# Vue.js Express & Sql

Vue Setup

1. npm install -g @vue/cli
2. vue --version (to check version)
3. vue init webpack client
4. navigate to folder then run npm i
5. run npm run dev **or** npm run start
6. If there are issues instead of running vue init webpack client run vue create client
7. Then run npm run serve

Nodejs Setup

1. Create backend folder
2. Run npm init
3. Npm I nodemon eslint –save

In package.js under scripts insert the following "start": "nodemon src/app.js --exec \"npm run lint && node\"",

    "lint": "eslint \*\*/\*.js"

1. If there is an error on npm start, initialise eslint by inserting following command:

node ./node\_modules/eslint/bin/eslint.js --init

1. Create a folder called src have a app.js in there and run a console log then test by running npm start
2. npm i express body-parser cors morgan –save
3. in the app .js setup server
4. const express = require('express');
5. const bodyParser = require('body-parser');
6. const cors = require('cors');
7. const morgan = require('morgan');
8. const app = express();
9. app.use(morgan('combined'));
10. app.use(bodyParser.json());
11. app.use(cors());
12. app.get('/status', (req, res) => {
13. res.send({
14. message: "We are within"
15. })
16. })
17. app.listen(8081)

Vue Setup continued

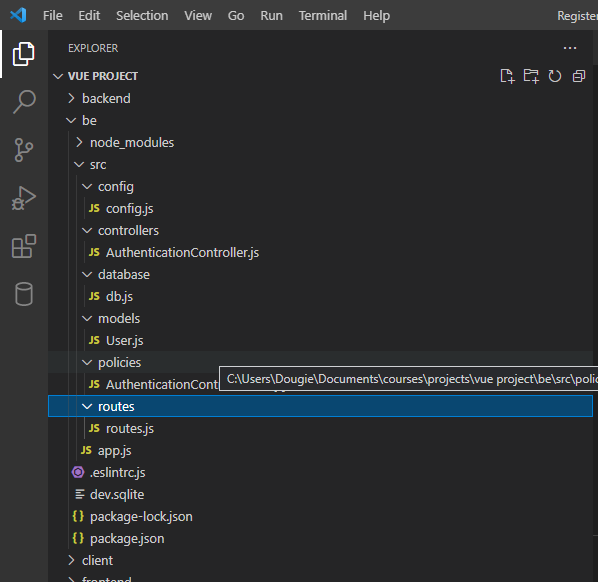
1. npm i axios (this will help with http requests)
2. create services folder the add api.js file to connect to backend with this code
3. import axios from 'axios';
4. export default () => {
5. return axios. create({
6. baseURL: `http://localhost:8082/`
7. })
8. }
9. Create a authenticationService.js file
10. The sample code should look like this
11. import api from '@/services/api';
12. export default {
13. register (credentials) {
14. return api().post('register', credentials);
15. }
16. }
17. Then create the register endpoint both on backend and frontend.
18. On frontend go to router/index.js and setup new component route
19. import Vue from 'vue'
20. import Router from 'vue-router'
21. import HelloWorld from '@/components/HelloWorld'
22. import Register from '@/components/Register'
23. Vue.use(Router)
24. export default new Router({
25. routes: [
26. {
27. path: '/',
28. name: 'HelloWorld',
29. component: HelloWorld
30. },
31. {
32. path: '/register',
33. name: 'Register',
34. component: Register
35. }
36. ]
37. })
38. The under componentrs create a new file called Register.vue and test it by having the following sample code
39. <template>
40. <div> //this roor elemet is important everything is for this template
41. <h1>Register</h1> //should be under this div
42. </div>
43. </template>
44. <script>
45. export default {
46. name: 'Register',
47. data () {
48. return {
49. }
50. }
51. }
52. </script>
53. <!-- Add "scoped" attribute to limit CSS to this component only -->
54. <style scoped>
55. </style>
56. After Testing this we can then update the component to set input fields, two way data binding using v-model as follows
57. <template>
58. <div>
59. <h1>Register</h1>
60. <input type="email" name="email" v-model="email" placeholder="Enter Email"> <br>
61. <input type="password" name="password" v-model="password" placeholder="Enter Password"><br>
62. <button @click="register">Register</button>
63. </div>
64. </template>
65. <script>
66. import authenticationService from '../services/authenticationService'
67. export default {
68. name: 'Register',
69. data () {
70. return {
71. // v-model is a two way binding which we can grab in this controller
72. email: '',
73. password: ''
74. }
75. },
76. methods: {
77. async register () {
78. const response = await authenticationService.register({
79. email: this.email,
80. password: this.password
81. })
82. console.log(response.data)
83. }
84. }
85. }
86. </script>
87. <!-- Add "scoped" attribute to limit CSS to this component only -->
88. <style scoped>
89. </style>

