Programmation Web
TP n° 7 : Application *full stack* avec VueJS et Express

Le but de ce TP est d'apprendre à développer une application dite *full stack* (c'est à dire, dans un même projet, programmer la partie exécutée par le client et la partie exécutée par le serveur.) et à configurer un tel projet via `npm`.

Dans ce TP, à titre d'exemple de programmation de vue réactive, vous programmerez un jeu de tic-tac-toe en ligne.

I) Avec un seul projet `npm`

La façon la plus simple de développer une application VueJS + Express consiste à créer un package `npm` pour node avec express comme vous l'avez déjà fait précédemment et de traiter VueJS comme une bibliothèque js classique (de la même façon que jQuery) : les pages html servies par votre serveur contiennent des balises `<script>` référençant

- d'une part la bibliothèque, comme par exemple :

  ```
  <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
  ```

- d'autre part vos scripts personnels, comme par exemple :

  ```
  <script src="app.js"></script>
  ```

  où `app.js` est un fichier servi par une route statique d'Express, stocké, par exemple, dans le répertoire `public/` de votre projet.

Exercice 1 :

Créez un projet node+express et définissez une route statique servant le répertoire `public/`. Placez dans ce répertoire la page en VueJS développée dans le TP6, de telle sorte qu'en se connectant à l'adresse du serveur dans un navigateur web, la page que vous avez écrite s'affiche bien de la même façon que dans le TP précédent (alors que vous l'affichiez en local).

II) Avec des projets *front end* et *back end séparés*

Souvent on préfère clairement séparer les parties *front end* (exécutée par le client) et *back end* (exécutée par le serveur) d'une application.
Plusieurs raisons à cela :

- C'est un principe de base de programmation : subdiviser les responsabilités d'un logiciel entre des modules avec un périmètre restreint (tout en essayant de minimiser les inter-dépendances).
- Le « back » et le « front » peuvent nécessiter des traitements différents en amont (avant de lancer l'application) : notamment de la compilation (si langage différent de javascript), du packaging, de la vérification statique ("lint"), des tests unitaires, ...
  Si le « front » est just un fichier `.js` perdu dans le répertoire des fichiers statiques du « back », aucun traitement automatique en amont n'est possible pour le « front ».
- Le « front », s'il est séparé, peut être servi, statiquement, par un serveur séparé de celui qui exécute nodejs+express.
  Cela peut avoir des avantages, en production, en termes de répartition de charge.

Dans ce cas, le serveur du « back » n'a plus besoin de routes statiques, ce qui le rend un peu plus simple (d'autant qu'il n'est pas forcément évident de savoir quels fichiers servir statiquement quand ceux-ci ont été générés automatiquement par des outils indépendants de ceux utilisés pour le « back ».)

Il se trouve que le projet « front » peut être géré par `npm`[1], de la même façon que le « back ».
Dans la section suivante nous allons voir comment configurer un projet « front » VueJS avec npm et Vue CLI.

III) Projet « front » naïf

Avant de passer à la suite, rappelons tout de même qu'il est facile de créer un serveur séparé pour le « front » : créez un projet NodeJS séparé via `npm` à l'aide de `npm start` puis à l'aide du module `http` (voire avec Express), définissez une route statique vers le répertoire contenant les fichiers du « front » (html+css+js pour le navigateur web).
Veillez à ce que ce serveur s'exécute sur un port différent du serveur du « back » (en production, le serveur « front » s'exécutera sur le port HTTP standard, à savoir le port 80).

IV) Projet « front » VueJS avec npm et Vue CLI

L'outil Vue CLI est un outil en ligne de commande servant à initialiser des projets utilisant le framework VueJS, gérés par `npm`. Cette initialisation dépend des réponses faites à quelques questions posées au développeur (il est aussi possible de partir de modèles types).
Dans cette section, nous allons apprendre à initialiser notre projet VueJS grâce à cet outil.

Exercice 2 : Installer Vue CLI

Il faut commencer par installer Vue CLI. Cela peut être fait depuis `npm`. Pour une installation globale :

```
npm install -g @vue/cli
```

Malheureusement, cela présente deux problèmes :

- Par défaut, `npm install -g` ne fonctionne que si votre utilisateur a les permissions pour installer des bibliothèques globalement[2]. Typiquement, sous Linux, il faut pouvoir écrire dans `/usr/local/lib/node_modules` et, pour cela, il faut être `root`.
  Pour changer ce comportement par défaut, vous pouvez configurer le fichier `$HOME/.npmrc`[3], par exemple de sorte à ce que les installations globales se fassent dans un répertoire `.npm-packages` sous votre répertoire utilisateur (donc sans droits d'administrateur).
- Par ailleurs, les outils installés par `npm` ne sont pas installés par défaut dans un répertoire listé dans la variable d'environnement `PATH`, ce qui fait qu'ils ne peuvent pas exécutés de n'importe où sans en indiquer le chemin.

