# JSCHAT

LanChat

## Contenu

## Description du projet

Le projet "LanChat" est un ensemble de 2 applications fonctionnant en symbiose.
- Un serveur principal HTTP qui supervise la communication entre pairs via une API Rest
- Une application Web qui permet de "chatter" ainsi que la gestion des utilisateurs, des privilèges et des statistiques

## Fonctionnalités attendues

Les fonctionnalités attendues sont:

**Utilisateur**
- Ajouter un utilisateur
- Modifier un utilisateur
- Supprimer un utilisateur
- Désactiver un utilisateur (sans le supprimer)
- Lister les utilisateur
- Afficher les détails d'un utilisateur

**Discussion**
- Un utilisateur s'identifie par son pseudo et son mot de passe
- Un utilisateur peut créer un salon de discussion
- Un utilisateur peut rejoindre un salon de discussion
- Un utilisateur peut créer une discussion privée avec un autre utilisateur
- Un utilisateur peut bloquer un autre utilisateur
- Un utilisateur peut signaler un autre utilisateur
- Un utilisateur peut envoyer un message dans les salons/discussions auxquels il a accès

**Administration, modération**
- Certains utilisateurs ont des pouvoirs spéciaux:
    - Créer des salons verrouillés par mot de passe
    - Gérer les utilisateurs (ajout, modification, suppression)
- Les salons de discussion sauvegardent l'historique des messages (100 maximum)
- Les discussions privées sont sauvegardées sans limite.

L'application propose 1 salon de discussion par défaut. Ce salon ne peut pas être supprimé.
A la connexion, les utilisateurs rejoignent automatiquement ce salon par défaut.

## Découpage de l'application

Le logiciel sera divisé en 2 applications (et donc 2 phases distinctes) :

1) Une API NodeJS Rest restituant les données au format JSON

2) Une application HTML/CS/JS permettant de manipuler ces données dans un navigateur

## Étapes du projet

La 1ère étape consiste à créer un dépôt git (sur github ou gitlab) et d'y inviter les membres de votre équipe.

**/!** 1 seul dépôt par équipe ! **/!**

Le dépôt doit respecter la structure suivante :

```
nom_du_projet
        /docs/
        /src/
        /src/api/
        /src/app/
        /tests/
        README.md
```

Le répertoire "**docs**" contient la documentation du projet au format Markdown, HTML ou PDF.
Le répertoire "**src**" contient 2 sous répertoires :

- **api** : contient le code source de l'api (phase 1)
- **app**: contient le code source de l'application frontend (phase 2)

Le répertoire "**tests**" contient les fichiers de tests de vos objets.

Au début, vous devrez réaliser le travail qui sera demandé via le système de tickets de github.
Cela concerne essentiellement la modélisation des modèles d'objets nécessaire à l'application.

Une fois la couche "Modèle" validée, elle sera rendue visible en tant qu'API Rest.

La 2nde phase consistera à développer l'application web HTML permettant d'interagir avec l'API.
Un document PDF détaillé vous sera fourni à ce moment là.

Vous l'avez compris, tout le code NodeJS se trouve sous le répertoire "/src/api/".
Le répertoire "/src/app/" sera utilisé dans la 2nde phase du projet.

## Tester mon code

Si vous développez un objet (une classe) ou une fonction, il est intéressant de pouvoir tester son travail.

Le répertoire "**tests**" est prévu pour cela.

Par exemple, je dois implémenter une classe "Person".

1) je code ma classe et en fait un module NodeJs ( `module.exports = Person;` )
2) Dans le répertoire "**tests**", je crée un fichier "Person_test_mickael.js"
3) J'importe ma classe ( `const Person = require('../src/api/Models/Person.js');` )
4) J'y place mon code de test (créer des Person, les modifier, les afficher dans la console…)
5) Je teste 😊

--- FIN DU DOCUMENT ---

http://www.arfp.asso.fr