**Projet Flink**  
  
- Création de l’application React avec :  
npx create-react-app flink

-Verfier la version pour éviter de problèmes.  
  
git add .  
git commit  
git checkout dev

- Creation de branch pour chaque collaborateur :  
git merge feature/user  
git push origin dev   
  
- Ajout de Fonts dans la page index.html  
- Création de fichier du style.css  
-Création de dossier assets pour les images  
  
-Pour télécharger les assets d’un dépôt distante en allant dans le repository, sélectionner le bouton « <> code », cliquer sur Download Zip pour télécharger les documents.  
  
- Création de pages :  
Home.jsx  
Home.css  
npm start pour lancer le programme  
  
- Pour installer les icônes de google :  
npm install @material-ui/core  
  
Pour utiliser les icones  
import {Person} from « @material-ui/core »   
  
-Création of dossier components  
  
-Création of dossier topbar dans components

-Creation of fichier topbar.jsx  
- Creation et import du fichier topbar.css   
import “./topbar.css”  
  
-import de composant Topbar  
  
Création des <div> pour les différents parts du Navbar pour le stylisation des éléments dans css  
« topbarContainer »

« topBarRight »

« topBarLeft»

« topBarCenter »

« topBarLinks»

« topBarLink»

-Création de styles pour les éléments du topBar

-Création of dossier feed, rightbar, leftbar dans components  
-Creation of fichiers .jsx et css pour chaque composant  
  
import de composants dans le fichier Home.jsx avec :

export default function Home() {

  return (

    <div><Topbar/></div>

  )

}

Fichiers à l’interieur du dossier « components » :  
-feed  
-post

-share

-sidebar

-topbar  
  
Pour importer data contenu dans notre objet dummyData et créer automatiquement les Posts on import le fichier dummyData dans notre fichier Feed.jsx  
  
import {Posts} from « ../../dummyData » ;

Et on utilise la function .map pour itérer dans notre objet et effectuer une function que pour chaque post créera un post avec sont ID et nom.

{Posts.map((p) => (

<Post key={p.id} post={p} /P>)  
{`assets/${post.photo}`}

Pour actualiser chaque post pour contenir l’information(images, textes, etc,)  
dans notre fichier Post.jsx on appel notre objet post directement avec :

{post.date}  
{post?.desc} pour créer seulement si le post contient un desc.  
  
Pour trouver l’utilisateur correspondant au post, on importe dans notre fichier post.jsx :

import {Users} from « ../../dummyData » ;

Nous créons un filtre pour comparer les IDs entre les objets Users et post, puis récupérons le username du premier utilisateur correspondant.  
  
{Users.filter((u) => u.id === post?.userId)[0].username}