

## Rapport Client/Server Side

**TOUNSI Molka**  
**ALAYA Hana**  
**BENATHMANE Ayoub**  
**MISSAOUI Ahmed**

### Le sujet et les objectifs :

Nous avons réalisé une plate-forme de support pour des cours en ligne. Cette plate-forme web fournira à N personnes travaillant ensemble les services suivants :

Pour la partie Front-end (Client Side) :

- Chat audio, vidéo et texte en N-N (4 personnes max par room)
- Carte permettant de situer la position géographique des participants
- Transfert de fichier direct en peer to peer
- FileSystem partagé et versionné

Pour la partie Back-end (Server Side) :

- L'utilisateur peut s'identifier avec son login et mot de passe stockés dans MongoDB
- Une fois connecté, l'utilisateur ne visualise que la liste des rooms où il a droit d'y accéder et les noms des autres membres, avec le nombre des membres connectés

Rooms				
_id	name	connectedUser	created_at	authorizedUsers
54d3466c205f705410004e1d	Master Web	1	Thu Feb 05 2015 11:31:08 GMT+0100 (Paris, Madrid)	BENATHMANE,TOUNSI

[create](#) [logout](#) [join](#)

- L'utilisateur peut créer une salle virtuelle et inviter des membres à cette dernière

### Vite! Créez Votre Room :)

Name

Master Web

Authorized Users

BENATHMANE

MISSAOUI

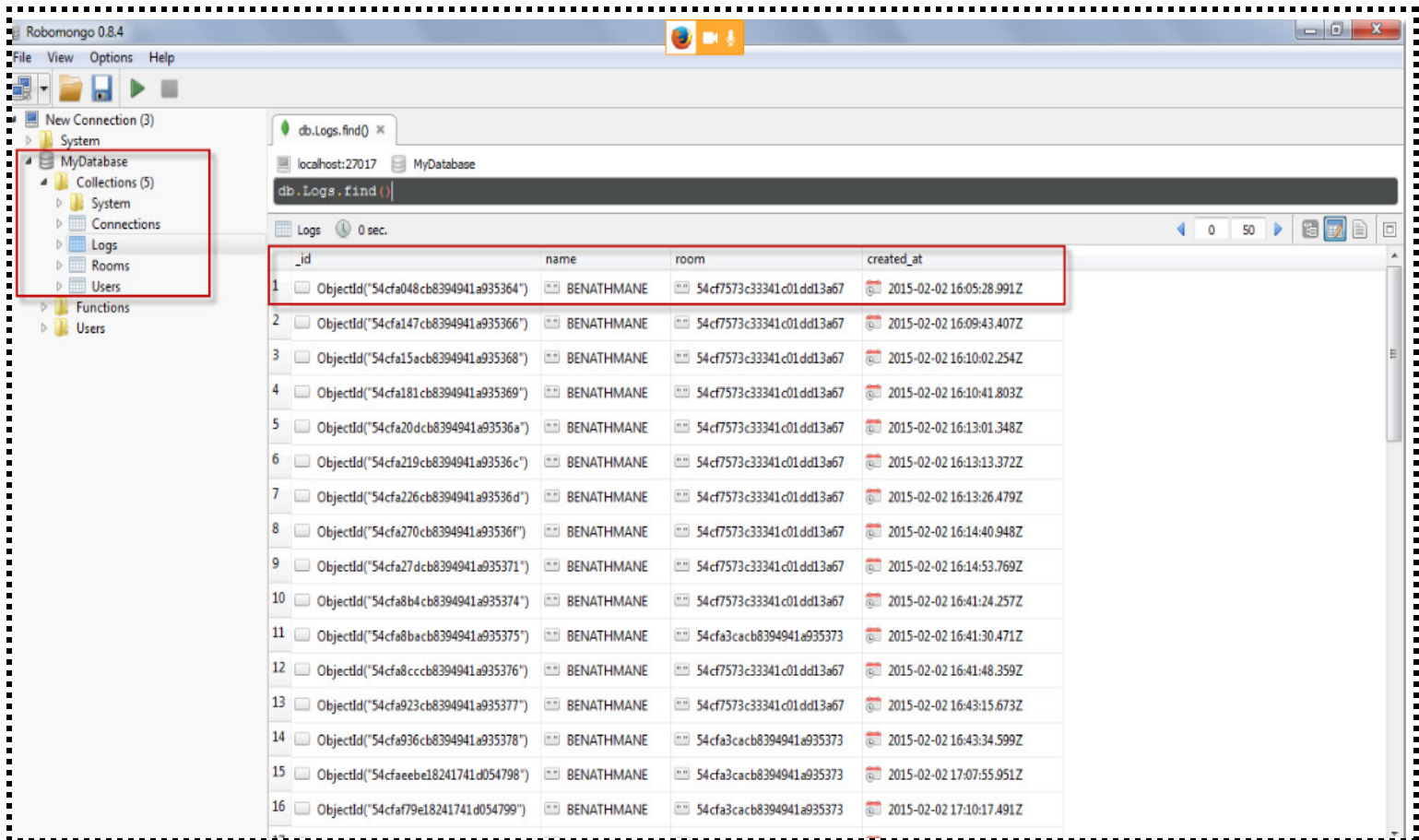
TOUNSI

ALAYA

Submit

Cancel

- Gestion des logs : Stocker l'historique des connexions utilisateurs (quelle room en quelle date et heure)



### Les contributions individuelles des membres du groupe :

- BENATHMANE Ayoub : **Chat + localisation + Amélioration des pages Html + JADE + Bootstrap (Front End).**
- MISSAOUI Ahmed : **Back-end (MongoDB + express + Room N\*N (Vidéos) + Transfert de Fichiers Peer, Amélioration des pages Html + JADE (Front End).**
- TOUNSI Molka : **Back-end (MongoDB + express) + Front End (Les pages Html + JADE).**
- ALAYA Hana : **Système de fichier partagé (express+formidble) + Amélioration des pages Html + JADE (Front End) + les tests unitaires (Back-end).**

### Les technologies utilisées :

- HTML, CSS, JavaScript.
- NodeJS.
- MongoDB.