# Backend connect successfully and can receive a response

# Sequelize

1. It provides you with Data models where each models maps to table in the database
2. Npm I sequalize –save
3. Npm I sqlite3 –save
4. Create routes folder and route.js
5. Create a models folder for the database objects and include 2 files. Index.js and User.js
6. Create a config folder and config.js for database config and port config
7. In The .eslint.json under env, change browser to node
8. App.js should look like this:
9. const express = require('express');
10. const bodyParser = require('body-parser');
11. const cors = require('cors');
12. const morgan = require('morgan');
13. const {sequelize} = require('./models')
14. const config = require('./config/config')
15. const app = express();
16. app.use(morgan('combined'));
17. app.use(bodyParser.json());
18. app.use(cors());
19. require('./routes/routes')(app)
20. sequelize.sync()
21. .then(() => {
22. app.listen(config.port)
23. console.log(`Server started on ${config.port}`)
24. })
25. Then **routes/routes.js** should look like this:
26. module.exports = (app) => {
27. app.post('/register', (req, res) => {
28. res.send({
29. message: `user ${req.body.email} registered`
30. })
31. })
32. }
33. Then **models/User.js** should look like this:
34. module.exports = (sequelize, DataTypes) =>
35. sequelize.define('User', {
36. email: {
37. type: DataTypes.STRING,
38. unique: true
39. },
40. password: {
41. type: DataTypes.STRING
42. }
43. })
44. Then **models/index.js** should look like this:
45. const fs = require('fs')
46. const path = require('path')
47. const Sequelize = require('sequelize')
48. const config = require('../config/config')
49. const db = {}
50. const sequelize = new Sequelize (
51. config.db.database,
52. config.db.user,
53. config.db.password,
54. config.db.options
55. )
56. // Helper methods to go through eaxh file
57. fs
58. .readdirSync(\_\_dirname)
59. .filter((file)=> {
60. file !== 'index.js'
61. })
62. .forEach((file) => {
63. const model = sequelize.import(path.join(\_\_dirname, file))
64. db[model.name] = model
66. })
67. db.sequelize = sequelize
68. db.Sequelize = Sequelize
69. module.exports = db
70. Then **config/config.js** should look like this:
71. module.exports = {
72. port: process.env.PORT || 8082,
73. db: {
74. database: process.env.DB\_NAME || 'tabtracker',
75. user: process.env.DB\_USER || 'tabtracker',
76. password: process.env.DB\_PASSWORD || 'tabtracker',
77. options: {
78. dialect: process.env.DIALECT || 'sqlite',
79. host:process.env.HOST || 'localhost',
80. storage: './tabtracker.sqlite'
81. }
82. }
83. }
84. Now we can simplify the routes file by creating a controller folder and have specific controllers for certain tasks. In this instance we will create a controller that will handle all requests pointing to authentication endpoint such us register and signup
85. Then take it back to the routes file
86. **Controlers/AuthenticationController** must look like this
87. module.exports = {
88. register (req, res) {
89. res.send({
90. message: `user ${req.body.email} registered`
91. })
92. }
93. }
94. Then **routes/routes.js** should look like this:
95. module.exports = {
96. register (req, res) {
97. res.send({
98. message: `user ${req.body.email} registered`
99. })
100. }
101. }

///////////////////////////////////////////////////////////////////////////////////////////////////////

