Web Programming

Server Node.JS – Client Vue.JS - CRUD

1 - Setting up a Github Repository

Open your github account and create a repository. https://github.com/

In your computer, create a folder that will be used to connect to your git repository.

```
i.e.: cd documents/project-vue-git
```

Add your local folder as a workspace in your Visual Studio Code.

Open the terminal and check if git is installed on your machine with the command

```
# git --version
```

If not, install git by downloading it from the following link: https://git-scm.com/downloads

Copy your git https access address from your git account.

In Visual Studio Code, open the terminal in your local project

Clone it in the terminal using:

```
# git clone https://github.com/<your account>/<your project >
```

Navigate to your git project: In this tutorial, we will use the git project called:

```
#cd project-vue
```

You will create 2 main folders inside the project.

One folder for the server and another for the vue-js project.

Server will run all backend queries, node js and database requests (MySQL).

2 -Setting up the Server

Before moving, you must have a local server with MySQL database installed on your computer to run the local database, for example: (XAMPP, MAMP, Wamp) if you don't have one, please download from this link: https://www.apachefriends.org/index.html

Start your localhost services (XAMPP)

Return to Visual Studio Code terminal and in the project-vue directory create a new folder called server, you can do it using the code.

```
# mkdir server
```

Create a node.js project with a package.json to do this in the server folder, run the command.

```
# cd server

# npm init
```

Complete the following configuration:

```
package name: (server) server

version: (1.0.0)

description: server with nodejs, express, sequelize, cors, mysql

entry point: (index.js) server.js

test command:

git repository:

keywords: node, express, sequelize, mysql, cors

author:

license: (ISC)
```

And press yes to save the package.json

We now need to install the necessary modules: express, sequelize, mysql2 and body-parser.

- Express is for building the Rest API : https://expressjs.com/
- body-parser helps to parse the request and create the req.body object:https://www.npmjs.com/package/body-parser
- cors provides Express middleware to enable CORS with various options. https://expressjs.com/en/resources/middleware/cors.html

Run the command to install it

```
# npm install express sequelize mysql2 body-parser cors --save
```

Create the server.js file that will be used to start your server.
In this file type the following code:

server/server.js

```
const express = require('express')
const express = require('express')
const bodyParser = require('body-parser')
const cors = require('cors')
const app = express()
const corsOptions = {
    origin: 'http://localhost:8081'
}
app.use(cors(corsOptions))
//models
const db = require('./app/models')
//sequelize
db.connex.sync()
//  db.sequelize.sync({ force: true }).then(() => {
//     console.log("Drop and re-sync db.")
//   });
app.use(bodyParser.json())
app.use(bodyParser.urlencoded({ extended: true}))
//test
app.get('/', (req, res) => {
    res.json({message: 'Welcome'})
})
//routes
require('./app/routes/product.route')(app)
const PORT = process.env.PORT || 8080
app.listen(PORT, () => {
    console.log(`Server is running on port ${PORT}.`)
})
```

Into server directory, create an app folder:

In the *app* folder, create a separate *config* folder for configuration with *db.config.js* file like this:

Create a database in your local server: for the project the database name will be db_product.

```
# CREATE DATABASE db_product;
```

Use your own credentials to fill up this file.

server/app/config/db.config.js

```
module.exports = {
    HOST: 'localhost',
    USER: 'root',
    PASSWORD: '',
    DB: 'db_product',
    dialect: "mysql",
  };
```

Initialize Sequelize

Sequelize use a MVC (Model-View-Controller) architecture to manage the database request.

Create a models folder under the app where we will manage all data models.