- Les modules NodeJS : body-parser, client-sessions, easyrtc, express, formidable, jade, mongodb, node-static et socket.io
- EasyRTC : API Javascript qui simplifie le codage nécessaire pour l'utilisation des applications WebRTC car il propose pleins d'exemples avec une documentation bien expliqué.
- RoboMongo : Est un outil open source de gestion MongoDB shell-centric (c'est à dire l'interface d'administration)

Pour les technologies Html, CSS, et NodeJS, on les utilise actuellement dans le cadre du projet web de données pour le développement d'un guide touristique, sauf que la base de données est une triple Store permettant de stocker des données RDF et non pas MongoDB.

### Les points forts de notre application :

Notre application a été développée en utilisant Node.js qui est une technologie assurant le développement agile . Node.js assure un échange de données asynchrone, ce qui permet une meilleure réactivité des applications web. Utiliser Node.js nous a donc permis d'assurer une bonne réactivité , l'agilité de notre développement et un échange asynchrone.

Nous avons utilisé MongoDB qui est une base de données NoSQL (Not only SQL) orientée document . Ce type de base de données est conçu pour stocker les données sous format JSON . Elle est caractérisée par le manque de schéma prédéfini de sorte que les données ne doivent pas forcément avoir un schéma fixe . On parle de schemaless.

Nous avons aussi utilisé le WebRTC, une technologie récente qui permet de communiquer en temps réel et directement entre navigateurs web. Utiliser cette technologie a été très enrichissant et nous a permis d'évaluer les perspectives révolutionnaires qu'elle offre pour le web.

### Ce que nous ferions si nous avions encore deux semaines :

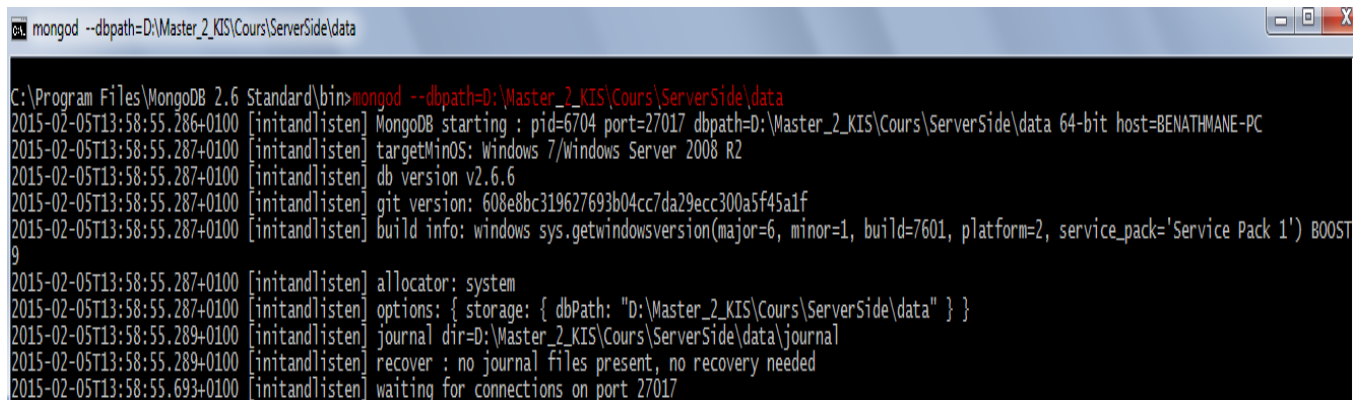
Le projet demandé nécessite l'utilisation de plusieurs nouvelles technologies dont nous n'avons pas les connaissances antérieures. Se familiariser avec ces frameworks représente un challenge que nous avons surmonté même si ça n'a pas été gagné d'avance. Le temps de réalisation a été un inconvénient mais aussi une motivation nous poussant à surmonter le défi.