Nous pouvons régler les deux problèmes en suivant les instructions ci-dessous :

1. créer le répertoire : `mkdir ~/.npm-packages`
2. configurer npm : ajouter la ligne suivante dans votre fichier `~/.npmrc` (créez le au besoin) :

   ```
   prefix = ${HOME}/.npm-packages
   ```

1. Ou outil équivalent, tel que `yarn`.
2. https://docs.npmjs.com/cli/v7/configuring-npm/folders
3. https://docs.npmjs.com/cli/v7/configuring-npm/npmrc

3. configurer le path : ajouter la ligne suivante dans votre fichier `.bashrc` (créez le au besoin) :

   ```
   export PATH=$HOME/.npm-packages/bin:$PATH
   ```

4. Vérifiez que ça marche : reconnectez-vous pour que le nouveau `PATH` soit pris en compte (ou bien exécutez `export PATH=$HOME/.npm-packages/bin:$PATH`), puis installez Vue CLI et exécutez-le :

   ```
   npm install -g @vue/cli
   vue
   ```

   Vous devriez alors voir s'afficher l'aide de Vue CLI.

Remarquez que vous pouvez désormais installer d'autres outils via 'npm install -g', notamment :
- `npm` lui-même ! Cela permet de le mettre à jour globalement.
- pareil pour Node JS (package `node`) ![4]
- vous pouvez aussi installer `yarn`, une alternative plus moderne et plus rapide à `npm` (et compatible !).

Exercice 3 : Création du projet

Maintenant on peut initialiser le projet.

1. Vous pourrez explorer à loisir les options de Vue CLI, mais pour aller à l'essentiel, la commande à exécuter est :

   ```
   vue create mon_app
   ```

   (remplacez `mon_app` par le nom de votre projet « front »)
   Répondez aux questions : vous pouvez tout laisser par défaut, en choisissant la version de Vue qui vous convient (entre 2 et 3).
   Alternative : la commande `vue ui` permet de lancer une interface graphique proposant les mêmes fonctionnalités qu'en ligne de commande.
2. Explorez les fichiers créés par Vue CLI. Regardez notamment :
   - le fichier `package.json` : regardez les dépendances ajoutées, ainsi que les différentes cibles `npm` ajoutées dans la section `scripts` (par exemple, la cible `"serve" veut dire que vous pouv` serve, pour lancer le serveur « front »).
   - le fichier `main.js` où vous ajouterez votre code
   - les fichiers `.vue` qui sont des « *composants monofichiers* » (*single file components* ou SFC) : des fichiers contenant à la fois un template html, du code javascript et du CSS pour décrire l'entièreté d'un composant.
     Vous y trouverez les différents éléments que vous avez définis dans les composants du TP6, juste disposés autrement.
3. Lancez `npm run serve` et connectez-vous à http://localhost:8080/ pour vérifier que ça marche.

Exercice 4 :

Créez un projet comme dans l'exercice précédent, et adaptez vos composants du TP6 sous la forme de SFC (fichiers `.vue`).
Vérifiez que le site obtenu a le même comportement qu'au TP6.

4. Il peut aussi être une idée d'installer Node JS localement dans un projet pour imposer une version précise de Node JS dans le projet : `npm install node`, sans le `-g`, dans le répertoire du projet.

V) Tic-tac-toe (ou « Morpion »)

À faire si vous avez fini le reste avant qu'on en vous donne le sujet du projet (sinon passez directement au projet !).

Exercice 5 :

Au choix : initialisez un projet « back » (NodeJS + Express) et un projet « front » (VueJS), ou bien créez un unique projet NodeJS + Express avec un « front » programmé dans un `.js` servi statiquement.

Dans le « back », programmez les routes suivantes :

- requête `GET` sur `jeu/` : retourne l'état du plateau de tic-tac-toe sous forme d'un objet JSON :
  ```
  {
    "tour": "X",
    "plateau": [["O", "", ""], ["", "X", ""], ["", "", ""]]
  }
  ```
- requête `POST` sur `start/` : réinitialise le jeu (peu importe le contenu de la requête)
- requête `POST` sur `jouer/` : joue le coup décrit par l'objet JSON posté, de la forme :
  ```
  {
    "joueur": "O",
    "ligne": "2",
    "colonne": "1"
  }
  ```
  Renvoie le nouveau plateau si le coup est accepté (selon les règles du jeu de tic-tac-toe[5]), `null` sinon.

Dans le « front », faites en sorte d'afficher un tableau de 3 par 3 boutons, dont les étiquettes contiennent le dernier état connu du plateau et tels que cliquer sur le bouton de coordonnées (i, j) envoie un coupe en (i, j) pour le joueur associé à ce client, en utilisant l'API décrite plus haut.

5. Est-il encore utile de les rappeler ? Sinon, c'est ici : https://fr.wikipedia.org/wiki/Tic-tac-toe.