Ce travail pratique compte pour 25 points de la note finale du cours. La date de remise du travail sera indiquée sur le Moodle du cours.

**Section I : (3 points) Création du projet dans gitHub et Atlas**

Commencez par créer un projet privé dans gitHub que vous devez nommer en suivant ces règles : « SERVICES\_FAM\_TP1 » où FAM sont les 3 premières lettres de votre nom de famille. Par exemple, pour moi mon projet se nomme SERVICES\_PIL\_TP1.

Ajoutez un fichier README.md pour décrire le projet. Inscrivez un titre et votre nom ensuite vous pouvez copier-coller l’énoncé du travail comme description. Vous devez ajouter un fichier dans votre git pour ignorer les fichiers se trouvant dans les dossiers : « node\_modules ». Ajoutez tous les fichiers nécessaires au bon fonctionnement de votre projet dans git et ensuite publiez le tout dans votre projet gitHub (en faisant un push, vous devriez tester que votre projet est bien publié en faisant un git clone dans un autre dossier sur votre ordinateur).

Dans Moodle vous allez me remettre le présent fichier qui va contenir ces deux informations :

1. Votre répertoire gitHub comme ceci :

git remote add origin https://github.com/OctazGitHub/ServiceTp1.git

1. Votre token d’accès privé comme ceci:

ghp\_QEzOusMv8iAjb0gu6b2ymWoeuQmQgu0jPpXU

**Il** sera important dans votre remise de remettre le présent fichier et un dossier compressé contenant tout votre projet (soit tous les fichiers nécessaires à l’exécution du projet). Votre fichier principal pour que je puisse démarrer votre projet doit se nommer « index.js ». N’incluez pas votre « node\_modules » dans ce dossier compressé.

Vous devez, ensuite, ajouter une base de données MongoDB dans Atlas que vous allez nommer services\_TP1 et une collection « messages ». La collection messages va contenir des documents Mongo représentant les messages d’un « canal de discussion » dans le format suivant :

**[ { \_id : identifiant du document (une chaîne),  
 titre : le titre du document (du message de discussion – une chaîne),  
 auteur : l’auteur du document (une chaîne),  
 description : la description du message (une chaîne),  
 date : la date que le document a été créé (format Date),  
 langue : la langue utilisé dans le message (une chaîne),  
 commentaires : un tableau (une liste) de plusieurs sous-documents dans le format suivant :  
 [ { commentaire : (une chaîne), auteur : (une chaîne), date (date du commentaire) },   
 { … }…  
 ]   
 }, {…} ]**

Voici un exemple utilisant ce format :

**[ { "\_id": "61770f4aabe8a0b17b5f1889",**

**"titre": "Comment faire fructifier votre argent sans risques",**

**"auteur": "joseph@trucmachin.com",**

**"description": "Dans ce sujet nous allons discuter des possibilités de faire fructifier votre argent sans prendre de risques",**

**"langue": "français",**

**"date": "2022-10-23T04:00:00.000Z",**

**"commentaires": [**

**{ "commentaire": "Tu dis n'importe quoi",**

**"auteur": "jtecroispas@ici.com",**

**"date": "2022-10-25T04:00:00.000Z"**

**}**

**]},**

**{ "\_id": "123",**

**"titre": "22222 Comment faire fructifier votre argent sans risques",**

**"auteur": "2222joseph@trucmachin.com",**

**"description": "222 Dans ce sujet nous allons blabla",**

**"date": "2022-10-25T05:00:00.000Z",**

**"langue": "français",**

**"commentaires": [**

**{ "commentaire": "Tu dis n'importe quoi",**

**"auteur": "jtecroispas@ici.com",**

**"date": "2022-10-25T04:00:00.000Z"**

**},**

**{ "commentaire": "autre chose",**

**"auteur": "jtecroispas@ici.com",**

**"date": "2022-10-25T06:00:00.000Z"**

**}]**

**}]**

**Section II : (12 points) Projet nodeJS de services Web**

Pour cette partie, vous devrez faire un programme à partir des fichiers et exemples que nous avons fait pendant le cours. Il s’agit d’écrire un programme qui démarre un serveur Web afin de réaliser un API de type REST pour accéder aux données des messages (décrit plus haut) se trouvant dans Atlas. Votre programme doit utiliser le fichier index.js pour se connecter à la base de données Atlas (voir section I ci-dessus) et activer les routes nécessaires afin de servir les URL suivants (ces routes doivent être définies dans un fichier .js se trouvant dans le répertoire « ./routes »):

* GET « /api/messages » pour récupérer tous les messages de la BD (limite à 250 messages)
* GET « /api/messages/id » pour récupérer le message dont l’identifiant est id
* GET « /api/messages/recherches/texte » pour récupérer tous les messages de la BD (limite à 250 messages) dont la description contient le texte mentionné (messages triés par date décroissante)
* POST « /api/messages » (avec le document dans le corps de la requête– body) pour ajouter le document dans les messages de la BD (en utilisant une valeur par défaut pour la date, vous n’aurez pas à fournir la date, votre API utilisera la date courante, Date.now)
* PUT « /api/messages/id » pour modifier un message de la BD (dont le id est mentionné dans l’URL alors que le nouveau contenu est dans le corps de la requête)
* DELETE « /api/messages/id » pour supprimer le message correspondant au id

De plus, votre programme doit définir un schéma de données à l’intérieur d’un fichier .js se trouvant dans le répertoire « ./modeles ».

Testez votre API dans un client REST (je vous suggère celui disponible dans VScode).

**Section III : (10 points) Ajout d’une page Web permettant d’utiliser l’API**

Pour cette partie, vous devrez faire une page Web permettant l’interaction entre l’utilisateur et votre API. Ce programme utilisera la méthode de votre choix pour faire appel à l’API et présenter un contenu dynamique à votre utilisateur. Ce programme doit utiliser JavaScript et JQuery afin de faire appels à toutes les fonctionnalités de votre API. Plus de détails suivront concernant cette partie du travail.