At this point folder must now look like this. Add necessary folders/files

1. Then **app.js** should look like this:
2. const express = require('express')
3. const bodyParser = require('body-parser');
4. const cors = require('cors');
5. const morgan = require('morgan');
6. const sequelize = require('./database/db')
7. const config = require('./config/config')
8. const app = express()
9. app.use(morgan('combined'));
10. app.use(bodyParser.json());
11. app.use(cors());
12. require('./routes/routes')(app)
13. sequelize.sync({ force: true})
14. .then(() => {
15. console.log('db is ready')
16. })
17. app.listen(config.port, () => {
18. console.log('App is running')
19. })
20. Then **routes/route.js** should look like this:
21. const AuthenticationController = require('../controllers/AuthenticationController')
22. const AuthenticationControllerPolicy = require('../policies/AuthenticationControllerPolicy')
23. module.exports = (app) => {
24. app.post('/register',
25. AuthenticationControllerPolicy.register,
26. AuthenticationController.register)
27. }
28. Then **policies/AuthenticationPolicy.js** should look like this:
29. // we will verify the data wether it will pass or fail
30. const Joi = require('joi')
31. module.exports = {
32. register (req, res, next) {
33. const schema = {
34. email: Joi.string().email(),
35. password: Joi.string().regex(
36. new RegExp('^[a-zA-Z0-9]{8,32}$')
37. )
38. }
39. const {error} = Joi.validate(req.body, schema)
40. if (error) {
41. switch (error.details[0].context.key) {
42. case 'email':
43. res.status(400).send({
44. error: 'You must provide a valid email address'
45. })
46. break
47. case 'password':
48. res.status(400).send({
49. error: `The password provided failed to match the following rules:
50. <br>
51. 1. It must contain ONLY the following characters: lower case, upper case, numerics.
52. <br>
53. 2. It must be at least 8 characters in length and not greater than 32 characters in length.
54. `
55. })
56. break
57. default:
58. res.status(400).send({
59. error: 'Invalid registration information'
60. })
61. }
62. } else {
63. next()
64. }
65. }
66. }
67. Then **models/user.js** should look like this:
68. const { Model,DataTypes} = require('sequelize')
69. const sequelize = require('../database/db')
70. class User extends Model {}
71. User.init({
72. email: {
73. type: DataTypes.STRING,
74. unique: true
75. },
76. password: {
77. type: DataTypes.STRING
78. }
79. }, {
80. sequelize,
81. modelName:'user',
82. timestamps: false
83. })
84. module.exports = User
85. Then **database/db.js** should look like this:
86. const { Sequelize } = require('sequelize')
87. const config = require('../config/config')
88. const sequelize = new Sequelize(config.db.database, config.db.user, config.db.password, config.db.options)
89. module.exports = sequelize
90. Then **controllers/authenticationController.js** should look like this:
91. const User = require('../models/User')
92. // const AnotherModel = require('../models/TherequiredModel)
93. module.exports = {
94. async register (req, res) {
95. try {
96. console.log(req.body)
97. const user = await User.create(req.body)
98. res.send(user)
99. } catch (err) {
100. res.status(400).send({
101. error: 'This email account is already in use.'
102. })
103. console.log(err.message)
104. }
105. }
106. }
107. Then **config/config.js** should look like this:
108. module.exports = {
109. port: process.env.PORT || 3001,
110. db: {
111. database: process.env.DB\_NAME || 'test-db',
112. user: process.env.DB\_USER || 'user',
113. password: process.env.DB\_PASSWORD || 'pass',
114. options: {
115. dialect: process.env.DIALECT || 'sqlite',
116. host:process.env.HOST || 'localhost',
117. storage: './dev.sqlite'
118. }
119. }
120. }
121. Then **package.json** should look like this:
122. {
123. "name": "be",
124. "version": "1.0.0",
125. "description": "backend2",
126. "main": "index.js",
127. "scripts": {
128. "start": "nodemon src/app.js --exec \"npm run lint && node\"",
129. "lint": "eslint \*\*/\*.js",
130. "test": "echo \"Error: no test specified\" && exit 1"
131. },
132. "author": "Dougie",
133. "license": "MIT",
134. "dependencies": {
135. "@hapi/joi": "^15.0.3",
136. "body-parser": "^1.20.0",
137. "cors": "^2.8.5",
138. "express": "^4.18.1",
139. "joi": "^10.6.0",
140. "morgan": "^1.10.0",
141. "nodemon": "^2.0.16",
142. "sequelize": "^6.19.2",
143. "sqlite3": "^5.0.8"
144. },
145. "devDependencies": {
146. "eslint": "^8.15.0",
147. "eslint-plugin-vue": "^9.0.1"
148. }
149. }