Create the index.js file for the models.

server/app/models/index.js

```
const dbConfig = require ('../config/db.config.js')
const Sequelize = require('sequelize')
const connex = new Sequelize(dbConfig.DB, dbConfig.USER, dbConfig.PASSWORD, {
    host: dbConfig.HOST,
    dialect: dbConfig.dialect,
    operatorAliases: false
})
const db = {}
db.Sequelize = Sequelize
db.connex = connex
db.products = require('./product.model.js')(connex, Sequelize)
module.exports = db
```

Define the first model. Create a file *product.model.js* like this.

server/app/model/product.model.js

```
module.exports = (connex, Sequelize) => {
    const Product = connex.define('product',{
        name:{
            type: Sequelize.STRING
        },
        photo:{
            type: Sequelize.STRING
        },
        price:{
            type: Sequelize.REAL
        },
        description:{
            type: Sequelize.TEXT
        },
        type:{
            type: Sequelize.STRING
        }
    })
    return Product
}
```

You can check the allowed data types here: https://sequelize.org/v5/manual/data-types.html

This Sequelize model represents the products table in the MySQL database. These columns will be generated automatically: id, name, photo, price, description, type, createdAt, updatedAt.

After initializing Sequelize, we don't need to write any CRUD functions, Sequelize supports them all:

- create a new data: `create(object)`
- get all data: `findAll()`
- find a data by id: `findByPk(id)`
- update a data by id: `update(data, where: { id: id })`
- remove a data: `destroy(where: { id: id })`
- remove all data: `destroy(where: {})`
- find all data by type: `findAll({ where: { type: ... } })`

Controller

Inside the app create a folder called controllers and inside it creates the product.controller.js file with these CRUD functions:

- create
- findAll
- findOne
- update
- delete
- deleteAll
- findAll

server/controllers/product.controller.js

```
const db = require('../models')
const Product = db.products
const Op = db.Sequelize.Op

exports.create = (req, res) => {
  //console.log(req.body)
  Product.create(req.body)
    .then(data => {
        res.send(data)
    })
    .catch(err => {
        res.status(500).send({
            message:
              err.message || 'could not insert data'
        })
    })
}
exports.findAll = (req, res) => {
    //console.log('you are here')
    Product.findAll()
    .then(data => {
        res.send(data)
    })
    .catch(err => {
        res.status(500).send({
            message:
                err.message || 'Some error occurred while loading the data'
        })
    })
}
exports.findOne = (req, res) => {
    //console.log(req.params.id)
    const id = req.params.id
    Product.findByPk(id)
    .then(data => {
        res.send(data)
    })
    .catch(err => {
        res.status(500).send({
            message:
                err.message || `Some error occurred while loading the data id
${id}`
        })
    })
}
exports.update = (req, res) => {
    const id = req.params.id;
    Product.update(req.body, {
      where: { id: id }
    })
      .then(num => {
        if (num == 1) {
          res.send({
            message: 'Product was updated successfully.'
          });
        } else {
          res.send({
            message: `Cannot update product with id=${id}. Maybe product was not
found or is empty!`
          });
        }
      })
      .catch(err => {
        res.status(500).send({
          message: `Error updating Produt with id=${id}`
        })
      })
  }
exports.delete = (req, res) => {
    const id = req.params.id;
    Product.destroy({
      where: { id: id }
    })
      .then(num => {
        if (num == 1) {
          res.send({
            message: "Product was deleted successfully!"
          })
        } else {
          res.send({
            message: `Cannot delete product with id=${id}. Product not found!`
          })
        }
      })
      .catch(err => {
        res.status(500).send({
          message: `Could not delete Porduct with id=${id}`
        })
      })
  }
```

Routes

Finally, let's connect the model to the controllers using a route. Create a Routes folder under app and in this folder create the file named product.route.js

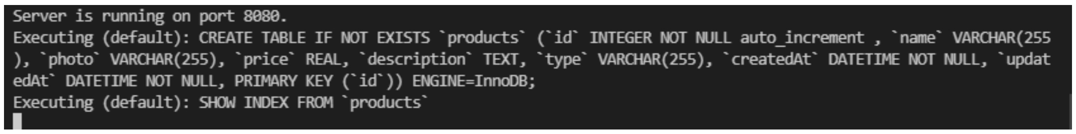
server/app/routes/product.route.js

```
module.exports = app => {
    const product = require('../controllers/product.controller.js')

    const router = require('express').Router()
    //insert new data
    router.post('/', product.create)
    // retrieve all data
    router.get('/', product.findAll)
    //get one data
    router.get('/:id', product.findOne)
    //updat data
    router.put('/:id', product.update)
    //delete by id
    router.delete('/:id', product.delete)

    app.use('/api/product', router)
}
```

