beerIUT

Projet final dans le cadre des matières "Technologies pour la production de logiciels"(m4105C) et "Programmation web client riche"(m4103C). Ce projet a pour objectif de rassembler les connaissances acquises dans le cadre de ces matières en Javascript côté serveur avec NodeJS pour la première et avec le framework Vue côté client pour la seconde.

Node

- **Node installation sur Windows**

  Il suffit d'aller sur le site officiel Node.js (https://nodejs.org/) et de télécharger l'installateur. Assurez-vous également que "git" est disponible dans votre PATH, "npm" pourrait en avoir besoin (vous pouvez trouver git ici).

- **Node installation sur Ubuntu**

  Vous pouvez installer facilement les nodejs et npm avec apt install, il suffit d'exécuter les commandes suivantes.

  ```
  $ sudo apt install nodejs
  $ sudo apt install npm
  ```

- **Autres OS**

  Vous pouvez trouver plus d'informations sur l'installation sur le site officiel Node.js (https://nodejs.org/) et le site officiel NPM (https://npmjs.org/).

Si l'installation a réussi, vous devriez être en mesure d'exécuter la commande suivante.

```
$ node --version
v8.11.3

$ npm --version
6.1.0
```

Si vous avez besoin de mettre à jour `npm`, vous pouvez le faire en utilisant `npm` ! C'est cool, non ? Après avoir exécuté la commande suivante, il suffit d'ouvrir à nouveau la ligne de commande et d'être content.

```
$ npm install npm@latest -g
```

Installation

- Projet

```
$ unzip projet_JS_Deldicque_Martel.zip
$ cd beerIUT
$ npm install
```

- Extension navigateur internet

Afin de pouvoir requêter l'API de Google en étant en localhost (CORS issue) nous avons trouver le moyen de faire fonctionner le requêtage en passant par l'installation d'une extension sur le navigateur.

1. **Pour Google Chrome :**

   "Allow CORS" ==> extension présente dans le Chome Web Store disponible ici

2. **Pour Firefox :**

   "CORS Everywhere" ==> extension présente sur addons.mozilla.org disponible là

Configurer l'application

Si vous désirez changer le port par défaut (i.e "3000")

Ouvrez `beerIUT/server.js` et éditez la ligne suivante en remplaçant la dernière valeur par le port désiré.

```
const port = process.argv[2] || 3000;
```

Lancer l'application

```
$ node server.js
```

Puis connectez-vous sur `http://localhost:3000/` sous réserve que vous ayez laisser le port par défaut, sinon référez vous au paragraghe précédent.

Présentation des choix technologiques et des fonctionnalité implémentées

Nous avons fait le choix d'implémenter Vue.JS, pour répondre aux attentes du cours de client riche, où l'utilisation d'un framework était demandé.

Cette implémentation nous a demandé de "refacto" notre projet pour passer d'un routage des pages web côté serveur au côté client. Ce qui permet à notre serveur de servir au client uniquement une seule page "index.html" où Vue intègre un routeur qui sert la page d'accueil ou la page d'information au besoin.

Pour faciliter, accélérer le développement avec Vue nous avons intégré au projet, Vuetify.

Nous utilisons différentes API :

1. Leaflet, permet de récupérer une carte.
2. OpenWeather, permet de récupérer la météo sur une ville.
3. Google AutoComplete, permet de récupérer une adresse à partir d'une recherche.
4. Bière, permet de récupérer des bières et brasseries, api fait maison.

Notre projet grâce à ces apis, nous permet de rechercher une ville aidé par une autocomplétion.
Cette recherche nous emmène sur une page où l'on peut voir la météo et l'heure actuelle du lieu recherché. L'heure est calculée grâce à la timezone renvoyée par OpenWeather. En plus de ces fonctionnalités, les brasseries présentes dans un rayon de 20Km de la ville recherchée sont affichées sur la carte et dans la liste.
Un survol de la souris sur les brasseries permet d'afficher une pop-up contenant le nom de la brasserie.
Pour chaque brasserie il est possible d'accéder au site internet et au téléphone.
Il est possible de sélectionner une brasserie, cela permet d'apercevoir toutes les bières disponibles et de rejoindre le chat.

