UNIVERSIDADE CATÓLICA DOM BOSCO
Valorizando talentos

Tópicos Avançados em Programação I

Prof. Me. Marcos Alves

marcos@ucdb.br

LABORATÓRIO – parte 2

Finalizando *a estrutura* ***Back-End***

– API com Web Services, JavaScript e MongoDB

O objetivo deste laboratório é colocar em prática os conhecimentos abordados durante o semestre. Este será o **ÚNICO TRABALHO** a ser utilizado para avaliação do aluno. Parte da nota será dada pelas entregas parciais durante as próximas aulas e apresentação da versão final conjunta com a disciplina de RAD1.

Para tanto, cada aluno deverá:

1º. Vamos completar nosso projeto com a *collection* Pedido.

2º. *Schema* para *collection* Pedido

Nesta *collection* entrego o *schema* completo, pois é mais complexo, e é primeira vez que faremos uma implementação de vínculo entre *collections* e também do tipo *one-to-many* (1:N).

```
    produtos: [{
        type: Schema.Types.ObjectId,
        ref: "Produto"
    }],
   dataPedido: { type: Date, require: true, default: Date.now },
   client: {
        type: Schema.Types.ObjectId,
        ref: "Client"
    },
    status: { type: String, require: true, default: "pendente" }
```

3º. Sobre as Rotas de Pedido

Vou entregar as rotas prontas, porém, nem todas as *functions* dos *services* estão implementadas. É tarefa sua identificá-las e fazer a respectiva implementação no arquivo **pedido-services.js**

```
const express = require('express')
const router = express.Router()
const PedidoService = require('../services/pedido-service.js')
const pedidoService = new PedidoService()
```

```
router.post('/incluir', async (req, res) => {
    let data = { produtos: req.body.produtos, dataPedido: req.body.dataPedido,
                 client: req.body.client, status: req.body.status }
        let newPedido = await pedidoService.save(data)
    res.json(newPedido)
})

router.get('/listar', async (req, res) => {
    let pedido = await pedidoService.list()
    res.json(pedido)
})

router.get('/listar/:status', async (req, res) => {
    let status = req.params.status
    let pedidos = await pedidoService.listPedidoStatus(status)
    res.json(pedidos)
})

router.get('/:id', async (req, res) => {
    let id = req.params.id
    let pedido = await pedidoService.getPedido(id)
    res.json(pedido)
})

router.get('/pedidoClient/:id', async (req, res) => {
    let id = req.params.id
    let album = await pedidoService.listPedidoDeCliente(id)
    res.json(album)
})


router.delete('/delete/:id', async (req, res) => {
    let pedido = await pedidoService.delete(req.params.id)
    res.json({ msg: "pedido deletado com sucesso", pedidoDeleted: pedido })
})

router.delete('/deleteAllPedido/:idClient', async (req, res) => {
    await pedidoService.deleteAllPedidos(req.params.idClient)
    res.json({ msg: "Pedidos deletados com sucesso" })
})

router.put('/:id', async (req, res) => {
    let id = req.params.id
    let pedido = { produtos: req.body.produtos, dataPedido: req.body.dataPedido,
        client: req.body.client, status: req.body.status }
    let pedidoUpdate = await pedidoService.update(id, pedido)
    res.json({ select: pedidoUpdate })
})
```

```
router.put('/status/:idPedido', async (req, res) => {
    let id = req.params.idPedido
    let status = req.body.status
    const pedido = await pedidoService.updateStatus(id, status)
    return res.json(pedido)
})

module.exports = router
```

4º. Sobre os Services de Pedido

Aqui apenas algumas das *functions* (marcadas em amarelo no item anterior), as demais estão citadas nas rotas apenas e devem ser implementadas aqui.

```
const Pedido = require('../models/pedido-model.js')
module.exports = class PedidoService {
    async save(data) {
        return await Pedido(data).save()
    }

    async list() {
        return await Pedido.find({})
    }

    async delete(id) {
        return await Pedido.findByIdAndDelete(id)
    }

    async update(id, update) {
        return await Pedido.findByIdAndUpdate(id, update)
    }

    async listPedidoStatus(s) {
        const regex = new RegExp(s, 'i') // i for case insensitive
        return await Pedido.find({ status: { $regex: regex } })
    }
}
```

5º. Sobre o Server.js

Está na íntegra aqui.

```
require('dotenv').config();

const clientRoutes = require('./routes/client-routes.js')
const userRoutes = require('./routes/user-routes.js')
const produtoRoutes = require('./routes/produto-routes.js')
const pedidoRoutes = require('./routes/pedido-routes.js')

const express = require('express');
const app = express();
const port = process.env.PORT || 3001;
const mongoose = require('mongoose');
const { response } = require('express');

app.use(express.json());
app.use(function (req, res, next) {
    res.header("Access-Control-Allow-Origin", "*");
    res.header("Access-Control-Allow-Methods", "GET, PUT, POST, DELETE");
    res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-
Type, Accept");
    next();
});

mongoose.connect(process.env.DB_HOST, { useNewUrlParser: true, useUnifiedTopology: t
rue });
mongoose.set('useFindAndModify', false);

app.get("/",  (req, res)=>{ res.send("TADS API Rodando...") });

app.use('/client', clientRoutes)
app.use('/user', userRoutes)
app.use('/produto', produtoRoutes)
app.use('/pedido',pedidoRoutes)

app.listen(port, () => {
    console.log(`APP listening at http://localhost:${port}`)
});
```