We can now test the server. To do this, under the server directory, run the command

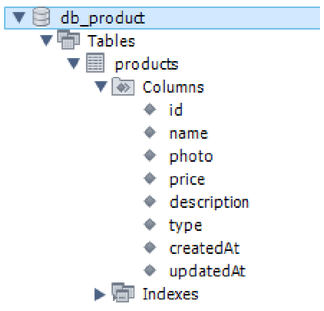
```
#server node.js
```

After running your server, you will receive a message that the table has been created.



```
Server is running on port 8080.
Executing (default): CREATE TABLE IF NOT EXISTS `products` (`id` INTEGER NOT NULL auto_increment , `name` VARCHAR(255
), `photo` VARCHAR(255), `price` REAL, `description` TEXT, `type` VARCHAR(255), `createdAt` DATETIME NOT NULL, `updat
edAt` DATETIME NOT NULL, PRIMARY KEY (`id`)) ENGINE=InnoDB;
Executing (default): SHOW INDEX FROM `products`
```

You can go to your database to confirm it.

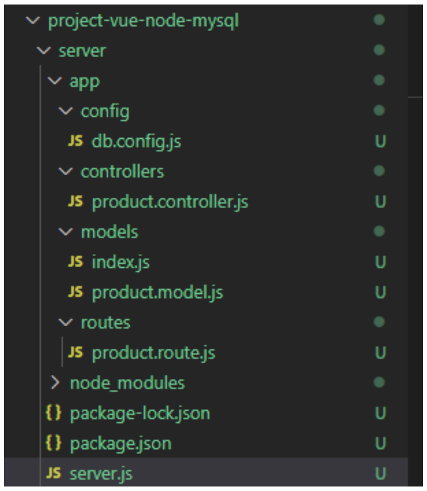


```
▼ db_product
  ▼ Tables
    ▼ products
      ▼ Columns
          id
          name
          photo
          price
          description
          type
          createdAt
          updatedAt
      ▶ Indexes
```

Open your browser and type: http://localhost:8080, you will see our welcome message.

Done, your server is ready to go and integrate with your Vue.js

In the end, your server folder should look like this.



```
project-vue-node-mysql
  server
    app
      config
        db.config.js                U
      controllers
        product.controller.js       U
      models
        index.js                    U
        product.model.js            U
      routes
        product.route.js            U
    node_modules
    package-lock.json               U
    package.json                    U
    server.js                       U
```

Now that the first part is done, you can commit it to git hub.

In the terminal, use ctr c to stop the server.

cd ../ to return a folder.

You should now be on project-vue-node-mysql.

To check the github status of changes run:

```
#git status
```

To save it into github run the following commands

```
#git add --all

#git commit -m 'init server with node.js'

#git push
```

You can go to your git account and verify that all the directories are pushed there in your origin/main branch. You can also check your commit history.

References: https://www.bezkoder.com/vue-js-node-js-express-mysql-crud-example/

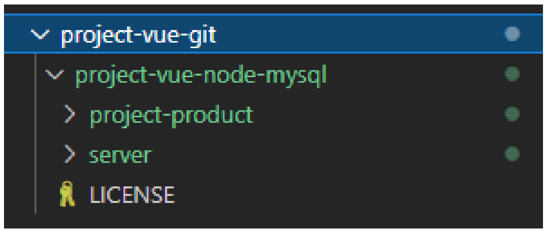
3 - Setting up Vue.Js Frontend App

Second part your our application is to create a vue.js front end project

Inside the folder project -vue-git run the following code to create a project and follow the previous tutorial to set it up. We will use the name project-product for the vue project in this tutorial.

```
# cd project-product

# vue create project-product
```

After creating the project you will see that now the project has 2 folders: server and project-product.



Server is the backend connection with the MVC model connected to the database via port 8080 (localhost:8080), and project-product is the vue.sj interface. Later we will use port 8081 to run this project.

