| Node.js

| Projet QRIdent

1 – But de l'application

Cette application **NodeJS** comportera les techniques sur :

1. L'utilisation du framework CSS **Bootstrap**
2. L'utilisation d'une sauvegarde **Locale**
3. L'utilisation du SGBD **MongoDB** avec le connecteur **Mongoose**

Elle porte sur la création et la gestion par une application back-end d'un QR Code en fonction d'informations entrées dans un formulaire.

En ajoutant d'autres **f**onctionnalités, elle peut servir de base pour une mission relative à la gestion des entrées de personnes dans des locaux dits « sensibles » comme des espaces soumis à d'éventuelles radiations dans une centrale nucléaire.

2 – Mise en place du projet

Pour ce projet back-end complet, l'utilisation du middleware **Express** dans son acception architecture MVC (Modèle/Vue/Contrôleur) est nécessaire.

- ***Création de l'application***

T2.1 | Exécutez les commandes ci-dessous :

```
$ express -c stylus --view=ejs qrident
$ cd qrident
qrident$ npm install
```

T2.2 | Procédez aux changements de versions nécessaires résultant de l'instruction `npm outdated` et en effectuant ensuite un `npm update` ; **pensez d'abord à supprimer le fichier** `package-lock.json` (*pourquoi ?* → vous devez en connaître la réponse)

- ***Installation d'un module nécessaire***

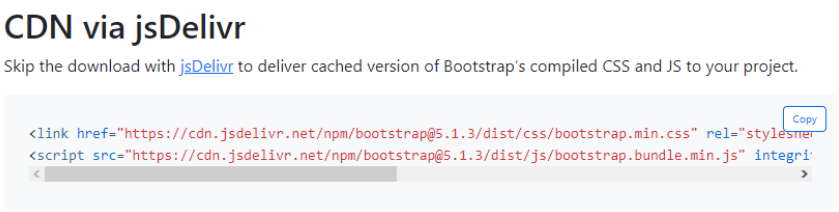
Pour cette application, et comme nous allons créer des « QR Code », nous aurons besoin du module permettant d'en générer et donc la documentation se trouve ici :

https://www.npmjs.com/package/qrcode

T2.3 | Installez le module avec l'option `--save` pour une installation dans le projet et non globale.

- ***Installation du framework Bootstrap***

L'intégration du framework CSS Bootstrap se fera de la façon la plus simple possible, c'est à dire par l'intermédiaire des liens CDN :



Liens que vous trouverez sur le site de Bootstrap :
https://getbootstrap.com/docs/5.1/getting-started/download/

T2.4 | Prenez le fichier `index.ejs` et l'image dans l'archive fournie et utilisez-les pour, en lançant l'application par `nodemon` obtenir ceci :

3 – Mise en place du QRCode

Cette mise en place comportera une partie liée à la vue et une pour son traitement au niveau du code.

- ***Initialisations***

En l'occurrence : la construction d'un nouveau contrôleur impose une nouvelle vue.

T3.1 | Créez un contrôleur `createqr.js` dans le répertoire des routes et un fichier `createqr.ejs` dans le répertoire des vues.

T3.2 | Cette nouveau contrôleur nécessite deux lignes supplémentaires dans `app.js`, faites-le… à vous de trouver !

T3.2 | Mettez tout d'abord dans ce contrôleur les lignes ci-dessous :

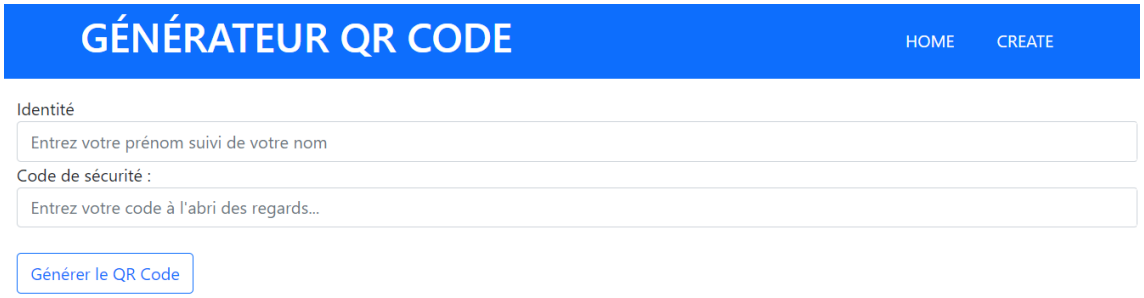
```
var express = require('express');
var router = express.Router();
var path = require('path');

// Page initiale
router.get('/', function(req, res, next) {
  res.render('createqr', { title: 'Générateur QR Code' });
});

module.exports = router;
```