Le projet intègre aussi un petit système de session très simple en faisant abstraction des contraintes de sécurité et d'optimisation, c'est à dire que le serveur fourni une session au client grâce à un identifiant mais sans authentification et ne gère pas les fin de session.
Ce petit système peut se voir côté client avec les couleurs de chat pour les utilisateurs, une couleur est propre à une session.

Présentation de la structure côté serveur Serveur m4105c

1. Requêter l'API (celles nous étant utiles sur le projet - descriptif complet dans le fichier `test.sh`)

- **API beerRoutes**

Methode	URL	Description
GET	/api/beerRoutes/	Récupérer toutes les bières
	/api/beerRoutes/:id	Récupérer une bière par identifiant unique
	/api/beerRoutes/deg/:deg	Récupérer les bières par degré d'alcool égal ou supérieur

- **API breweryRoutes**

Methode	URL	Description
GET	/api/breweryRoutes/	Récupérer toutes les brasseries

Methode	URL	Description
	/api/breweryRoutes/:id	Récupérer une brasserie par identifiant unique
	/api/beerRoutes/near?lat={lat}&long={long}&radius={radius}	Récupérer les brasseries présentes dans un rayon défini avec {lat} ==> la latitude, {long} ==> la longitude et {rad} ==> le rayon désiré

- **/api/breweryRoutes/:id**

error answer code : {"errorCode":21,"message":"Entity not found"}

success answer code : {"id":100,"breweries":"Bell's Brewery Inc.","address1":"8938 Krum Ave.","address2":"","city":"Galesburg","state":"Michigan","code":"49053","country":"United States","phone":"269.382.2338","website":"...","filepath":"","descript":"In ...","last_mod":"2010-07-22T22:00:20+02:00","coordinates":"42.2843,-85.4538"}

- **/api/breweryRoutes/near**

error answer code (position définie sur Nantes) : [] //Ce retour est volontaire afin de pouvoir gérer nos traitements internes

success answer code (rayon de 5000m sur Londres) : [{"id":563,"breweries":"Fuller, Smith & Turner PBC","address1":"Chiswick Lane South","address2":"","city":"London","state":"London","code":"","country":"United Kingdom","phone":"44-(0208)-996-2000","website":"http://www.fullers.co.uk/","filepath":"","descript":"","last_mod":"2010-07-22T22:00:20+02:00","coordinates":"51.4877,-0.2498"},{"id":1385,"breweries":"Youngs & Company Brewery","address1":"26 Osiers Road","address2":"Wandsworth","city":"London","state":"","code":"SW18 1NH","country":"England","phone":"020 8875 7000","website":"http://www.youngs.co.uk","filepath":"","descript":"","last_mod":"2010-07-22T22:00:20+02:00","coordinates":"51.4611,-0.1966"}]

- **API webRoutes**

Methode	URL	Description
GET	/api/webRoutes/	Fournit le fichier index.html
	/api/webRoutes/login?search={adresse}	Renvoie les données de connexion cliente après avoir précisé une ville à la place de la variable {adresse}

- **Test des routes**

Comme demandé, un fichier `test.sh` est à disposition afin de pouvoir tester les routes sur l'API.

###2. Partie Sockets Le client établit la connexion avec le serveur au chargement de l'index.html. Lorsque le client sélectionne une brasserie il déclenche l'évènement: "joinChat", intercepté par le serveur

qui le fait rejoindre une room.
Lorsque le client envoie un message il déclenche l'évènement: "sendMessage", intercepté par le serveur qui récupère grâce au cookie la session de l'utilisateur et ainsi la room et envoie à tous les utilisateurs de cette room le message via l'évènement "sendMessage".
Lorsque le client intercepte l'évènement "sendMessage", il affiche le message.

Présentation de la structure côté client (M4103C - Programmation web - client riche)

Pour la structure côté client il s'agit d'un modèle MVVC à travers Vue.js et le Two-way-binding. Le projet à un fichier HTML, index.html. Ce fichier intègre Vue et son composant routeur, qui permet d'afficher deux vues différentes, représentées comme composant Home.js et Main.js. Ces deux composants sont chargés d'afficher leurs pages respectives et ont un modèle et une vue qui leur est propre.

Dans static, on retrouve tous les fichiers qui sont disponibles pour le client, dans components les composants, dans js, les scripts clients.