For now, you can run the service and test the project in the browser. You should see the Vue.JS welcome page.

```
# cd project-product

# npm run serve
```

Before continuing save the vue project into git.

ctr c to stop the server.

cd ../ to return to the project-vue-node-mysql.

```
#git add --all

#git commit -m 'init vue.js project'

#git push
```

VS Code Tip:

To avoid the same fix issue, create a .vcode folder in the main project and a settings.json with the following code.

project-vue-git\.vscode\settings.json

```
{
    "eslint.workingDirectories": [
        {"mode": "auto"}
    ]
}
```

Install bootstrap in your project

```
# npm install bootstrap jquery popper.js
```

Import booststrap into the main.js file

project-product/src/main/js

```
import { createApp } from 'vue'
import App from './App.vue'
import router from './router'
import 'bootstrap'
import 'bootstrap/dist/css/bootstrap.min.css'
import './assets/style/styles.css'

const app = createApp(App)
app.use(router)
app.mount('#app')
```

Routes

Add the vue.js routes to the project In the src/routes folder set Router as following:

project-product/src/routes/index.js

```
import { createRouter, createWebHistory } from 'vue-router'
import HomeView from '../views/HomeView.vue'
import ProductView from '../views/ProductView.vue'
import ProductAddView from '../views/ProductAddView.vue'
import ProductEditView from '../views/ProductEditView.vue'
import AboutView from '../views/AboutView.vue'

const routes = [
  {
    path: '/',
    name: 'home',
    component: HomeView
  },
  {
    path: '/product/:id',
    name: 'product',
    component: ProductView
  },
  {
    path: '/product-add',
    name: 'product-add',
    component: ProductAddView
  },
  {
    path: '/product-edit/:id',
    name: 'product-edit',
    component: ProductEditView
  },
  {
    path: '/about',
    name: 'about',
    component: AboutView
  }
]

const router = createRouter({
  history: createWebHistory(),
  routes
})

export default router
```

Add the routes into the App.vue file.

project-product/src/App.vue

```
<ul class="navbar-nav me-auto mb-2 mb-lg-0 ms-lg-4">
     <li class="nav-item"><router-link class="nav-link active" aria-
current="page" to="/">All Products</router-link></li>
     <li class="nav-item"><router-link class="nav-link active" aria-
current="page" to="/product-add'">Add Product</router-link></li>
     <li class="nav-item"><router-link class="nav-link active" aria-
current="page" to="/about">About</router-link></li>
</ul>
```

Install Axios

Axios is a Javascript library used to make HTTP requests from node. js or XMLHttpRequests from browser and it supports native JS Promise API

Run the following command to install Axios

```
# npm install axios
```

Then, in src folder, we create *http-common.js* file like this:

project-product/src/http-common.js

```
import axios from 'axios'
```

```
export default axios.create({
  baseURL: 'http://localhost:8080/api',
  headers: {
    'Content-type': 'application/json'
  }
})
```

Create a data service

The service will use axios from the HTTP client above to send HTTP requests .

Under src create a folder named services and a file ProducDataService.js

project-product/src/services/ProductDataService.js

```
import http from '../http-common'
class ProductDataService {
  getAll () {
    return http.get('/product')
  }

  get (id) {
    return http.get(`/product/${id}`)
  }

  create (data) {
    return http.post('/product', data)
  }

  update (id, data) {
    return http.put(`/product/${id}`, data)
  }

  delete (id) {
    return http.delete(`/product/${id}`)
  }

  deleteAll () {
    return http.delete('/product')
  }

  findByType (type) {
    return http.get(`/product?type=${type}`)
  }
}
export default new ProductDataService()
```

At app.Vue

List of all products

In App.vue, insert the following vue script to load all products from the database and load them into the HomeView and Sidebar cart.

```
  mounted () {
    ProductDataService.getAll()
      .then(response => {
        this.inventory = response.data
        console.log(response.data)
      })
      .catch(e => {
        console.log(e)
      })
  }
```

Configure Port for Vue 3

Since most HTTP servers use a CORS configuration that accepts resource sharing limited to certain sites or ports, we also need to configure the port for our application.

