Ce travail pratique compte pour 15 points de la note finale du cours. La date de remise du travail sera indiquée sur le Moodle du cours. Un des critères d’évaluation de ce travail sera la rigueur et l’exactitude dans l’exécution des tâches demandées.

**Section I : (5 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\_DEV1 » où FAM sont les 3 premières lettres de votre nom de famille. Par exemple, pour moi mon projet se nomme SERVICES\_PIL\_DEV1.

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/alain-clg/SERVICES\_PIL\_DEV1.git

1. Votre token d’accès privé comme ceci qui sera valide pour 30 jours :

ghp\_EbUOdXCBf6tDYG0pknabc123911klBT14IhN3R

https://github.com/Redgy2/SERVICES\_PER\_DEV1.git

ghp\_gptbrNprHTtjzFlIzmaJkWPTPYWIOJ1fUIJ7

Il sera important dans votre remise de remettre le présent fichier et un dossier compressé en format ZIP (tout autre format comme 7ZIP ou RAR **ne sera pas** corrigé) 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. Votre fichier package.json doit être modifié afin de permettre de démarrer le projet dans nodemon en utilisant « npm run dev ».

Vous devez, ensuite, ajouter une base de données MongoDB dans Atlas que vous allez nommer services\_DEV1 et une collection « messages ». La collection messages va contenir des documents Mongo représentant les messages d’un « canal de discussion (chat) » 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"**

**}**

**]},**

**mess**

**Section II : (10 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/description/: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)
* GET « /api/messages/titre/:texte » pour récupérer tous les messages de la BD (limite à 250 messages) dont le titre contient le :texte mentionné (messages triés par date décroissante)
* GET « /api/messages/auteur/:texte » pour récupérer tous les messages de la BD (limite à 250 messages) dont l`auteur contient le :texte mentionné (messages triés par date décroissante)
* DELETE « /api/messages/:id » pour supprimer le message dont l’identifiant est :id
* POST « /api/messages/ » pour ajouter le message (le JSON du message contient le champ \_id)
* PUT « /api/messages/:id » pour modifier le message

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

Ajoutez l’extension REST CLIENT dans vsCode et créez un fichier de tests que vous allez nommer test.rest afin de pouvoir tester votre API. Assurez-vous que votre API fonctionne correctement en testant les différentes routes avec des différents paramètres.

N’oubliez pas d’ajouter tous ces fichiers dans votre projet et de remettre deux fichiers dans Moodle, le premier le présent fichier Word et le second un dossier compressé (.ZIP). N’oubliez pas aussi de publier tous ses fichiers dans github. Testez que vous arrivez à « cloner » votre projet github sur votre ordinateur.