We will use three packages, Express, mongoose and axios, which can be installed as follows:

$ npm install express mongoose axios –save

**Creating Database Connection**

Let’s start with the basic step to establish a DB connection. In the db folder, create a **db.js** to code to connect the database. This is just a small demo app, but the large and complex applications also have different databases for individual services.

**// db.js**

const mongoose = require('mongoose');

mongoose.connect(process.env.MONGO\_URI, {

useNewUrlParser: true,

useUnifiedTopology: true,

useFindAndModify: false,

useCreateIndex: true

}).then(() => {

console.log('Connection successful!');

}).catch((e) => {

console.log('Connection failed!');

})

In **db.js**, we have to require a *mongoose* package for MongoDB database connection.

The function **connect()** will take two arguments – *uri and options*.

**Creating Book Service**

Create a Book folder for Book service, inside the Book folder we will create **Book.js** for creating a Book Model.

**// Book.js**

const mongoose = require('mongoose');

const bookSchema = mongoose.Schema({

title: {

type: String,

require: true

},

author: {

type: String,

require: true

},

numberPages: {

type: Number,

require: false

},

publisher: {

type: String,

require: false

}

})

const Book = mongoose.model("book", bookSchema);

module.exports = Book;

Now we will need to create a server for the Book service.

**Create a Server for Book Service**

Create **books.js** inside the Book folder. Since we are learning to build microservices we will have different servers for services.

**// books.js**

require("dotenv").config();

const express = require('express');

// Connect

require('../db/db');

const Book = require('./Book');

const app = express();

const port = 3000;

app.use(express.json())

app.post('/book', (req, res) => {

const newBook = new Book({...req.body});

newBook.save().then(() => {

res.send('New Book added successfully!')

}).catch((err) => {

res.status(500).send('Internal Server Error!');

})

})

app.get('/books', (req, res) => {

Book.find().then((books) => {

if (books.length !== 0) {

res.json(books)

} else {

res.status(404).send('Books not found');

}

}).catch((err) => {

res.status(500).send('Internal Server Error!');

});

})

app.get('/book/:id', (req, res) => {

Book.findById(req.params.id).then((book) => {

if (book) {

res.json(book)

} else {

res.status(404).send('Books not found');

}

}).catch((err) => {

res.status(500).send('Internal Server Error!');

});

})

app.delete('/book/:id', (req, res) => {

Book.findOneAndRemove(req.params.id).then((book) => {

if (book) {

res.json('Book deleted Successfully!')

} else {

res.status(404).send('Book Not found!');

}

}).catch((err) => {

res.status(500).send('Internal Server Error!');

});

});

app.listen(port, () => {

console.log(`Up and Running on port ${port} - This is Book service`);

})

We have used express for creating a server, made the db file required for connecting with the database, and Book model for storing data.

The Book service has four routes:

* Add books
* Get all the books from the database
* Get a particular book
* Delete a book

Run book service at **port 3000**.

**Creating Customer Service**

Create a Customer folder for Customer service, inside the Customer folder we will have *Customer.js* for model, just like we did for Book service in the previous section.

**// Customer.js**

const mongoose = require('mongoose');

const CustomerSchema = mongoose.Schema({

name: {

type: String,

require: true

},

age: {

type: Number,

require: true

},

address: {

type: String,

require: true

}

})

const Customer = mongoose.model("customer", CustomerSchema);

module.exports = Customer;

Now we will need to create a server for Customer service.

**Create a Server for Customer Service**

We will have a separate server for Customer service as well. Create a file named *customer.js*.

**// customer.js**

require("dotenv").config();

const express = require('express');

// Connect

require('../db/db');

const Customer = require('./Customer');

const app = express();

const port = 5000;

app.use(express.json())

app.post('/customer', (req, res) => {

const newCustomer = new Customer({...req.body});

newCustomer.save().then(() => {

res.send('New Customer created successfully!');

}).catch((err) => {

res.status(500).send('Internal Server Error!');

})

})

app.get('/customers', (req, res) => {

Customer.find().then((customers) => {

if (customers) {

res.json(customers)

} else {

res.status(404).send('customers not found');

}

}).catch((err) => {

res.status(500).send('Internal Server Error!');

});

})

app.get('/customer/:id', (req, res) => {

Customer.findById(req.params.id).then((customer) => {

if (customer) {

res.json(customer)

} else {

res.status(404).send('customer not found');

}

}).catch((err) => {

res.status(500).send('Internal Server Error!');

});

})

app.delete('/customer/:id', (req, res) => {

Customer.findByIdAndRemove(req.params.id).then((customer) => {

if (customer) {

res.json('customer deleted Successfully!')

} else {

res.status(404).send('Customer Not Found!');

}

}).catch((err) => {

res.status(500).send('Internal Server Error!');

});

});

app.listen(port, () => {

console.log(`Up and Running on port ${port}- This is Customer service`);

})

Create the same four routes for Customer service as we did for the Book service

Run Customer service at **port 5000**.

**Creating Order Service**

Create an *Order.js file* inside the Order folder for model, just like we did for Book and Customer service.

**// Order.js**

const mongoose = require('mongoose');

const orderSchema = mongoose.Schema({

customerID: {

type: mongoose.SchemaTypes.ObjectId,

require: true

},

bookID: {

type: mongoose.SchemaTypes.ObjectId,

require: true

},

initialDate: {

type: Date,

require: true

},

deliveryDate: {

type: Date,

require: false

}

})

const Order = mongoose.model("order", orderSchema);

module.exports = Order;

Now we will need to create a server for the Order service.

**Create a Server for Order Service**

Create order.js inside the Order folder.

**// order.js**

require("dotenv").config();

const express = require('express');

const mongoose = require("mongoose");

const axios = require('axios');

// Connect

require('../db/db');

const Order = require('./Order');

const app = express();

const port = 9000;

app.use(express.json())

app.post('/order', (req, res) => {

const newOrder = new Order({

customerID: mongoose.Types.ObjectId(req.body.customerID),

bookID: mongoose.Types.ObjectId(req.body.bookID),

initialDate: req.body.initialDate,

deliveryDate: req.body.deliveryDate

});

newOrder.save().then(() => {

res.send('New order added successfully!')

}).catch((err) => {

res.status(500).send('Internal Server Error!');

})

})

app.get('/orders', (req, res) => {

Order.find().then((orders) => {

if (orders) {

res.json(orders)

} else {

res.status(404).send('Orders not found');

}

}).catch((err) => {

res.status(500).send('Internal Server Error!');

});

})

app.get('/order/:id', (req, res) => {

Order.findById(req.params.id).then((order) => {

if (order) {

axios.get(`http://localhost:5000/customer/${order.customerID}`).then((response) => {

let orderObject = {

CustomerName: response.data.name,

BookTitle: ''

}

axios.get(`http://localhost:3000/book/${order.bookID}`).then((response) => {

orderObject.BookTitle = response.data.title

res.json(orderObject);

})

})

} else {

res.status(404).send('Orders not found');

}

}).catch((err) => {

res.status(500).send('Internal Server Error!');

});

})

app.listen(port, () => {

console.log(`Up and Running on port ${port} - This is Order service`);

})

Run customer service at **port 9000**.

We will create three routes in Order service:

* Add an order. You need to pass *bookId, customerId, initialDate, and deliveryDate* for it.
* Get all the orders from the database
* Get customers and book details for a particular order.

To run the last route make sure all the services are running in the background. Because we are getting details from book service and customer service as well.

<https://arunrajeevan.medium.com/understanding-mongoose-connection-options-2b6e73d96de1> /////to understand mongodb connections