In the project root folder, create the vue.config.js file with the following content:

```
const { defineConfig } = require('@vue/cli-service')
module.exports = defineConfig({
  transpileDependencies: true
})
module.exports = {
  devServer: {
    port: 8081
  }
}
```

CRUD

List all products

project-product/src/view/homeView.vue

Template

```
<template>
    <div class="home">
        <!-- Search bar -->
        <!-- Section-->
        <section class="py-5">
            <div class="container px-4 px-lg-5 mt-5">
                <div class="row gx-4 gx-lg-5 row-cols-2 row-cols-md-3 row-cols-
xl-4 justify-content-center">
```

```
                    <div v-for="(product, i) in inventory" :key="product.id"
class="col mb-5">
                        <div class="card h-100">
                            <!-- Product image-->
                            <img class="card-img-top"
:src="require('@/assets/img/450/'+product.photo)"  :alt="product.name" />
                            <!-- Product details-->
                            <div class="card-body p-4">
                                <div class="text-center">
                                    <!-- Product name-->
                                    <h5 class="fw-bolder">{{ product.name }}</h5>
                                    <!-- Product price-->
                                    ${{ product.price.toFixed(2) }}
                                </div>
                            </div>
                            <!-- Product actions-->
                            <div class="card-footer p-4 pt-0 border-top-0 bg-
transparent">
                                <div class="text-center">
                                    <input class="form-control qt" type="number"
v-model.number="product.quantity">
                                    <a class="btn btn-outline-dark mt-auto"
@click="addToCart(product.name, i)">Add to cart</a>
                                    <br/>
                                    <router-link :to="'/product/'+product.name"
class="text-primary">See more</router-link>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </section>
    </div>
</template>
```

Vue.js

```
<script>
export default {
  props: ['inventory', 'addToCart']
}
</script>
```

View one product

project-product/src/view/productView.vue

Template

```
    <template>
    <div class="food">
      <section class="py-5">
          <div class="container px-4 px-lg-5 my-5">
              <div class="row gx-4 gx-lg-5 align-items-center">
                  <div class="col-md-6"><img class="card-img-top mb-5 mb-md-0"
:src="require('@/assets/img/600/'+product.photo)" :alt="product.name" /></div>
                  <div class="col-md-6">
                      <div class="small mb-1">Code: {{ product.id }}</div>
                      <h1 class="display-5 fw-bolder">{{ product.name }}</h1>
                      <div class="fs-5 mb-5">
                          <span class="text-decoration-line-through"></span>
                          <span>${{ product.price.toFixed(2) }}</span>
                      </div>
                      <p class="lead">{{ product.description }}</p>
                      <div class="d-flex">
                          <input class="form-control text-center me-3"
id="inputQuantity" type="num" style="max-width: 3rem" v-
model.number="product.quantity"/>
                          <button class="btn btn-outline-dark flex-shrink-0"
type="button" @click="addToCart(product.name, productIndex)">
                              <i class="bi-cart-fill me-1"></i>
                              Add to cart
                          </button>
                      </div>
                     <hr class="my-4">
                      <div class="d-flex">
                          <router-link class="btn btn-outline-success flex-
shrink-0 mt-10" :to="'/product-edit/'+product.id">
                              <i class="bi-pen me-1"></i>
                              Edit product
                          </router-link>
                      </div>
                  </div>
              </div>
          </div>
      </section>
    </div>
</template>
```

Vue.js

```
<script>
export default {
  props: ['inventory', 'addToCart'],
  computed: {
    product () {
      const product = this.inventory.find((p) => {
        return p.name === this.$route.params.id
      })
      return product
    },
    productIndex () {
      const index = this.inventory.findIndex((p) => {
        return p.name === this.$route.params.id
      })
      return index
    }
  }
}
</script>
```