Si nous avons eu plus de temps , nous aurions pu améliorer certains aspects de l'application comme le design , le chat , le système de fichiers partagés, l'envoi d'invitation à une room par email ...

## Comment tester l'application :

### 1. Lancer MongoDB :

MongoDB nécessite un dossier de données pour stocker ses fichiers. Vous pouvez créer ce dossier à n'importe quel endroit dans votre disque sous le nom **“data”**, et se rendre au dossier



```
cmd mongod --dbpath=D:\Master_2_KIS\Cours\ServerSide\data
C:\Program Files\MongoDB 2.6 Standard\bin>mongod --dbpath=D:\Master_2_KIS\Cours\ServerSide\data
2015-02-05T13:58:55.286+0100 [initandlisten] MongoDB starting : pid=6704 port=27017 dbpath=D:\Master_2_KIS\Cours\ServerSide\data 64-bit host=BENATHMANE-PC
2015-02-05T13:58:55.287+0100 [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2015-02-05T13:58:55.287+0100 [initandlisten] db version v2.6.6
2015-02-05T13:58:55.287+0100 [initandlisten] git version: 608e8bc319627693b04cc7da29ecc300a5f45a1f
2015-02-05T13:58:55.287+0100 [initandlisten] build info: windows sys.getwindowsversion(major=6, minor=1, build=7601, platform=2, service_pack='Service Pack 1') BOOST
2015-02-05T13:58:55.287+0100 [initandlisten] allocator: system
2015-02-05T13:58:55.287+0100 [initandlisten] options: { storage: { dbPath: "D:\Master_2_KIS\Cours\ServerSide\data" } }
2015-02-05T13:58:55.289+0100 [initandlisten] journal dir=D:\Master_2_KIS\Cours\ServerSide\data\journal
2015-02-05T13:58:55.289+0100 [initandlisten] recover : no journal files present, no recovery needed
2015-02-05T13:58:55.693+0100 [initandlisten] waiting for connections on port 27017
```

**“bin”** de votre installation MongoDB et lancer la commande :

*`mongod --dbpath=CHEMIN_VERS_VOTRE_DOSSIER_DATA\data`*

### 2. Lancer l'Application :

Se rendre dans le dossier de l'application, exécuter la commande **“npm install”** pour installer les dépendances puis **“node server.js** (nécessite un serveur NodeJS installé)



```
BENATHMANE-PC /D:\Master_2_KIS\Cours\ClientSide\PvP\ClientSide <master>
$ node server.js
info - socket.io started
info - EasyRTC: Starting EasyRTC Server (v1.0.12) on Node (v0.10.33)
info - EasyRTC: EasyRTC Server Ready For Connections (v1.0.12)
Fichier existant : Ali ABDELKAFI.pdf <0b7b65a1a8edeacb446f18791b9bd565>
Fichier existant : catalog-v001.xml <22ec748a8f4567e53a8db62f69d130e9>
Fichier existant : article_cloud.pdf <9d13c32f206ea81b18a8f145f5b5920f>
```

Lorsque la base est vide, des données de démarrage seront insérées dans MongoDB créant ainsi 4 utilisateurs de base et une Room appelée WEB:

IDENTIFIANT : AHMED - MOT DE PASSE : AHMED

IDENTIFIANT : AYOUB - MOT DE PASSE : AYOUB

IDENTIFIANT : MOLKA - MOT DE PASSE : MOLKA

IDENTIFIANT : HANA - MOT DE PASSE : HANA

### Les Unit Tests:

Pour réaliser les tests unitaires, nous avons utilisé “**MOCHA**” qui est un framework Javascript permettant de réaliser des tests unitaire.