T3.4 | Mettez ensuite le code dans la vue `createqr.ejs` :

```
<!DOCTYPE html>
<html>
```

```
  <head>
    <title><%= title %></title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
crossorigin="anonymous">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p"
crossorigin="anonymous"></script>
  </head>
  <body>

    <nav class="navbar navbar-expand-lg bg-primary text-uppercase">
      <div class="container">
        <h1 class="text-white"><%= title %></h1>
        <div class="collapse navbar-collapse" id="navbarResponsive">
          <ul class="navbar-nav ms-auto">
            <li class="nav-item mx-0 mx-lg-1"><a class="nav-link py-3 px-0 px-lg-3
rounded text-white" href="/">Home</a></li>
            <li class="nav-item mx-0 mx-lg-1"><a class="nav-link py-3 px-0 px-lg-3
rounded text-white" href="/createqr">Create</a></li>
        </ul>
        </div>
      </div>
    </nav>

    <header class="masthead text-center">
      <div class="container d-flex align-items-center flex-column">
       <p></p>
      </div>
    </header>

    <div class="container-fluid align-items-center">
      <div class="form-group">
        <form action="/createqr/scan" method="POST" class="form">
          <div class="form-group">
            <label for="identite">Identité</label>
            <input type="text" class="form-control" name="identite" id="identite"
placeholder="Entrez votre prénom suivi de votre nom">
          </div>
          <div class="form-group">
            <label for="code">Code de sécurité :</label>
            <input type="password" class="form-control" id="code" placeholder="Entrez
votre code à l'abri des regards...">
          <div class="form-group">
          <br />
          <button type="submit" class="btn btn-outline-primary">Générer le QR
Code</button>
        </form>
      </div>
    </div>

  </body>
</html>
```

Tout ceci doit aboutir à la page ci-dessous :



- ***Traitement de la création***

Cette vue contient un formulaire dont l'action doit correspondre à une méthode POST. Dans le code de la vue, la méthode s'intitule `scan`.

T3.5 | Intégrez tout d'abord en haut du fichier du contrôleur l'inclusion du module **qrcode** :

```
var qrcode = require("qrcode");
```

T3.6 | Ajoutez ensuite la méthode POST traitant du retour du formulaire :

```
router.post("/scan", (req, res, next) => {
    let input_identite = req.body.identite;
    let input_text = req.body.champsaisie;
    let contenuQR = input_identite + "\n" + input_text;

    qrcode.toDataURL(input_text, (err, src) => {
      if (err) res.send("Un problème est survenu !!!");

      res.render("createqr", {
        title: "Générateur QR Code",
        saisie: false,
        qr_code: src
      });
    });
});
```

L'affichage du QRCode est envoyé (on le voit par le `res.render`) dans la même vue, il faut donc utiliser le template EJS afin d'afficher des vues alternatives.

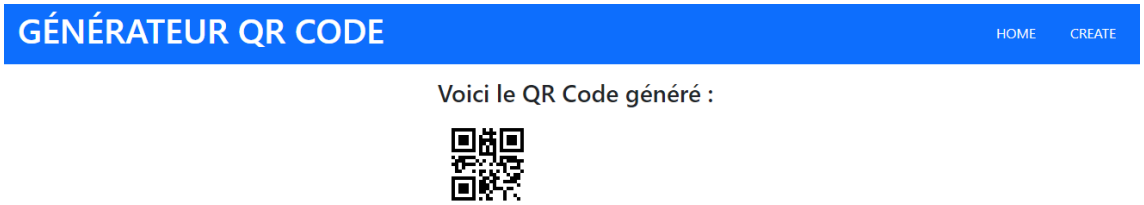
A ce propos vous trouverez la documentation d'EJS ici :
https://ejs.co/#docs