Add a new product

project-product/src/view/Product-Add.vue

Template

```
<template>
    <div class="product-add">
      <!-- Section-->
      <section class="py-5">
        <div class="container px-4 px-lg-5 mt-5">
          <div
            class="
              row
              gx-4 gx-lg-5
              row-cols-2 row-cols-md-3 row-cols-xl-2
              justify-content-center
            "
          >
            <div class="col-sm-12">
              <h4 class="mb-3">Add new product</h4>
              <div class="needs-validation" novalidate>
                <div class="row g-2">
                  <div v-if="!submitted">
                  <div class="col-12">
                    <label for="productName" class="form-label"
          >Product Name</label
        >
        <input
          type="text"
          class="form-control"
          id="productName"
          placeholder=""
          v-model="product.name"
          required
        />
        <div class="invalid-feedback">Valid name is required.</div>
      </div>
      <div class="col-12">
        <label for="productPhoto" class="form-label"
          >Product Photo</label
        >
        <input
          type="text"
          class="form-control"
          id="productPhoto"
          placeholder=""
          v-model="product.photo"
          required
        />
        <div class="invalid-feedback">
          Valid photo path is required.
        </div>
      </div>
      <div class="col-12">
        <label for="productPrice" class="form-label">Price</label>
        <div class="input-group has-validation">
          <span class="input-group-text">CAD</span>
          <input
            type="text"
            class="form-control"
            id="productPrice"
            placeholder=""
            v-model.number="product.price"
            required
          />
          <div class="invalid-feedback">Price is required.</div>
        </div>
      </div>
      <div class="col-12">
        <label for="productDescription" class="form-label"
          >Product Description</label
```

```
                    >
                    <textarea
                      class="form-control"
                      id="productDescription"
                      placeholder=""
                      v-model="product.description"
                    ></textarea>
                    <div class="invalid-feedback">Valid description</div>
                  </div>
                  <div class="col-12">
                    <label for="productType" class="form-label"
                      >Product Type</label
                    >
                    <select
                      class="form-control"
                      id="productType"
                      placeholder=""
                      v-model="product.type"
                      required
                    >
                      <option value="">Select</option>
                      <option value="Burger">Burger</option>
                      <option value="Italian">Italian</option>
                      <option value="Indian">Indian</option>
                      <option value="Thai">Thai</option>
                    </select>
                    <div class="invalid-feedback">
                      Valid photo path is required.
                    </div>
                  </div>
                  <button class="w-100 btn btn-secondary btn-lg mt-3"
type="button" @click="saveProduct">Save </button>
                  </div>
                  <div v-else>
                    <div  class="alert alert-success alert-dismissible fade show"
role="alert">
                    <strong> You submitted successfully!</strong>
                    <button type="button" class="btn-close" data-bs-
dismiss="alert" aria-label="Close"></button>
                    </div>
                    <button class="w-100 btn btn-success btn-lg mt-3"
type="button" @click="newProduct">New product </button>
                  </div>
                  <hr class="my-4">
                </div>
              </div>
```

```
            </div>
          </div>
        </div>
      </section>
    </div>
</template>
```

Vue.js

```
<script>
import ProductDataService from '@/services/ProductDataService'

export default {
  props: ['addInv'],
  data () {
    return {
      submitted: false,
      product: {
        name: '',
        photo: '',
        price: '',
        description: '',
        type: ''
      }
    }
  },
  methods: {
    saveProduct () {
      ProductDataService.create(this.product)
        .then(response => {
          this.product.id = response.data.id
          this.addInv(this.product)
          this.submitted = true
        })
        .catch(e => {
          console.log(e)
        })
    },
    newProduct () {
      this.submitted = false
      this.product = {}
    }
  }
}
</script>
```