Lancer le script shell par la commande “**sh run\_tests.sh**”.

Après le lancement et à la fin des tests, les résultats s’affichent dans la console comme le montre la figure suivante:

```
Node.js app launching on PID: 10283. Starting tests in 3 seconds. Log files for the Node.js app are logged to test.log
```

```
collectiontest
  ✓ this api call should create a new collection (38ms)
  ✓ this api call All Users (79ms)
  ✓ this api call user by name and mp
  ✓ this api call user by name test 2 name and mp
  ✓ this api call get collection by id (66ms)
  ✓ this api call get connected user (76ms)

6 passing (274ms)

info - socket.io started
info - EasyRTC: Starting EasyRTC Server (v1.0.12) on Node (v0.10.35)
info - EasyRTC: EasyRTC Server Ready For Connections (v1.0.12)
add test data
add test data
add test data
add test data
add test data
Users
{ name: 'ahmed', password: '345' }
POST /Users 201 8ms - 123b
Warning: missing space before text for line 36 of jade file "/Users/ALAYA/Desktop/dernier/ClientSide/views/data.jade"
GET /Users 200 315ms - 116.81kb
GET /user/AHMED/AHMED 212 3ms - 0b
GET /user/ahmed/345 212 4ms - 0b
entree Room
Warning: missing space before text for line 36 of jade file "/Users/ALAYA/Desktop/dernier/ClientSide/views/data.jade"
GET /Rooms/?nom=MOLKA 200 61ms - 113.2kb
Warning: missing space before text for line 36 of jade file "/Users/ALAYA/Desktop/dernier/ClientSide/views/data.jade"
GET /getConnectedUser 200 1ms - 9b
```

### Quelques imprimés écrans des résultats des tests:

Création d’une nouvelle collection par la méthode POST (/:collection)

(14) ObjectId("54d...	{ 4 fields }	Object
_id	ObjectId("54d5628ac46b...	ObjectId
name	ahmed	String
password	345	String
created_at	2015-02-07 00:55:38.57...	Date

### **EasyRTC retour d'expérience:**

Nous utilisons EasyRTC dont la documentation est disponible sur Github : (<https://github.com/priologic/easyrtc>).

EasyRTC est une librairie de haut niveau, qui nous permet d'intégrer des fonctionnalités sans passer du temps à développer ou à refaire des choses existantes, il suffit de comprendre les principes de chaque module et l'intégrer dans notre application.

Par exemple :

- `easyrtc.enableDataChannels(true);` permet d'activer le transfer de fichiers.
- `easyrtc.easyApp("easyrtc.webProject","selfVideo","callerVideo1","callerVideo2","callerVideo3"], loginSuccess,loginFailure);` ⇒ définir une application . Chaque application peut contenir plusieurs rooms.
- `easyrtc.joinRoom(roomName, null, loginSuccess,loginFailure);` permet à un utilisateur de se connecter à une Room

Pour que easyRTC fonctionne, il faut ajouter sa dépendance dans package.json, mais aussi utiliser son fichier javascript `easyrtc.js` pour la gestion des rooms et `easyrtc_ft` pour le transfert de fichier.

Pour pouvoir l'intégrer, il est nécessaire de définir des balises spécifiques avec des ids spécifiques. Il faut être attentif lors de l'exploitation.

Pour plus d'informations:

Documentation: <http://easyrtc.com/docs/>

API : <http://www.easyrtc.com/docs/browser/easyrtc.html>

### **Ressources supplémentaires utilisés:**

Nous nous sommes basés pour la réalisation de la partie du système de fichier partagé sur les travaux suivantes:

<http://openclassrooms.com/forum/sujet/nodejs-application-de-partage-de-fichier-chat>

<https://github.com/etienne-gauvin/partage>

<http://naholr.fr/2011/06/presentation-de-node-js-et-du-framework-express/>

<http://zeptajs.com/>

<https://github.com/felixge/node-formidable>

Pour les tests unitaires, nous nous sommes basés sur des exemples des tests d'API REST avec mocha et nodejs:

<https://github.com/mochajs/mocha>

[https://github.com/oottinger/express\\_api\\_testing\\_with\\_mocha](https://github.com/oottinger/express_api_testing_with_mocha)

<https://thewayofcode.wordpress.com/2013/04/21/how-to-build-and-test-rest-api-with-nodejs-express-mocha/>

<http://webapplog.com/express-js-4-node-js-and-mongodb-rest-api-tutorial/>