Le principe d'EJS en termes de code sera :

```
<% if (saisie) { %>
   ...
<% } else { %>
   <h3>Voici le QR Code généré :</h3>
   <img src=<%=qr_code %> alt="QR Code">
<% } %>
```

T3.7 | Ajoutez dans le contrôleur la variable `saisie` (à la bonne valeur et en fonction du type que vous devez trouver !) afin d'être transmise dans la vue.

T3.8 | Modifiez la vue de façon à ce qu'une vue de génération donne :



T3.9 | Installez un lecteur de QRCode sur votre smartphone et vérifiez la bonne saisie des informations.

4 – MongoDB et l'ORM Mongoose

- ***Principes***

L'utilisation d'un ORM pour l'établissement d'une correspondance entre une base de données et les objets JavaScript utilisés par NodeJS s'impose pour des traitements plus complexes.

Ici, nous allons utiliser **Mongoose** pour gérer l'enregistrement des QR Code générés.

- ***La dépendance supplémentaire pour Mongoose***

Mongoose est une librairie JavaScript à classer dans la catégorie des **ODM** (*Object Data Mapping*), en gros équivalent **ORM** (*Object Relational Mapping*) des **SGBDR**. Mongoose transforme des données MongoDB en des objets utilisables sous NodeJS / ExpressJS.

T4.1 | Lancez NPM pour l'installation des dernières versions des modules **MongoDB** et **Mongoose** (**4.4.1** et **6.2.4**) pour l'application.

- ***La modélisation de l'enregistrement***

Puisque nous sommes dans une structure de type MVC, avec l'ODM Mongoose on construit donc un modèle pour l'enregistrement d'un QRCode que l'on va sauvegarder.

T4.2 | Créez un répertoire `models` à la racine de l'application

T4.3 | Dans ce répertoire, créez un fichier nommé `utilisateur.js` chargé de la description (schéma) de l'enregistrement à sauvegarder (c'est une base ou exemple) :

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

let Utilisateur = new Schema({
  identite: String,
  code: String,
  poste: String,
});

module.exports = mongoose.model('Utilisateurs', Utilisateur);
```

Trois items dans ce schéma de type `string`, pour faire au plus simple… On a donc :

- le champ `identite` : *nom et le prénom de la personne*
- le champ `code` : *code de sécurité de la personne*
- le champ `poste` : *nature du poste de la personne*

- ***Modification du fichier `app.js`***

Classiquement là aussi il faut ajouter des lignes dans le fichier `app.js` ; attention par contre à l'emplacement de ces lignes.

T4.4 | Ajoutez dans le fichier `app.js` le recours au module, à la fin des premiers `require` :

```
const mongoose = require('mongoose');
```

Pour le fichier `app.js`, il ne reste plus qu'à configurer Mongoose.

T4.5 | Procédez à l'ajout des lignes ci-dessous APRÈS les lignes concernant les utilisations des routes et AVANT l'`app.use` de l'erreur 404 :

```
// Configuration de mongoose
mongoose.connect('mongodb://localhost/qrcodes', {
  useUnifiedTopology: true, useNewUrlParser: true });
let mdb = mongoose.connection;
  mdb.on('error', console.error.bind(console, 'connection error:'));
  mdb.once('open', function() {
    console.log("Connection Mongoose ok !");
});
```

Il est évident que vous devez avoir MondoDB d'installer sur votre système et un outil comme Robo3T.

T4.6 | Pour tester le bon démarrage de l'ensemble, démarrez d'abord MongoDB, ensuite l'application **QRCodes** pour obtenir en mode console la phrase de bonne connexion Mongoose :

```
[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./bin/www`
Connection Mongoose ok !
```

T4.7 | Utilisez le logiciel **Robo3T** (ou **Compass**, ou en mode **console** ;-)) pour créer la base **QRCodes** et la collection `Utilisateurs`.

Exemple de commande :