Update and delete a product

project-product/src/view/product-edit.vue

Template

```
<template>
    <div class="product-add">
      <!-- Section-->
      <section class="py-5">
        <div class="container px-4 px-lg-5 mt-5">
          <div
            class="
              row
              gx-4 gx-lg-5
              row-cols-2 row-cols-md-3 row-cols-xl-2
              justify-content-center
            "
          >
            <div class="col-sm-12">
              <h4 class="mb-3">Edit product</h4>
              <div class="needs-validation" novalidate>
                <div class="row g-2">
                  <div>
                  <div class="col-12">
                    <label for="productName" class="form-label"
                      >Product Name</label
                    >
                    <input
                      type="text"
                      class="form-control"
                      id="productName"
                      placeholder=""
                      v-model="product.name"
                      required
                    />
                    <div class="invalid-feedback">Valid name is required.</div>
                  </div>
                  <div class="col-12">
                    <label for="productPhoto" class="form-label"
                      >Product Photo</label
                    >
                    <input
                      type="text"
                      class="form-control"
                      id="productPhoto"
                      placeholder=""
                      v-model="product.photo"
                      required
                    />
                    <div class="invalid-feedback">
                      Valid photo path is required.
                    </div>
                  </div>
                  <div class="col-12">
                    <label for="productPrice" class="form-label">Price</label>
                    <div class="input-group has-validation">
                      <span class="input-group-text">CAD</span>
                      <input
                        type="text"
                        class="form-control"
                        id="productPrice"
                        placeholder=""
                        v-model.number="product.price"
                        required
                      />
                      <div class="invalid-feedback">Price is required.</div>
                    </div>
                  </div>
                  <div class="col-12">
                    <label for="productDescription" class="form-label"
                      >Product Description</label
                    >
                    <textarea
                      class="form-control"
                      id="productDescription"
                      placeholder=""
                      v-model="product.description"
                    ></textarea>
                    <div class="invalid-feedback">Valid description</div>
                  </div>
                  <div class="col-12">
                    <label for="productType" class="form-label"
                      >Product Type</label
                    >
                    <select
                      class="form-control"
                      id="productType"
                      placeholder=""
                      v-model="product.type"
                      required
                    >
```

```
                      <option value="">Select</option>
                      <option value="Burger">Burger</option>
                      <option value="Italian">Italian</option>
                      <option value="Indian">Indian</option>
                      <option value="Thai">Thai</option>
                    </select>
                    <div class="invalid-feedback">
                      Valid photo path is required.
                    </div>
                  </div>
                  <button class="w-100 btn btn-success btn-lg mt-3" type="button"
@click="updateProduct">Update </button>
                  </div>
                  <div>
                    <div v-if="submitted"  class="alert alert-success alert-
dismissible fade show" role="alert">
                    <strong> Your product was updated successfully!</strong>
                    <button type="button" class="btn-close" data-bs-
dismiss="alert" aria-label="Close"></button>
                    </div>
                    <button class="w-100 btn btn-danger btn-lg mt-3"
type="button" @click="deleteProduct">Delete </button>
                  </div>
                  <hr class="my-4">
                </div>
              </div>
            </div>
          </div>
        </div>
      </section>
    </div>
</template>
```

Vue.js

```
<script>
import ProductDataService from '@/services/ProductDataService'
export default {
  props: ['inventory', 'removeInv', 'remove', 'updateInv'],
  data () {
    return {
      submitted: false,
      id: parseInt(this.$route.params.id),
      product: {},
      message: '',
      messageBox: false
    }
  },
  methods: {
    updateProduct () {
      ProductDataService.update(this.id, this.product)
        .then(response => {
          console.log(response.data)
          this.updateInv(this.productIndex, this.product)
          this.submitted = true
        })
        .catch(e => {
          console.log(e)
        })
    },
    deleteProduct () {
      ProductDataService.delete(this.id)
        .then(response => {
          console.log(response.data)
          this.removeInv(this.productIndex)
          this.remove(this.product.name)
          this.$router.push({ name: 'home' })
        })
        .catch(e => {
          console.log(e)
        })
    }
  },
  computed: {
    productIndex () {
      const index = this.inventory.findIndex((p) => {
        return p.id === this.id
      })
      return index
    }
  },
  mounted () {
    ProductDataService.get(this.id)
      .then(response => {
        this.product = response.data
        console.log(response.data)
      })
      .catch(e => {
        console.log(e)
      })
  }
}
```

```
</script>
```

