|  |  |
| --- | --- |
|  | **Javascript côté serveur** |
| **ExpressJS et Mongoose** | |

Installation de MongoDB

1. Télécharger et installe MongoDB
2. Créer un dossier appelé « mongodb »
3. A l’intérieur de ce dossier créer un sous-dossier appelé « data »
4. Dans le dossier mongodb, aller sur la ligne de commande et exécuter la commande suivante pour démarrer le serveur BD :

mongod --dbpath=data --bind\_ip 127.0.0.1

Configuration de Mongoose :

1. Installation de mongoose dans notre dossier de travail

npm install mongoose –save

Le flag --saveajoute les dépendances et leurs versions au fichier package.json

1. Préparer un fichier de configuration qu’on appellera par exemple **database/mongodb.json**

{

"mongo":{

"uri":"mongodb://localhost:27017/mydb"

}

}

1. Dans app.js configurer Mongoose et connecter la base de données

|  |
| --- |
| //import database var mongoose = require('mongoose'); var configDB=require('./database/mongodb.json');  //mongo config const connect = mongoose.connect( |
| configDB.mongo.uri ,  { useNewUrlParser: true , useUnifiedTopology: true  },  ()=> console.log("Connected to DB !!") ); |

Configuration de Modèle :

1. Ajouter le modèle représentant l’entité Contact sous models/contact.js

|  |
| --- |
| const mongoose = require('mongoose'); var Schema = mongoose.Schema;  var Contact= new Schema(  {  FullName : String,  Phone : Number  }  ); module.exports = mongoose.model('contacts', Contact); |

Nous avons inclus le module Mongooseavec une référence de Schema. On commence toujours par définir le schéma puis on le relie avec la collection «contacts».

Configuration des routes :

1. Créer le nouveau fichier des routes associées au modèle contact sous « routes/contacts.js » et l’ajouter dans app.js
2. On a besoin de mettre à jour le fichier contacts.js pour ajouter (via POST) et de lire (via GET) les données(data) à partir de la collection Mongo.

9.1. Ajouter la requête GET avec le path (/contacts/) pour afficher la liste de tous les contacts :

|  |
| --- |
| var Contact= require('../models/contact')    router.get('/', function(req, res, next) {  Contact.find(  (err, contacts) => { res.render(  'form.twig',  { title : "Contact list", cont : contacts  }  );  }  ); |

});

* + - Pour utiliser le moteur de template Twig, il faudrait l’installer avec la commande

Npm install twig –save

Il est possible d’utiliser le moteur de template Jade

* + - Et le changer dans app.js comme suit :

app.set('view engine', 'twig');

* + - Créer un nouveau fichier «form.twig» sous le répertoire views

<html>

<body>

<h1> {{ title }} </h1>

<ul>

{% for c in cont %}

<li>{{ c.FullName }}</li>

{% endfor %}

</ul>

</body>

</html>

title et cont est une variable renvoyée à partir de ./routes/form.js

9.2. Ajouter la requête POST avec le path (/contacts/) pour poster un nouveau contact dans la base de données

|  |
| --- |
| router.post('/', function(req, res, next) { new Contact({  FullName : req.body.FullName,  Phone : req.body.Phone  })  .save(  (err,newcontact)=>{ if (err)  console.log("error message : "+err); else{ console.log(newcontact);  res.json(" : Contact :" + newcontact.\_id +" added");  }  }  ); |

**11.Travail à faire :**

Faire les modifications nécessaires pour :

* Afficher l’interface d’ajout d’un contact
* Rechercher un contact par son id
* Supprimer un contact
* Modifier un contact