de voir la liste des dossiers, fichiers et liens (même cachés) où vous vous trouvez.

La commande ls -l vous permet d'avoir plus d'informations sur les différents éléments présents dans un dossier notamment les permissions (en lecture, en écriture et en exécution)

La commande clear vous permet comme son nom peut le laisser entendre de nettoyer le terminal mais en réalité il mets cache simplement les inputs et ouputs précédents au-dessus.

La commande history vous permet de voir la liste des commandes précédemment tapées.

La commande mkdir permet de créer un dossier par exemple mkdir ChezElle permet de créer le dossier ChezElle

La commande touch permet de créer un fichier

La commande chmod permet de changer les droits sur élément Par exemple rendre exécutable pour tous le fichier boiler.sh on tapera : chmod +x boiler.sh

La commande cat permet de visualiser le contenu d'un fichier Par exemple cat boiler.sh

La commande ./ permet d'exécuter un fichier sh Par exemple ./boiler.sh

3 Git

```
git init
git status
git remote add origin https://github.com/${username}/
    ${repo}
git remote -v

push_workflow(){
    git status
    git add .
    git commit -m $1
    git log ----oneline
    git push origin master
}

push_workflow $1
```

Explications La première commande git init initialise un dépôt git. Il s'agit de faire de votre dossier un dépôt git local qui pourra être relié à un dépôt git distant que vous aurez par exemple créé sur Github.

La liaison se fait avec la commande git remote add origin à laquelle vous ajoutez l'url du dépôt git distant.

La commande git remote -v vous permet de voir si en fetch et en push vous allez agir sur la bonne "origin"

Si jamais vous avez fait une faut d'orthographe faites un git remote remove origin et recommencez.

Ensuite quand vous travaillez dans votre application vous allez un moment donné vouloir sauvegarder votre travail en local. Donc vous allez indexer les

modifications nouvellement effectués dans le dossier puis une fois cela fait avec git add . vous allez vouloir effectuer la sauvegarde, ce qu'on appelle le commit. Donc il s'agit de faire git commit -m'feat/qqch' ce message entre apostrophes est à votre discrétion.

Personnellement lorsque c'est une mise à jour j'écris feat/qqch et quand c'est un déboggage je mets fix/qqch

Une fois que vous avez sauvegardé en local vous pouvez vérifier que le commit a été effectué en faisant un git log ou un git log –oneline pour avoir une version abrégée des informations

Ensuite une fois que vous avez tout vérifié vous allez vouloir envoyer ce commit qui représente une version a un instant t de votre application au dépôt distant (sur github)

Vous allez donc vouloir faire en sorte de pouvoir récupérer directement sur github la dernière version commitée de votre application ou de permettre à vos collaborateurs de le faire.

Vous allez donc "pusher" votre application

git push origin master

Avec cette commande vous dites que vous souhaitez que la sauvegarde locale (dernière sauvegarder) soit envoyé sur la branche master du dépôt distant.