| Construire une API de démonstration en Node.js
| TD – (Nouvelle version)

Éléments de base

→ Voir le TD précédent

Création de l'API

Pré-requis : Archive `songs.zip`.

T01 | Dans votre répertoire utilisateur de vos projets NodeJS, décompressez l'archive.

Notes importantes :

- ✔ Cette archive comporte les réalisations du TD précédent jusqu'à la tâche **T10** incluse.
- ✔ Le projet suit de façon plus strict les recommandations d'une architecture RESTful :

https://blog.nicolashachet.com/developpement-php/larchitecture-rest-expliquee-en-5-regles/

- ✔ Le fichier `songs.json` se place à la racine, le dossier `raw` ne contenant que les fichiers mp3.
- ✔ Ce projet intègre en plus un utilitaire : `nodemon` (explication donné en cours)

Création des méthodes usuelles

- ***GET***

  → Voir le TD précédent

- ***POST***

Pour cette méthode, nous passerons par le logiciel **Insomnia** afin de pouvoir envoyer des POST en simulant l'envoi d'un formulaire.

T11 | Ajoutez dans le fichier `index.js`, (**AVANT le listen**) la méthode :

```
app.post('/songs',(req,res)=>{
   const { titre } = req.body;
   const { auteur } = req.body;
   const { genre } = req.body;

   res.send({
      donnees:`${titre} ${auteur} ${genre}`
   });
});
```

Dans cette méthode, il se pose un problème : nous allons envoyer des données sous forme JSon. Il faut donc indiquer (et donner) la possibilité au middleware **Express** de traiter du JSon dans le corps d'une requête.

T12 | Ajoutez après les constantes du début de fichier ceci :

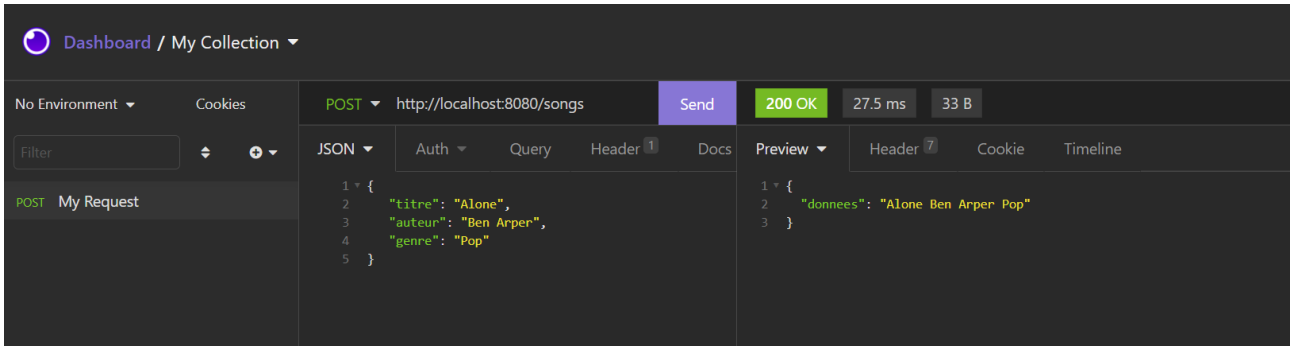
```
app.use(express.json());
```

La chanson supplémentaire :

```
{
```

```
    "titre": "Alone",
    "auteur": "Ben Arper",
    "genre": "Pop"
}
```

Le reste se passe avec Insomnia :



On voit bien le passage de la chanson supplémentaire. Il suffira dans la méthode POST de l'ajouter au fichier…

La méthode construite ne respecte pas la norme REST, il faut donc la réécrire pour qu'elle corresponde à la méthode POST chargée d'ajouter une chanson).

T13 | Changez la méthode POST par celle-ci, (**AVANT le listen**) :

```
// Création d'une chanson
app.post('/songs/create',(req,res)=>{
   const { titre } = req.body;
   const { auteur } = req.body;
   const { genre } = req.body;

   fs.readFile(path.join(__dirname, './songs.json'), 'utf8', function (err, songs) {
          if (err) { throw err; }
          let liste = JSON.parse(songs);

          liste.push({
             titre: titre,
             auteur: auteur,
                 genre: genre
         });

         fs.writeFile('./songs.json', JSON.stringify(liste), (err) => {
                 if (err) {
                 console.log(`Error writing file: ${err}`);
                 }
         });

          res.send('Fichier actualisé');
   });
});
```

Dans ce TD, on ne s'occupera pas du transfert du fichier mp3. Il faut donc le faire ici manuellement…

- ***BONUS***

Puisqu'on parle de musique, écoutons-la ! Il nous faut donc un module pour le streaming audio.

T14 | Installez le module MediaPlayer pour le projet :

```
$api> npm install -save mediaserver@latest
```

T15 | Intégrez ce module au début du fichier `index.js` :

```
const ms = require('mediaserver');
```

T16 | Insérez la nouvelle méthode GET pour la lecture du morceau (à la suite des autres méthodes GET) :

```
// Joue le morceau
app.get('/songs/play/:id', (req, res) => {
   let { id } = req.params;

   fs.readFile(path.join(__dirname, './songs.json'), 'utf8', function (err, songs) {
          if (err) { throw err; }

          let liste = JSON.parse(songs);
          let long = liste.length;
          id = id - 1;

          if ((id >= 0) && (id < long)) {
                let morceau = path.join(__dirname,'./raw',liste[id].titre);
                ms.pipe(req,res, morceau+'.mp3');
          } else {
                res.status(418).send({ message: 'ID non valide !' });
          }
   });
});
```

Des explications supplémentaires sont données en cours pour le fonctionnement de cette dernière méthode…