```
db.Utilisateurs.insert({ "identite":"Yvan Samob", "code":"YS2000", "poste":"Technicien"})
```

- ***Affichage en tableau des utilisateurs***

Pour cette nouvelle fonctionnalité, il va nous falloir une nouvelle page.

T4.8 | Créez le contrôleur `utilisateurs.js`, la vue `utilisateurs.pug` et ajoutez la ligne du menu (avec le bon chemin !) qui manque ; n'oubliez pas les ajouts à faire dans le fichier `app.js`.

T4.9 | Intégrez le code ci-dessous dans `utilisateurs.js`, somme toutes assez simple :

```
var express = require('express');
var router = express.Router();
const Utilisateurs = require("../models/utilisateur");

// Page racine
router.get("/", async function (req, res, next) {
  Utilisateurs.find({}, function (err, result) {
    if (err) {
      res.send(err);
    } else {
      res.render("utilisateurs", {
        title: "Générateur QR Code",
        utilisateurs: result });
    }
  });
});

module.exports = router;
```

T4.10 | Et bien sûr, la vue associée du tableau :

```
<!DOCTYPE html>
<html>
  <head>
    <title><%= title %></title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
crossorigin="anonymous">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p"
crossorigin="anonymous"></script>
  </head>
  <body>

    <nav class="navbar navbar-expand-lg bg-primary text-uppercase">
      <div class="container">
        <h1 class="text-white"><%= title %></h1>
        <div class="collapse navbar-collapse" id="navbarResponsive">
          <ul class="navbar-nav ms-auto">
            <li class="nav-item mx-0 mx-lg-1"><a class="nav-link py-3 px-0 px-lg-3
rounded text-white" href="/">Home</a></li>
            <li class="nav-item mx-0 mx-lg-1"><a class="nav-link py-3 px-0 px-lg-3
rounded text-white" href="/createqr">Create</a></li>
            <li class="nav-item mx-0 mx-lg-1"><a class="nav-link py-3 px-0 px-lg-3
rounded text-white" href="/utilisateurs">Utilisateurs</a></li>
        </ul>
        </div>
      </div>
    </nav>

    <header class="masthead text-center">
      <div class="container d-flex align-items-center flex-column">
       <p></p>
      </div>
    </header>

    <div class="container-fluid align-items-center">
      <h1 class="cover-heading">Table du personnel</h1>
      <div class="container bg-light text-dark">
        <table class="table table-striped">
          <thead>
            <tr>
              <th>Identité</th>
              <th>Code</th>
              <th>Poste</th>
            </tr>
          </thead>
          <tbody>
            <% utilisateurs.forEach(function(entry) { %>
            <tr>
              <td><%= entry.identite %></td>
              <td><%= entry.code %></td>
              <td><%= entry.poste %></td>
            </tr>
            <% }); %>
          </tbody>
        </table>
      </div>
    </div>

  </body>
</html>
```

Exemple de résultat :

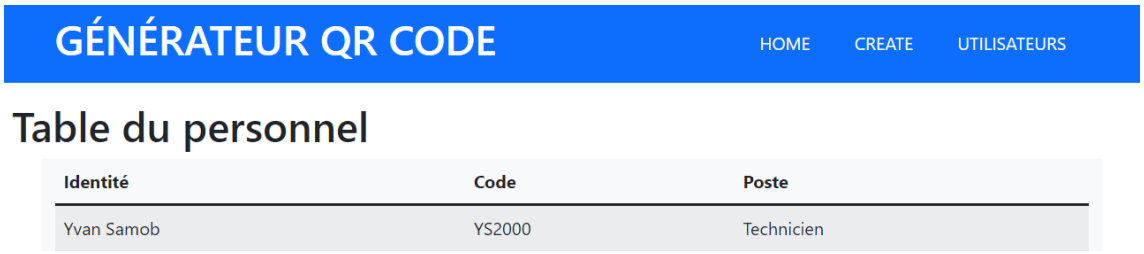


Table du personnel

Identité	Code	Poste
Yvan Samob	YS2000	Technicien