TD/TME 7 : Définition et Implémentation des Web Services

Nous allons dans ce TD spécifier les Web Services implémentés dans le cadre de notre site Web. Ces Web Services seront programmés en suivant les spécifications REST, et en retournant des résultats JSON [1]. Le travail effectué en TD servira de base pour la programmation de l'API.

1 Notions de bases

Q 1 Qu'est-ce qu'une API REST ?

Q 2 Comment organiser les services selon le principe d'API REST ?

Q 3 Quelles sont les implications sur l'url ? le code ? Comment tester ?

Q 4 Comment faire des tests sur l'API REST ?

2 Spécification du projet : documentation et architecture

Ce travail de spécification doit être un **travail précis** et il sera important de conserver **une documentation des services implémentés** pour la suite du projet.
Pour chaque service, il faudra spécifier :

Nom du web service	
URL du web service	
Description du service	
Paramètres en entrée	
Format de sortie	
Exemple de sortie	
Erreurs possibles	
Avancement du Service	
Classes/Fichiers JavaScript	
Informations additionnelles	

Le projet sera organisé de la manière suivante :
- `server` qui contient le code lié au serveur
- `client` qui contiet le code lié au client
- `common` qui contient le code partagé (client/serveur)

Le projet (côté serveur) sera organisé de la manière suivante (répertoire) :
- `package.json` généré lors du "npm init"
- `jsconfig.json` Le fichier qui configure les chemins pour javascript
- `src/app.js` Contient le code qui définit le serveur
- `src/index.js` Contient le code qui permet de lancer le serveur (listen)
- `src/api.js` Contient le code qui définit l'API
- `src/routes.js` Contient le code qui définit les routes
- `src/entities/`, un répertoire qui contiendra les différentes entités à manipuler, par exemple `src/entities/users.js` pour la gestion des utilisateurs. Vous pourrez par exemple définir une classe `Users` qui permet de gérer les utilisateurs avec des méthodes permettant de créer et interroger la base de données (mais vous êtes libres pour l'organisation que vous choisirez).

3 Premier Service : Création d'utilisateurs

1. http://www.json.org/

On va considérer pour tous les services les codes de status HTTP (réponse). Un message d'erreur succinct et plus détaillé peut être donné dans la réponse.

Q 5 Spécifiez le service permettant de créer un nouvel utilisateur.

Q 6 Donnez le JSON retourné en cas de succès et d'erreur, ainsi que le status HTTP de la réponse

Q 7 Ecrire les étapes en pseudo-code pour réaliser le service `createUser`, en considérant les connections à la base de données.

4 Second Service : Login

Nous allons considérer le système d'authentification suivant :

- Lors du *login*, une clef sera générée automatiquement par le serveur, et transmise sous forme de cookie. Si vous utilisez le middleware `express-session`, cela est fait automatiquement
- La clef aura une durée de vie déterminée
- Certaines clefs (clefs administrateur) auront une durée de vie illimitée

Q 8 Spécifier le service *login*.

Q 9 Ecrire l'algo correspondant en pseudo-code très simple.

Q 10 Quelles sont les fonctions dont vous allez avoir besoin ?

Q 11 Ecrire le service correspondant en JavaScript.

5 Troisième Service : Logout

Q 12 Spécifier le service de logout. Ecrire le code du service.

6 Autres Services

Q 13 Faire la liste des services **de base** qui devront être implémentées dans votre projet. Les services seront regroupés par familles :

- authentification
- messages
- amis
- recherche
- commentaires

Pour chaque service, vous spécifierez les entrées et sorties. La spécification complète et documentée des services sera à faire chez vous.