Final App.vue

project-product/src/App.vue

Template

```
<template>
        <!-- Navigation-->
        <nav class="navbar navbar-expand-lg navbar-light bg-light">
            <div class="container px-4 px-lg-5">
                <a class="navbar-brand" href="#"><img
src="./assets/img/foodies.gif" alt="foodies"></a>
                <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target="#navbarSupportedContent" aria-
controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle
navigation"><span class="navbar-toggler-icon"></span></button>
                <div class="collapse navbar-collapse"
id="navbarSupportedContent">
                    <ul class="navbar-nav me-auto mb-2 mb-lg-0 ms-lg-4">
                        <li class="nav-item"><router-link class="nav-link active"
aria-current="page" to="/">All Products</router-link></li>
                        <li class="nav-item"><router-link class="nav-link active"
aria-current="page" to="/product-add">Add Product</router-link></li>
                        <li class="nav-item"><router-link class="nav-link active"
aria-current="page" to="/about">About</router-link></li>
                        <li>
                          <div class="input-group">
                            <input type="text" class="form-control"
placeholder="Search by type" v-model="type">
                            <button type="button" class="btn btn-outline-
secondary" @click="searchType">Search</button>
                          </div>
                        </li>
                    </ul>
                    <form class="d-flex ">
                        <button class="btn btn-outline-dark" type="button" v-
on:click="toggleSideBar">
                            <i class="bi-cart-fill me-1"></i>
                            Cart
                            <span class="badge bg-dark text-white ms-1 rounded-
pill"> {{ totalQuantity }}</span>
```

```
                        </button>
                    </form>
                </div>
            </div>
        </nav>
        <!-- Header-->
        <header class="bg-warning py-5">
            <div class="container px-4 px-lg-5 my-5">
                <div class="text-center text-white">
                    <h1 class="display-4 fw-bolder">What are you looking for
?</h1>
                    <p class="lead fw-normal text-white-50 mb-0">Choose products
now !</p>
                </div>
            </div>
        </header>
  <router-view
  :inventory="inventory"
  :addToCart="addToCart"
  :addInv="addInventory"
  :removeInv="removeInventory"
  :remove="removeItem"
  :updateInv="updateInventory"
  />
  <Sidebar
  v-if="showSideBar"
  :toggle="toggleSideBar"
  :cart="cart"
  :inventory="inventory"
  :remove="removeItem"
  />
</template>
```

Vue.js

```
<script>
import ProductDataService from '@/services/ProductDataService'
import Sidebar from '@/components/SideBar.vue'

export default {
  components: {
    Sidebar
  },
  data () {
    return {
      showSideBar: false,
      inventory: [],
      cart: {}
    }
  },
  methods: {
    toggleSideBar () {
      this.showSideBar = !this.showSideBar
    },
    addToCart (name, index) {
      if (!this.cart[name]) this.cart[name] = 0
      this.cart[name] += this.inventory[index].quantity
      this.inventory[index].quantity = 0
      console.log(this.cart[name])
    },
    removeItem (name) {
      delete this.cart[name]
    },
    addInventory (data) {
      this.inventory.push(data)
    },
    removeInventory (index) {
      this.inventory.splice(index, 1)
    },
    updateInventory (index, data) {
      this.inventory[index].name = data.name
      this.inventory[index].photo = data.photo
      this.inventory[index].price = data.price
      this.inventory[index].description = data.description
      this.inventory[index].type = data.type
    }
  },
  computed: {
    totalQuantity () {
      return Object.values(this.cart).reduce((acc, curr) => {
        return acc + curr
      }, 0)
    }
  },
  mounted () {
    ProductDataService.getAll()
      .then(response => {
        this.inventory = response.data
        console.log(response.data)
      })
      .catch(e => {
        console.log(e)
      })
    }
  }
</script>
```

Don’t forget to commit your vue.js project into git.

ctr c to stop the server.

cd ../ to return to the project-vue-node-mysql.

```
        #git add --all

        #git commit -m 'Server + vue.js CRUD V1'

        #git push
```

Navigation Schema:

When click a function the system will look into:

ProductDataService ->server/routes/product.route.js -> server/routes/controllers

To load the full project form github : https://github.com/markitosanches/project-vue-node-mysql