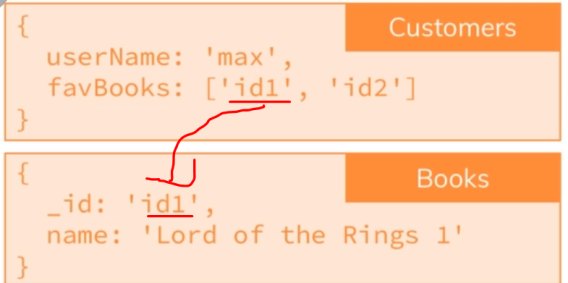
* Mongo Db

Mongo db dox : <https://docs.mongodb.com/manual/>

Main Page of dox: <https://docs.mongodb.com/manual/release-notes/4.4/#projection>

It stores a data in JSON (BSON) format (B = binary)

In mongodb we can call our data from different collections



So to add mongo db instead of Mysql (sequelize) we have to change in the database and we can do it by require mongo db and because the mongo db client is the one who connects with database so

const mongodb = require('mongodb');

const MongoClient = mongodb.MongoClient;

now after require above we just have to connect mongo client and it work as a function

const mongoConnect = (callback) => {

    MongoClient.connect('mongodb+srv://**mayank:Abcd4321**@cluster0.f5vog.mongodb.net/**shop**?retryWrites=true&w=majority')

    .then(client => {

        \_db = client.db(); // wait for the call back to return

        callback();

    })

    .catch(err => {

        console.log(err);

        throw err;

    });

}

So above we just connecting mongo client in the call back so it can return the call back when this function execution completed and so we can excess the client database in side \_db . the above operation is used for connectivity purpose . After that we have to get the data base for crud operation in it . the below one returns the data base of the client so we can do crud operation

// it is for crud operation in data base

const getDb = () => {

    if(\_db) {

        return \_db;

    }

    throw 'No database found';

}

exports.mongoConnect = mongoConnect;

exports.getDb = getDb;

const mongoConnect = require('./util/dataBase').mongoConnect;

mongoConnect(() => {

    app.listen(3000);

})

Write now we are not using any callbacks in main app.js

Now we have to change our product models the way of interacting with database (mongodb)

class Product {

  constructor(title,price,description,imageUrl, id) {

      this.title = title;

      this.price =price;

      this.description = description;

      this.imageUrl = imageUrl;

      // check if id is present or not (product is updated or new)

**this.\_id = id ?new mongodb.ObjectID(id) : null;** //it convert into binary so not considered in this for save

  }

// insert one will insert an object while insertMany can insert array

  save() {

    const db = getDb();

    let dbOp;

    if(this.\_id) {

      dbOp = db.collection('products')

      .updateOne({\_id: this.\_id}, {$set: this})// if id exist set all this

    }

    else {

      dbOp = db.collection('products').insertOne(this);

    }

    return dbOp

           .then(result => {

           console.log(result);

           })

           .catch(err => console.log(err));

Id is used to edit and delete product but lets see first the add product first time down below

Now we change post add product first in our admin.js page of controller

exports.postAddProduct = (req, res, next) => {

  const title = req.body.title;

  const imageUrl = req.body.imageUrl;

  const price = req.body.price;

  const description = req.body.description;

  const product = new Product(title,price,description,imageUrl);

  product.save()

  .then(() => {

    console.log('product added')

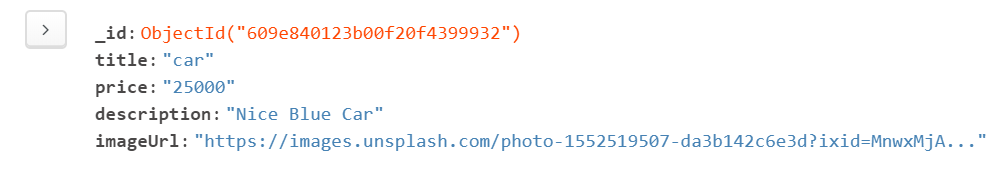
    res.redirect('/admin/products')

  })

  .catch(err => console.log(err))

};

Download mongo compass for GUI of cloud in your system after connecting you when you run

**** data created in JSON format

Now we create a fetch all static function of product so that we can use it in show product to our user

static fetchAll() {

    const db = getDb();

**// because find will return data one by one so it will go down if not use to array// to array holds the incoming values from find and when done then go next**

    return db.collection('products').find().toArray()

       .then(products => {

         console.log(products);

         return products;

       })

       .catch(err => console.log(err));

  }

To create find by id method in product so you can compare product id with data base and get that particular product id but you can not compare prodId directly because the id is store in BJON formate which is binary and to change your prodId in string format so change it in binary you have to call mongo

const mongodb = require('mongodb');

so that you can get the object id of same and the object name in mongo called and **starting from \_id**

 static findById(prodId) {

    const db = getDb();

    return db.collection('products')

      .find({**\_id**: new mongodb.ObjectID(prodId)}) //conversion to binary

      .next()

      .then( product => {

        console.log(product)

        return product;

      })

      .catch(err => console.log(err));

  }}

Replace all the product .id with **product.\_id** in view folder

And now we have to add delete single product fetcher in product model

 static deleteById(prodId) {

    const db = getDb();

    return db

    .collection('product')

**.deleteOne({\_id: new mongodb.ObjectID(prodId)})**

    .then(() => {

      console.log('deleted');

    })

    .catch(err => console.log(err));

Now we create a user in our mongo db compass by our own , we are doing it for just now until we learn about authentication later we will do it properly. Let suppose user is created

But user must be related to product as our product contains userId as foreign key so for that we pass user id at the time of adding product as an argument and save it too.

// saving the user details in the particular req

app.use((req,res,next)=> {

    user.findById('609fc0b9928899dd02056e98')

    .then(User => {

**req.user = new User(user.name, user.email, user.cart, user.\_id)**

        next(); // necessary

    })

    .catch(err => console.log(err));

})

In app.js we just copy the id from mongodb which we created in compass and finding in the user module

static findById(prodId) {

        const db = getDb();

        return db.collection('users')

**.findOne({\_id: new mongodb.ObjectID(prodId)})**

          .then(User => {

              console.log(User);

              return User;

          })

Do now we work on cart and cart items

As we know there is no schema in mongoDb(no sql) so we justremove our cart.js and carditem.js files from the module because we donot create a cart and cart item database separately instead we embed or pass the cart and its item in the user database because the user is the one who use cart and its item so what we can do is give the specific id for the cart to user in which it contains all the products id that needed.

addToCart(product) {

    const cartProductIndex = this.cart.items.findIndex(cp => {

      return cp.productId.toString() === product.\_id.toString();

    });

    let newQuantity = 1;

    const updatedCartItems = [...this.cart.items];

    if (cartProductIndex >= 0) {

      newQuantity = this.cart.items[cartProductIndex].quantity + 1;

      updatedCartItems[cartProductIndex].quantity = newQuantity;

    } else {

      updatedCartItems.push({

        productId: new ObjectId(product.\_id),

        quantity: newQuantity

      });

    }

    const updatedCart = {

      items: updatedCartItems

    };

    const db = getDb();

    return db

      .collection('users')

      .updateOne(

        { \_id: new ObjectId(this.\_id) },

        { $set: { cart: updatedCart } }

      );

  }

So we create a reference in our user database so the it takes the id as reference and can call it

