Advanced Web Programming

Lab 12

Backend – Create Own Web Server part 2

In lab 11 we start created web server app. You should have a MongoDB account and know about how web server works. Now start to create full stack app using my reference template example.

Exercise 1. In catalog Frontend you find example app to manage user. App is working using REST API to get the data from external Web server.
Your task is to prepare web server.

As a template I give you in catalog Backend template a web server project. This project it is a good and working solution. You should:
1. Modify my template by defined correct data model (mangoDB schema and model), routing and controllers. Remember that you server will be working with your fronted app that I give you as a starting point.
2. Create a database in your MangoDB account. Set a correct linking string to your database and account in db.js file (config directory).

As a help I give you some tutorial information how to develop web server.

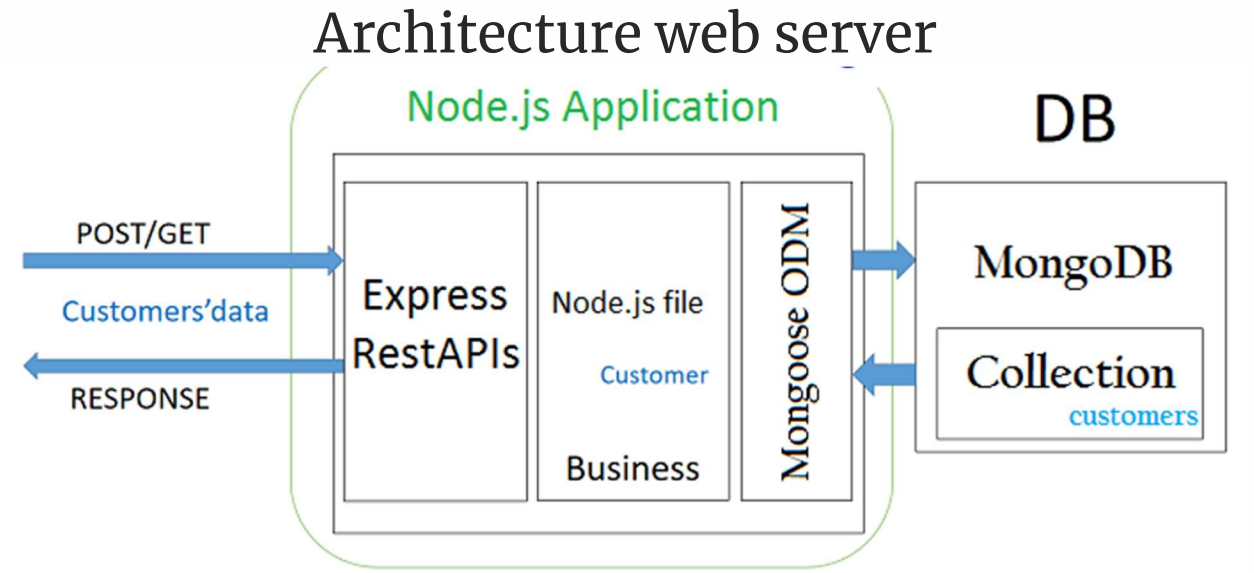
In this tutorial, we show you Node.js/EcpressJS Server example that uses Mongoose ODM to do CRUD with MongoDB and Vue.js as a front-end technology to make request and receive response.

Technologies

– Node.js/Express

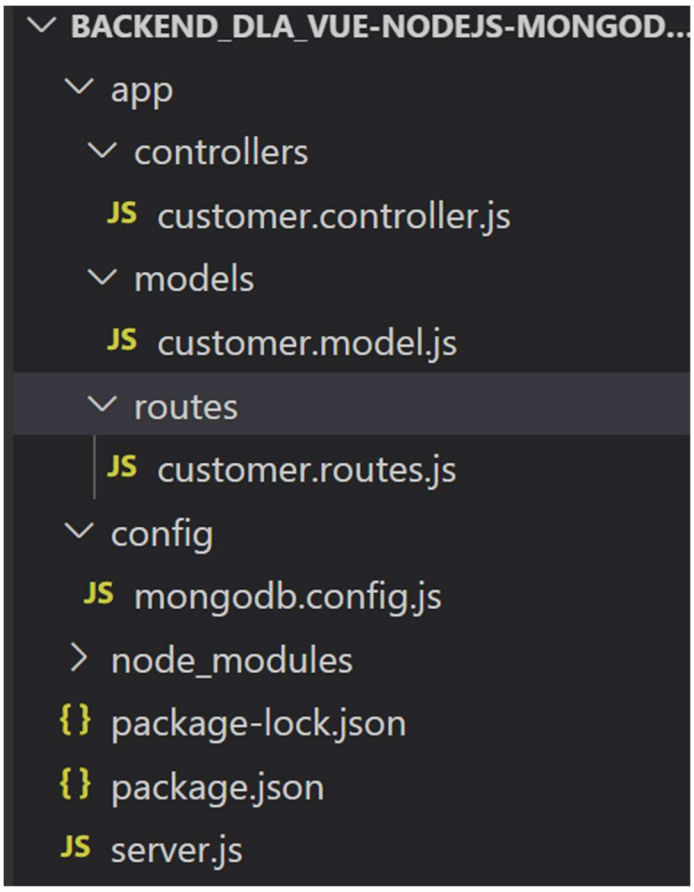
– Mongoose

– MongoDB account or install locally



CREATE TEMPLTE WEB SERVER

Web server should have the following project structure:



1. SETTING UP MONGODB CONNECTION

– Create a file `config/db.js` as below content:

```
// Export mongoose
const  mongoose = require("mongoose");

//Assign MongoDB connection string to Uri and declare options settings
//  Important!!!   set your real login and passwd in connection string
const uri =
"mongodb+srv://login:passwd@@cluster0.cxdh1.mongodb.net/testDB?retryWr
ites=true&w=majority";

// Declare a variable named option and assign optional settings
const  options = {
useNewUrlParser:  true,
useUnifiedTopology:  true
};

mongoose.Promise = global.Promise;
// Connect MongoDB Atlas using mongoose connect method
mongoose.connect(uri, options).then(() => {
     console.log("Database connection established!");
     console.log("Successfully connected to Atlas MongoDB.");
},
err  => {
{
console.log("Error connecting Database instance due to:", err);
}
})
.catch(err=>{
  console.log(err);
  console.log('Could not connect to MongoDB.');
```

```
});
```

CREATE MONGOOSE MODEL

– `./app/models/customer.model.js` file:

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

const CustomerSchema = mongoose.Schema({
    name: String,
       age: { type: Number, min: 18, max: 65, required: true },
       active: {
        type: Boolean,
        default: false
    }
});

module.exports = mongoose.model('Customer', CustomerSchema);
```

EXPRESS RESTAPIS

Route

– Define Customer's routes in `./app/routes/customer.routes.js` file:

```
module.exports = function(app) {

    var customers = require('../controllers/customer.controller.js');
```

```
    // Create a new Customer
    app.post('/api/customer', customers.create);

    // Retrieve all Customer
    app.get('/api/customers', customers.findAll);

    // Retrieve a single Customer by Id
    app.get('/api/customer/:customerId', customers.findOne);

    // Update a Customer with Id
    app.put('/api/customer/:customerId', customers.update);

    // Retrieve Customers Age
    app.get('/api/customers/age/:age', customers.findByAge);

    // Delete a Customer with Id
    app.delete('/api/customer/:customerId', customers.delete);
}
```

Controller

– Implement Customer's controller
in `./app/controllers/customer.controller.js` file:

```
const Customer = require('../models/customer.model.js');

// POST a Customer
exports.create = (req, res) => {
    // Create a Customer
    const customer = new Customer({
        name: req.body.name,
        age: req.body.age
    });
```

```
    // Save a Customer in the MongoDB
    customer.save()
    .then(data => {
        res.send(data);
    }).catch(err => {
        res.status(500).send({
            message: err.message
        });
    });
};

// FETCH all Customers
exports.findAll = (req, res) => {
    Customer.find()
    .then(customers => {
        res.send(customers);
    }).catch(err => {
        res.status(500).send({
            message: err.message
        });
    });
};

// FIND a Customer
exports.findOne = (req, res) => {
    Customer.findById(req.params.customerId)
    .then(customer => {
        if(!customer) {
            return res.status(404).send({
                message: "Customer not found with id " +
req.params.customerId
```

```
            });
        }
        res.send(customer);
    }).catch(err => {
        if(err.kind === 'ObjectId') {
            return res.status(404).send({
                message: "Customer not found with id " +
req.params.customerId
            });
        }
        return res.status(500).send({
            message: "Error retrieving Customer with id " +
req.params.customerId
        });
    });
};

exports.findByAge = (req, res) => {
       Customer.find({ age: req.params.age})
                       .then(
                              customers => {
                                     res.send(customers)
                              }
                       )
                       .catch(err => {
                              res.status(500).send("Error -> " + err);
                       })
}

// UPDATE a Customer
exports.update = (req, res) => {
    // Find customer and update it
    Customer.findOneAndUpdate({ _id: req.params.customerId }, {
```

```
        name: req.body.name,
              age: req.body.age,
              active: req.body.active
    }, {new: true})
    .then(customer => {
        if(!customer) {
            return res.status(404).send({
                message: "Customer not found with id " +
req.params.customerId
            });
        }
        res.send(customer);
    }).catch(err => {
        if(err.kind === 'ObjectId') {
            return res.status(404).send({
                message: "Customer not found with id " +
req.params.customerId
            });
        }
        return res.status(500).send({
            message: "Error updating customer with id " +
req.params.customerId
        });
    });
};

// DELETE a Customer
exports.delete = (req, res) => {
    Customer.findByIdAndRemove(req.params.customerId)
    .then(customer => {
        if(!customer) {
            return res.status(404).send({
```

```
                message: "Customer not found with id " +
req.params.customerId
            });
        }
        res.send({message: "Customer deleted successfully!"});
    }).catch(err => {
        if(err.kind === 'ObjectId' || err.name === 'NotFound') {
            return res.status(404).send({
                message: "Customer not found with id " +
req.params.customerId
            });
        }
        return res.status(500).send({
            message: "Could not delete customer with id " +
req.params.customerId
        });
    });
};
```

And now Implement main server file server.js.

When you finish start you web server typing in cmd:

```
$ node server.js
```

After that start you Fronted Vue app and test how your app works.

Exercise 2.

Now create by own web server to our clients Comments app, which you create in 8/9 lab.