LEGER
Romain

LP CDED

Nouveaux modes de développement
Rapport n°1 du 16/06/2015

1. Présentation (30 minutes)

Présentation du sujet par le professeur, prise de notes sur le projet à réaliser, les installations à effectuer et les différents composants du projet.

2. Prise en main (30 minutes)

Tout d'abord, j'ai décidé de choisir le terminal fourni par l'IUT. J'ai eu un problème avec le terminal Linux, pour cause d'installation de *Node.js* impossible, j'ai donc travaillé en attendant avec Quentin Kornmann. Nous avons essayé de comprendre les mécanismes de JavaScript avec le programme tp0, mis à disposition sur le Moodle. Puis nous avons lancé le programme grâce à Node pour comparer les réponses.

3. Installation des logiciels (1 heure)

J'ai d'abord travaillé sur l'environnement Linux, grâce aux terminaux de l'IUT. Mais le problème des droits administrateurs se posaient à chaque fois pour les installations. J'ai donc décidé de prendre mon ordinateur personnel pour effectuer les installations, le tout sous environnement Windows 10. La première tâche à faire était d'installer *Node.js*, accessible sur le site *https://nodejs.org/en/*. Ce dernier fournit directement le module "npm" nécessaire à ce projet.

Après cette installation primaire, j'ai créé un compte sous GitHub, pour avoir une traçabilité du projet, grâce à ses différentes versions. Pour compléter cet environnement, j'ai installé SourceTree, accessible à l'adresse https://www.sourcetreeapp.com/, qui fournit une interface graphique pour la gestion des comptes et dépôts GitHub, sans passer par la ligne de commande Git. J'ai donc lié le compte GitHub avec ce logiciel.

Je n'ai pas installé d'IDE particulier, ayant déjà de base sur mon ordinateur personnel Brackets et NotePad++. Je n'ai pas choisi Eclipse, la combinaison entre Brackets et SourceTree étant suffisant.

4. Création du serveur HTTP (1 heure)

La prochaine étape était d'installer le serveur HTTP pour effectuer des requêtes JavaScript depuis l'ordinateur. J'ai utilisé le mode console, pour lancer les différentes options de npm, telles que *npm init, npm install ejs, express --save, http, https*. J'ai ensuite créé et édité le fichier *server.js* pour permettre une connexion avec le serveur. Après plusieurs tentatives, et différentes version du fichier *server.js*, la connexion a pu s'établir sur le port 3000, avec un simple renvoi d'une réponse texte.

Puis, j'ai amélioré le programme *server.js* pour permettre de charger une page html à partir de ce fichier. D'abord, j'ai créé de multiples fichiers, et grâce à un router, je pouvais charger un fichier HTML à partir du fichier JavaScript de base. J'ai suivi le tutoriel accessible à cette page : https://codeforgeek.com/2014/06/express-nodejs-tutorial/. Enfin, j'ai fait un commit sur SourceTree pour sauvegarder le tout.

5. Amélioration en serveur HTTPS (2 heures)

La prochaine tâche à effectuer était de créer un certificat pour rendre le site web sous contrôle HTTPS. J'ai d'abord cherché longuement comment créer ce certificat, et je suis arrivé sur le site

suivant : http://www.developpez.net/forums/d1019281/environnements-developpement/windev/contribuez/creer-propre-certificat-numerique-personnel-auto-signe/, qui m'a permis de faire ce certificat. Après plusieurs problèmes, notamment le rajout de l'option *-config openssl/cfg*, dans les requêtes, le certificat est apparu. J'ai effectué l'étape 1, 2 et 3, la première montrant l'installation de OpenSLL sous Windows, puis la deuxième qui créé le certificat. Je me suis rendu compte à la fin que l'étape 3 était inutile. J'ai ensuite lancé une commande pour supprimer le mot de passe, et j'avais donc en sortie 2 fichiers : *.key* et *.crt*.

J'ai ensuite modifié le fichier *server.js* pour rendre effectif le serveur https. Après une erreur de certificat, le serveur HTTPS était maintenant disponible. J'ai passé la fin de cette partie à aider les personnes sur les certificats

6. Refactoring et début de Paper.js (1 heure)

J'ai profité de la dernière heure pour modifier le fichier *server.js*, qui utilisait beaucoup de fichiers inutiles, puis supprimer tous les fichiers non nécessaires présents sur le serveur. J'ai de plus commencé à comprendre l'environnement *Paper.js* après téléchargement de ce fichier. Je l'ai utilisé trivialement sur le fichier HTML disponible sur le site.

7. Prochaine étape

La prochaine tâche est de continuer à travailler avec l'environnement donné par *Paper.js*, pour résoudre les objectifs donnés pour le 30 septembre. Je pense que je devrais arriver à atteindre ces objectifs au bout de 4 ou 5 heures de travail, et je me donnerais le reste du temps pour essayer de comprendre le mécanisme du contrôle de l'animation par WebSocket, et si j'ai le temps, de faire les premiers tests de cette partie.