

# TD REST/VUE.JS

## SR10/AI16

### Récapitulatif

Dans notre projet, on développe une application qui respecte l'architecture MVC dont toute l'application fonctionne sur le serveur web (node) : lecture/écriture des données, traitement des requêtes, production du rendu (vues). Dans ce TD, nous allons expérimenter un autre type d'architecture appelée « single page architecture » dont l'application sera séparée en deux parties :

- Backend : développée avec le framework Express dont ses responsabilités : écrire/lire les données, traiter les requêtes via des apis REST (sauvegarder, modifier, retourner le modèle)
- Frontend : développée avec le framework Vue.js dont sa responsabilité est de fabriquer le rendu (vues)

### Partie 1 - Exposer les données via une API REST

Dans cette partie, nous allons créer une route qui retourne la liste des entités en json. Pour le faire il faut :

- Déclarer la route dans app.js :

```
var apiRouter = require('./routes/api');  
  
.....  
app.use('/api', apiRouter);
```

- Ajouter un fichier api.js dans votre dossier routes qui implémente le traitement des requêtes REST.

Nous utilisons la méthode « send » ou « json » du paramètre res (response) de callback de app.get

```
router.get('/users', function (req, res, next) {  
  
  result=userModel.readall(function(result){  
  
    res.status(200).json(result);  
  
  });  
});
```

***Vous pouvez visualiser le résultat en json dans le navigateur via l'url :***

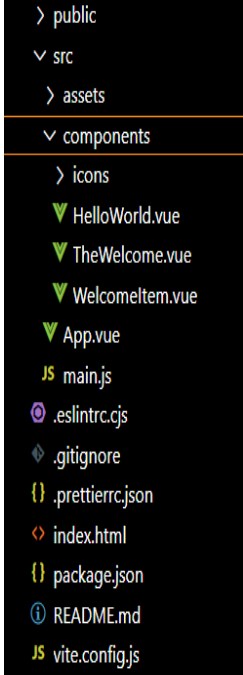
***http://localhost:3000/api/users***

Question :

- Créer l'API qui liste les offres avec leurs détails

## Partie 2 - Développer une petite application vueJS

- Créer un projet Vuejs : >npm init vue@latest
- Analyser la structure de votre projet Vuejs :

	Index.html	<p>Le point d'entrée en HTML en fournissant un élément (&lt;div id= "app"&gt;) dans lequel Vue.js se charge et importe le fichier main.js pour initialiser votre application.</p> <pre>&lt;!DOCTYPE html&gt; &lt;html lang="en"&gt;   &lt;head&gt;     &lt;meta charset="UTF-8"&gt;     &lt;title&gt;SR10&lt;/title&gt;   &lt;/head&gt;   &lt;body&gt;     &lt;div id="app"&gt;&lt;/div&gt;     &lt;script type="module" src="/src/main.js"&gt;&lt;/script&gt;   &lt;/body&gt; &lt;/html&gt;</pre>
	App.vue	La racine de votre application définie au format d'un composant Vue.js. C'est généralement quelque chose qui définit le modèle de votre page :
	main.js	Fichier JavaScript qui initialisera ce composant racine dans un élément de votre page. Il est également responsable de la configuration des plugins et des composants tiers que vous souhaitez peut-être utiliser dans votre application.
	Dossier components	Contient l'ensemble des composants Vue.js de votre application.

- Lancer votre application :
  - > cd <your-project-name>
  - > npm install
  - > npm run dev
- Modifier l'application pour afficher la liste des offres. Utiliser l'API javascript fetch pour interroger l'API REST que vous avez développé dans la première partie.
- Exemple pour lister des utilisateurs :
  - On ajoute un composant (dans le dossier components) qui fait appel à l'API REST et affiche l'objet json récupéré dans une table html.

```
<script setup>

import { ref } from 'vue';
```

```

const listItems = ref([]);

async function getData() {
  const res = await fetch("http://localhost:3000/api/users");
  const finalRes = await res.json();
  listItems.value = finalRes;
}

getData()

</script>

<template>
  <div>
    <h1> Liste des utilisateurs </h1>
    <table border="1">
      <thead>
        <tr>
          <th scope="col">Nom</th>
          <th scope="col">Prénom</th>
          <th scope="col">Email</th>
        </tr>
      </thead>
      <tbody>
        <tr v-for="item in listItems">
          <td>{{ item.nom }}</td>
          <td>{{ item.prenom }}</td>
          <td>{{ item.email }}</td>
        </tr>
      </tbody>
    </table>
  </div>
</template>

```

- Insérer le composant « UsersList » dans la page :

```

<script setup>
import UsersList from './components/UsersList.vue';
</script>

<template>
  <header>
    <!-- ===== personnalisez votre header ici =====
-->

  </header>

```

```

    <main>
      <UsersList />
    </main>
  </template>

  <style scoped>
    /***** ajouter votre style css ici *****/
  </style>

```

- Si vous lancez votre application vous allez avoir une page vide et une erreur sur la console. Cette erreur est de type CORS :  
<https://developer.mozilla.org/fr/docs/Web/HTTP/CORS/Errors>  
 Pour régler ce problème il faut aller dans votre application node express et ajouter les instructions suivantes dans app.js :

```

var cors=require('cors');
...
app.use(cors());

```

- Résultat :

## Liste des utilisateurs

Nom	Prénom	Email
Lounis	Ahmed	ahmed.lounis@utc.fr
test	test	test@test.fr