# Projet

**Version du 12 avril 2021, Date limite: 24 mai 2021 à 20h**

## I) Sujet

Il s'agit de réaliser un site de micro-blogage inspiré de Twitter ou de Mastodon.

### Fonctionnalités

Le site aura un design de page unique.
Il sera accessible aussi bien par un utilisateur non authentifié qu'un utilisateur authentifié (mais pas forcément avec les mêmes fonctionnalités affichées).
Par exemple, une zone de la page contiendra des champs pour s'authentifier quand on ne l'est pas ; quand on l'est, cette zone sera remplacée par un lien vers l'édition de son profil et un lien pour se déconnecter.
La zone principale de la page web affiche (pour tout le monde) une liste de publications correspondant à des critères déjà définis (mais modifiables), affichées de la plus récente à la plus ancienne. Cette liste est mise à jour automatiquement pour prendre en compte les dernières publications ajoutées sur le serveur.
Par défaut, cette zone affiche :

- les derniers messages mentionnant `@everyone` si on n'est pas connecté,
- sinon, les messages mentionnant soit `@everyone`, soit l'utilisateur connecté, ainsi que ceux écrits par un utilisateur auquel l'utilsateur connecté est abonné.

Des contrôles disposés sur le côté et/ou au dessus permettent de changer les critères. Notamment, on peut décider d'afficher, au choix :

- les publications mentionnant `@everyone`
- les publications mentionnant l'utilisateur connecté
- les publications des abonnements
- les publications de n'importe quel auteur choisi (soit on a cliqué sur son nom quelque part dans l'interface, soit on l'a recherché depuis le champs de recherche)
- les publications contenant un certain `#hashtag`.

(On peut envisager des façons de combiner des critères multiples.)
Une zone de l'écran permet de poster une nouvelle publication (seulement si on est connecté).
Une publication affichée consiste en :

- l'avatar et le login de l'auteur de la publication
- son horodatage
- le contenu de la publication (tel que posté dans le formulaire de publication)
- les réactions associées (*likes*, *dislikes*, ...)
- des liens pour éventuelles fonctionnalités supplémentaires (répondre en mentionnant l'auteur, « retweeter », s'abonner au flux de l'auteur...)

Des fonctionnalités basiques de texte enrichi peuvent être ajoutées en gérant le format *Markdown* dans le texte des publications (à gérer via une dépendance qui convertit le texte *Markdown* en HTML).

### Technologies utilisées

Le site, sous la forme d'une page unique, devra être implémenté en *HTML+CSS+javascript+VueJS* côté frontend [1] et *nodejs* avec *express* côté backend.
Le backend exposera au client une API AJAX adéquate lui permettant d'incrire ou authentifier un utilisateur, ainsi que de récupérer les publications à afficher (pensez à comment éviter d'avoir à re-télécharger l'ensemble des publications à chaque fois qu'on fait une requête).
Les dépendances, la compilation et l'exécution du projet seront gérées par *npm*.

## II) Rendu

Le travail doit être réalisé jusqu'à la date limite via un dépôt git pour chaque groupe. L'usage de ce gestionnaire de version permet d'éviter toute perte de donnée, et d'accéder si besoin à vos anciennes versions.

- Votre dépôt git doit être hébergé par le serveur GitLab de l'UFR d'informatique : `https://gaufre.informatique.univ-paris-diderot.fr`
- Votre dépôt doit être rendu privé dès sa création, avec accès uniquement aux membres du groupe et aux enseignants de ce cours. Tout code laissé en accès libre sur Gaufre ou ailleurs sera considéré comme une incitation à la fraude, et sanctionné.
- Il va de soi que votre travail doit être strictement personnel : aucune communication de code ou d'"idées" entre les groupes, aucune "aide" externe ou entre groupes. Nous vous rappelons que la fraude à un projet est aussi une fraude à un examen, passible de sanctions disciplinaires pouvant aller jusqu'à l'exclusion définitive de toute université.
- La seule façon de réutiliser du code existant est via les dépendances du projet déclarées dans `index.json`.
- Le déploiement du site doit pouvoir être fait (notamment par vos examinaters) de la façon suivante :
  1. s'assurer que le SGBD adéquat est installé (`mysql` ou `postgresql`);
  2. éventuellement exécuter un script d'initialisation de la base de données (fourni par vous);
  3. exécuter `npm install`, puis `npm start` dans le répertoire du projet backend pour lancer le backend;
  4. pour le frontend : s'il n'est intégré dans le backend, exécuter `npm install`, puis `npm start` dans le répertoire du projet frontend pour lancer le frontend.

1. Il est difficile de dire juste « côté client », vu qu'il est habituel de lancer un serveur séparé pour servir statiquement le script client qui utilise VueJS, ainsi que la page web qui le charge.